지능형 로봇 시스템을 위한 다중센서 융합에 의한 Task-Level Robot Language의 설계 및 구현

Design and Implementation of Task-Level Robot Languages for Intelligent Robot System.using Multi-Sensor Integration

徐 一 弘* · 呂 熙 珠**
(Il Hong Suh · Hee-Joo Yeo)

社團法人 大韓電氣學會
THE KOREAN INSTITUTE OF ELECTRICAL ENGINEERS

지능형 로봇 시스템을 위한 다중센서 융합에 의한 Task-Level Robot Language의 설계 및 구현

論 文
45~6~18

Design and Implementation of Task-Level Robot Languages for Intelligent Robot System using Multi-Sensor Integration

徐 一 弘* · 呂 熙 珠**
(Il Hong Suh · Hee-Joo Yeo)

Abstract - *Task-Level Robot Languages* for an intelligent robot system are proposed for the object-oriented task-description and programming. For this, several manipulator-level robot languages are extended for the robot to perform sensor-based motions such as the tracking of a moving target and on-line path modification by using vision or F/T sensor. And a *Task-Level Robot Language Interpreter* is designed to translate a task level robot program into a set of manipulator level languages including basic motion control commands and sensor based motion control commands. The control system is implemented by using VME-based 32-bit microprocessor boards and a realtime multitasking operating system(VxWorks), and verified by a practical experiment, in which the system controls two industrial robots to successfully execute a coordinated and concurrent motion.

Key Words : Sensor-Based Motion, Manipulator-Level Language, Task-Lovel Language

1. 서 론

산업 사회의 발달속에서 세계 각국은 자국의 시장 경쟁력을 높이기 위해 어느 때보다도 기술개발에 많은 투자와 노력을 기울이고 있다. 현재 산업 현장에서 생산 자동화를 추진함에 있어서의 핵심 기술은 로보트를 이용한 생산 공정의 자동화이며 그 기능이 보다 향상되고 복잡해짐에 따라 작업장내에 있는 여러 로보트 뿐만 아니라 주변장치들을 동시에 제어할 수 있고 주변의 환경 변화에도 쉽게 적응할 수 있는 로보트 제어 시스템에 대한 필요성이 점차 증대되고 있다[1, 2].

이와 함께 로보트 시스템을 편리하고 효과적으로 동작시키기 위한 로보트 제어 언어의 개발에도 관심을 기울이게 되었다. 산업용 로보트는 특정한 생산 작업을 위해 재프로그램할 수 있는 기계장치(Reprogrammable Device)이기 때문에 적절한 Programming Tool의 사용 여부에 따라 로보트의 지능화, 고기능화를 가능하게 하는 성능 지표로 되어 버렸다[6, 8].

기존 로보트 제어언어(Robot-Level Language)는 사용자가 로보트의 기본동작 제어 명령만으로 작업을 일일이 프로그램해야 한다는 어려움이 있을 뿐만 아니라, 응용할 수 있는 작업이 제약되어 있다. 또한, 기존 대부분의 로보트 언어가 로보트를 중심으로 작업을 교시하기 편리하게 만들어져 있으므로 전문 프로그래머나 시스템 개발자가 아닌 일선 작업자가 자유로이 프로그래밍을 하기에는 어려움이 남아 있다. 따라서 이러한 문제에 대한 해결책으로 로보트에 종속적인 요소를 배제시키고 작업 대상물을 중심으로 작업을 기술, 프로그래밍 하고자 하는 Task-Level Language에 대한 연구가 진행되어 왔다. Task-Level Language는 로보트 작업의 시작 상태, 종료 상태와 작업 대상물의 중간 상태를 연속적으로 프로그래밍 함으로써 로보트의 기구학이나 경로를 사용자가 따로 정의할 필요가 없는 강력한 Power를 제공하는 새로운 형태의 로보트 언어이다. 1977년 IBM사의 Lieberman과 Wesley에 의해 조립 시스템의 자동화를 위해 사용자가 좀 더 친숙한 문법과 단어를 사용한 프로그래밍과 기존의 조립 시스템을 집약시킨 AUTOPASS[3]가 연구 개발된 것이 Task-Level Language의 시초이다. 그 이후로 스코틀랜드 Edinburgh 대학에서의 RAPT 개발 등으로 끊임없이 연구가 진행되어 왔으나 Task-Level Language의 구현에는 여러가지 어려움이 남아 있었다. 작업 환경의 기하학적 모델링, 조립순서의 자동생성, 장애물 충돌회피, 힘 센서를 이용한 힘 제어 등이 선행 연구되어야 할 과제들로 이들 연구 결과에 대한 결합이 있어야 Task-Level 프로그래밍이 가능해지므로 아직까지는 실제 시스템에 적용 가능한 로보트 제어언어의 개발이 힘든 실정이다[3, 6].

따라서, Task-Level Language를 개발하기 위해서는 기존 Robot-Level Language에서 로보트가 변화하는 외부 환경에 능동적으로 대처하며 좀더 다양한 종류의 작업에 응용할 수 있도록 Sensor-Based Robot Motion Control의 기능 강화가 필수적이다. 따라서, 본 논문에서는 Sensor-Based Robot Control 기능을 강화하여 지능형 로봇을 위한 Task-Level Robot Language 개발로 접근하고자 한다. 이러한 Task-Level Language의 개발을 위해서 각 부품의 형상, 위치 및 상호 결합 관계를 모델링 과정을 통하여 데이터 베이스를 구축한 뒤 Task-Level Language로 기술된 프로그램이 Robot-Level Language로 변환되는 과정 및 실행 과정에서 참조될 수 있게 하였다. 이는 전혀 새로운 형태의 로보트 제어언어가 아니라, 기존에 개발한 로보트 중심의 제어언어(Robot-Level Language)[13]를

* 正 會 員 : 漢陽大 工大 電子工學科 敎授 · 工博
서울大 制御計測 新技術硏究 센터 硏究委員
** 正 會 員 : 漢陽大 大學院 電子工學科 博士課程
接受日字 : 1996年 1月 29日
最終完了 : 1996年 5月 10日

토대로 Sensor-Based Robot Motion Control의 기능을 강화하고 이를 조합하여, 각 Task에 따른 Task-Level 명령어 Set을 구성하여 로보트의 작업을 편리하게 기술할 수 있는 방법을 연구함으로써 좀 더 발전된 형태의 로보트 제어언어를 제시하고자 한다.

이를 위하여 본 연구에서는 신뢰성과 확장성이 뛰어난 CPU 30(32Bit CPU 보드), VME BUS와 실시간 운영체계(Real-Time O.S)인 VxWorks를 토대로 하여 동시 동작 및 분산처리 구조를 갖도록 하였으며, 이러한 시스템에 Ethernet을 통해 SUN Workstation과 연결하여 확장성 및 유연성이 높은 전체 시스템을 구축하였다.

2. 온-라인 경로보정을 위한 Sensor-Based Robot Language의 설계

2.1 온-라인 경로보정을 위한 Language Set 및 구조

기존 산업용 로보트에 사용되는 센서로는 로보트 위치나 속도 제어를 위한 레졸버, 엔코더, 타코메터와 홈 위치 및 리미트 위치 감시를 위한 센서등이 대부분이었다. 그러나, 정밀 조립작업이나 용접 작업과 같은 작업을 수행해야 할 경우 정밀 운동과 경로보정을 위해서는 힘, 접촉, 비젼 센서등이 필요하다. 예를 들면, Peg-In-Hole과 같은 조립작업의 경우 교시한 위치와의 오차가 생기게 되면 센서를 이용하지 않는 로보트의 경우는 에러를 발생하게 되지만, 센서를 사용한 로보트의 경우 이러한 오차에 대해 능동적으로 대처할 수 있다. 원하는 위치가 고정되어 있을 경우, 주어진 경로를 따르면서 매 샘플링 타임마다 외부 센서 데이터에 의해 로보트 매니퓰레이터의 경로를 보정하기 위해 기존의 **movl** 명령어에 위치 및 자세를 보정하도록 확장하였다. 여기서, filename은 sensory motion시에 참조하는 reference file이고, path_mod는 input port#10을 통해 전달되는 위치와 자세의 보정량으로 새로운 옵션에 의해 4x4 행렬 형태로 내부적으로 정의하였다. **movl**에 대한 Syntax는 다음과 같다.

```
define inport10 path_mod with file=filename;
movl robot1 to loc1 with correction=path_mod sp=50;
```

이때, 힘 제어시에 참조하는 File은 force.ref이고, 이는 User에 의해 정의되는 File로 힘제어에 사용되는 각종 Option Command들이다.

<force.ref>

① Desired Force : F_d

② Environmental Stiffness : K_e

③ Selection Matrix : S

movl 명령어의 센서 신호 융합을 위한 구조적인 모델은 그림 1과 같이 센서 CPU 보드와 필요한 몇몇 센서 신호처리 유니트로 구성되어 있다. 이때, 센서 유니트는 원시 데이타(Raw Data)를 적절한 데이타 포멧으로 변환하여 메인 CPU 보드로 전송한다. 이때, 센서의 선택은 **define** 명령어를 이용하여 사용자에 의해 정의되고, 다음번 목표 위치는 매 40 msec마다 식(1)에 의해 계산되고, 여기에서 Correction_value를 계산하기 위한 제어 알고리즘은 3절에서 설명된다.

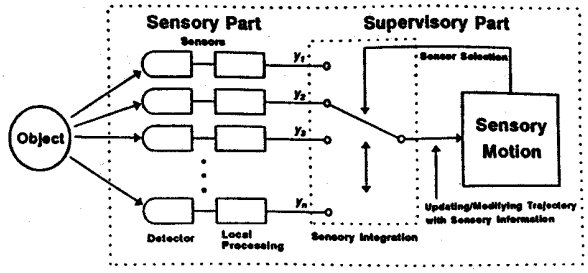

그림 1 센서 융합을 위한 구조적인 모델
Fig. 1 A Schematic Model for Sensory Integration

Next_position = Calculated_position + Correction_value (1)

여기서, Calculated_position은 매 Sampling Time마다 초기 위치와 목표위치 사이를 직선보간 할때 얻어지는 새로운 목표 위치이고, Correction_valu은 외부센서에 의한 수정량이다.

그와 같은 movl-with-correction 명령어는 그림 2와 같이 용접 작업이나 그라인딩 작업에서 아크나 비젼 센서를 사용하여 용접 경로를 보정하기 위해 필요한 정보를 제공해야하는 작업에 유용하게 적용될 수 있다.

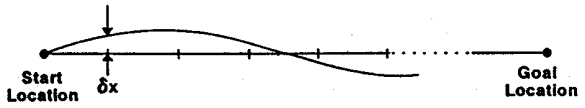

그림 2 온라인 경로 보정을 위한 구조
Fig. 2 A Scheme for On-Line Path Modification

2.2 온-라인 경로보정 제어 알고리즘

현재 주로 사용하는 산업용 로보트는 미리 위치를 알려준 다음 로보트의 엔드-이펙트를 위치 제어함으로써 주어진 일을 수행하고 원하는 동작을 수행한다. 그러나 조립작업이나 협조작업과 같이 상호간에 힘이 발생하고, 주변 환경을 잘 알 수 없는 상황 속에서는 위치 제어만으로 정확히 로보트를 제어할 수 없으므로 주변 환경과 로보트의 엔드-이펙트 사이에 접촉이 일어났을 때 정확한 로보트 매니퓰레이터의 제어를 위해서는 위치 제어의 단점을 보완하여 힘 제어를 행하여야만 한다.

기존의 위치/힘 제어 알고리즘은 회전축의 토오크를 출력 값으로 하기 때문에 각변위값을 입력으로 하는 상용의 로보트 제어에 사용되고 있는 위치 제어기를 사용할 수 없고 새로운 제어기를 구성해야 한다는 단점을 갖고 있다. 그래서 본 연구에서는 기존의 위치 제어기에 덧붙여서 수행할 수 있는 하이브리드 위치/힘 제어 알고리즘을 이용하여 퍼지 제어기를 설계하였다. 엔드-이펙트가 주변 환경과 접촉을 이루고 있을 경우 그 주변 환경, 엔드-이펙트, 센서등이 받는 힘과 위치 변위값에 비례관계가 성립하는 강성(Stiffness)값을 다음과 같이 유도한다.

$$\delta F = K_X \delta X \tag{2}$$

여기서, δX = 직교 좌표상에서의 위치 변위값

δF = 직교 좌표상에서의 앤드-이펙트의 힘 변위값

K_X = 직교 좌표상에서의 센서, 앤드-이펙트, 주변 환경등의 전체 강성

이된다. 이때, 환경에 대한 강성(K_X)은 위(2)번식을 이용하여, 로봇을 δX 변위 만큼 움직였을때 Feedback되는 Force Data를 여러번 반복 측정하여 평균치로 구할 수 있는데, 이는 힘 제어에서 중요한 파라미터이다. 이를 위하여 [14]에서는 환경 변화에도 온라인으로 Stiffness 값을 추정하도록 하는 Fuzzy Adaptive Stiffness Estimator(FASE)를 제안하여 좋은 힘 제어 성능을 얻을 수 있도록 하였다.

$$\tau = J^T \delta F \tag{3}$$

여기서, J^T= 자코비언의 전치 행렬

τ = 각 회전축의 토오크 값

이므로, 식(2)를 식(3)에 대입하면,

$$\tau = J^T K_X \delta X = J^T K_X J \delta\theta = K \delta\theta \tag{4}$$

를 구할 수 있다. 따라서,

$$\delta\theta = K^{-1} \tau,\ K = J^T K_X J \tag{5}$$

가 된다. 여기서, K_X는 회전축 상에서의 강성(Stiffness)이며, K_X^{-1}는 회전축 상에서의 유연성(Compliance)이다. 결과적으로 식(5)를 이용하여 힘 제어기의 출력 토오크 값을 각 회전축의 각변위값으로 변환시키는 방법을 사용하면 위치 제어기를 갖는 상용의 로보트 제어기를 힘 제어에 그대로 이용할 수 있다. 한편, 위치 제어는 기존의 위치 제어기 자체 내에 위치, 속도 및 전류 페루프가 있으므로 이것을 이용하면 새로운 제어기를 갖출 필요 없이 입력 명령값인 직교 좌표축상의 위치 변화값이 개루프를 형성하므로써 가능하다. 최종적으로 힘 제어 루프의 출력값인 위치 축상의 변위값과 위치 제어기의 출력값인 위치 변위값을 합한 최종 변위값을 기존의 위치 제어기에 입력 명령값으로 하여 위치와 힘을 동시에 제어하게 된다.

그림 3은 상용의 위치 제어기를 이용할 수 있는 하이브리드 위치/힘 제어 알고리즘을 이용하여 힘 제어 루프에 퍼지 제어기를 적용하였을 때의 블럭도이다. PID 제어기의 오버슈트와 정착 시간의 개선, J, K_x의 Uncertainty 보상 및 좋은 힘 응답을 위하여 퍼지 제어기를 도입하였다. PID 제어기에서의 스텝 응답에 대해 얻은 정량적인 데이타를 이용해 상승시간, 오버슈트 및 장착 시간을 개선하도록 [14]에서와 같이 If-then 형태의 언어적인 규칙을 결정하고 퍼지 제어기가 힘 오차(Error)와 오차 변화량(Error Change)에 대해 제어되어야 할 힘의 양을 출력하도록 설계하였다.

이 블럭도에서 볼 수 있듯이 퍼지 제어기의 입력은 힘의 오차와 오차의 변화율이며 값들이 양자화되고 추론 과정을 거쳐 제어되어야 할 힘의 양을 출력하게 된다. 그리고 강성(Stiffness) 행렬의 역행렬을 곱하여 위치 변위량으로 변환하고 자코비언(Jacobian) 행렬의 역행렬을 곱하여 각 변위값으로 변환

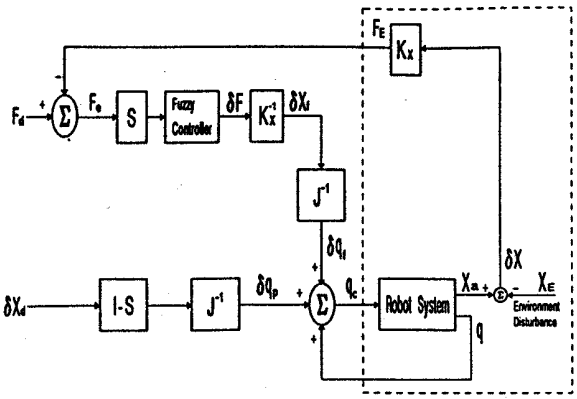

그림 3 퍼지 제어기를 이용한 하이브리드 위치/힘 제어 블럭도

Fig. 3 Block Diagram of Hybrid Position/Force Control using Fuzzy Controller

된 후 비로소 로보트에 명령으로 주게 된다. 퍼지 제어기는 퍼지 레벨의 세분화가 증가할수록 제어기의 출력값을 만들기 위한 추론법칙의 세분화가 증가하므로써 실시간 제어가 불가능해진다. 이러한 문제점을 해결하기 위하여 퍼지법칙을 양자화하여 룩업 테이블화 함으로써 추론 과정을 제거하여 실시간 제어를 할 수 있었다.

2.3 온-라인 경로보정 실험 : Surface Tracking

본 논문에서 사용된 온-라인 경로보정 실험을 위하여 힘센서 시스템[17]을 장착한 5축 수직 다관절 로보트 매니퓰레이터(PT200)를 이용하였다. 제어 알고리즘은 C 언어로 구현하였으며, 다중 로보트 제어기[13]로 실험하였다. 여기에 한 개의 32-비트 마이크로프로세서 보드(FORCE30[17])가 추가로 사용되었다. 힘과 토오크의 센싱을 위해 JR^3사의 6자유도 힘/토크 센서 시스템을 로보트의 앤드-이펙트에 장착하여 사용하였다.

힘 센서 각 성분 데이타의 1unit은 힘의 경우 $1uf = 10gf$, 토크의 경우 $1ut = 20gfcm$을 나타낸다. 일반적으로 대부분의 상업화된 산업용 매니퓰레이터는 RS-232C 시리얼 통신을 통해서 9600bps의 속도로 힘 센서로부터 정보를 얻기 때문에 약 30msec가 소요된다. 따라서, 본 논문에서는 보다 빠른 힘 응답을 위해서 VME용 Parallel 통신 보드를 제작하여 10msec의 전송속도, 샘플링을 40msec로 하였다.

앞에서 설명한 온-라인 경로보정 명령어인 **movl-with-correction**와 기본동작 명령어를 이용하여, 그림 4, 5와 같이 임의의 곡면을 따라 이동하는 Surface Tracking 작업을 다음과 같이 프로그램하여 실험하였다. 이때, 로보트는 $4\times10^4 N/m$의 강성(Stiffness)을 가지는 고정된 외부 환경에 대해 앤드-이펙트의 Z축에 대해 수직인 평면에 일정한 힘 ($500ut = 49N$)을 유지하면서, -X축 방향으로 곡면을 Tracking하는 실험이다. 본 연구에서는 최고 $10^5 N/m$의 강성을 갖는 물체에 대하여 힘 제어 알고리즘을 확인하였지만, 만약 강성(Stiffness)이 큰 경우는 조그만 위치 오차가 큰 힘을 발생시켜 로봇에게 큰 손상시킬 수 있으므로 추가적으로 고려되어야 하는 어려운 제어문제이다. 그림6의 Surface Tracking 응답곡선을 보면 PID 제어기의

경우 정상상태 오차가 있지만 퍼지 제어기의 경우 진동은 있지만 정상상태 오차는 거의 없음을 알 수 있다.

```
define inport1 force1 with file = force.ref;
movj robot1 to loc50 with sp = 10;
movl robot1 to loc51 with correction = force1;
end
```

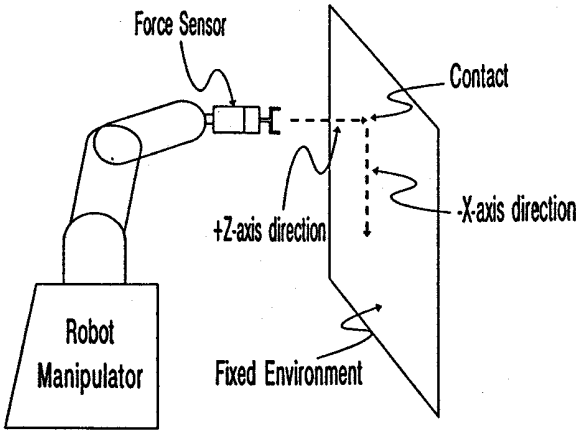

그림 4 하이브리드 위치/힘 제어시의 실험 환경
Fig. 4 Environmental Setup for Hybrid Position/Force Control

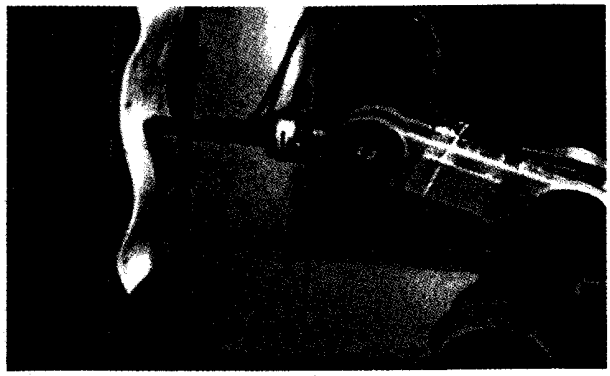

그림 5 Surface Tacking의 구현
Fig. 5 Implementation of Surface Tracking

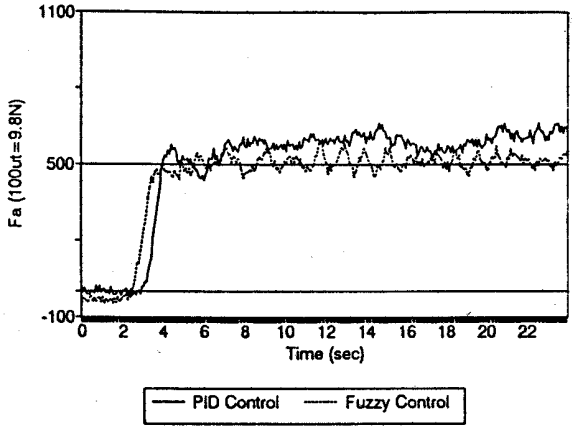

그림 6 Surface Tacking시의 힘 응답
Fig. 6 Force Response in Surface Tracking

3. 이동물체 추적을 위한 Sensor-Based Robot Language의 설계

3.1 이동 물체 추적을 위한 Language Set 및 구조

몇 개의 주어진 조건이 만족할 때까지 외부 센서 데이타에 의해 그림 7에서와 같이 이동물체를 추적하기 위하여 **move** 명령을 구현하였다.

여기에서 로보트나 외부 디바이스가 움직이는 위치는 Position Equation에 의해 정의된다. **move** 명령어의 문법체계는 다음과 같다.

```
move <robot> [ with <position equation> ] [ until <condition> ];
<position equation> = [ <position equation> + ] <4×4 H.T.M>;

<Example>
move robot1 with position_eq until time < 30;
position_eq = robot1 + vision;                    (6)
```

여기에서 robot1과 vision은 로봇 1의 현재위치와 비전 센서에 의한 수정량인 4×4 동차 변환행렬(Homogeneous Transformation Matrix)이다. 식 (6)에서 40msec마다 vision의 값을 계산하기 위한 제어 알고리즘은 2절에서 설명된다. 또한 '+', '-'은 각각 행렬의 직접 연산과 역 행렬의 곱을 쉽게 표현하기 위하여 사용된다. 이때 사용되는 조건들은 그림 8과 같고, 여기서 δ_1, δ_2, δ_3는 사용자가 정의하는 임계치이다.

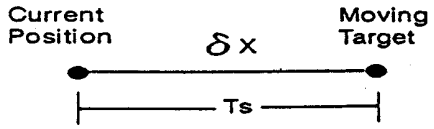

그림 7 이동물체 추적을 위한 구조
Fig. 7 Configuration for Tracking of Moving Target

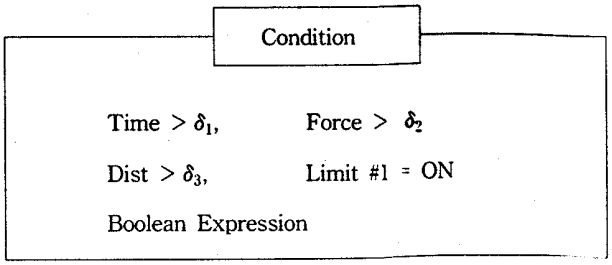

그림 8 조건 명령어 모음
Fig. 8 Conditional Statement Set

이와 같은 **move** 명령어는 Master & Slave Motion이나 Visual Servoing, Conveyor Tracking 작업등에 응용될 수 있다. 따라서 이렇게 제안·정의된 Sensor-Based 로보트 언어와 기존의 Robot-Level 언어를 함께 이용한다면 표현할 수 있는 작업의 종류와 기능이 더욱 많아져 보다 다양하고 효과적인 Task-Level Language를 구현할 수 있다.

3.2 시각 구동(Visual Servoing) 제어 알고리즘

최근 들어 로봇의 응용이 지능화 됨에 따라 로봇 스스로가 외부 환경을 감지하여 그에 맞게 적절히 작업을 수행해야할 필요성이 증대되고 있다. 로봇의 이러한 지능 능력은 여러 개의 환경센서(Environmental Sensor)를 사용함으로써 가능할 수 있다. 그 중에서 시각 센서에 의한 정보는 환경에 직접 접촉하지 않고도 전체를 인식할 수 있기 때문에 가장 강력한 센서 정보로 간주되고 있다.

본 논문에서는 위치서보(Position Servo)를 장착한 산업용 로보트의 시각 궤환(Visual Feedback) 정보를 이용하여 그림 12와 같이 SCARA 로보트가 Eye-in-Hand 형태로써 핸드에 부착된 CCD 카메라를 이용하여 3차원 공간상에서 움직이는 물체를 추적하게 된다. 실험을 간단히 하기 위하여 3차원 공간상에서의 X, Y, Z 위치(Position) 추적만을 고려하였고, Roll, Pitch, Yaw의 방향(Orientation)은 고려하지 않았다.

본 논문에서 수행된 시각구동 방법은 카메라 캘리브레이션(Camera Calibration)을 행하지 않고서, 그림 9와 같이 영상특징과 그 변화량만을 이용하여 시각구동을 수행하게 된다.

시각구동을 수행하기 전에 먼저 기준이 되는 영상특징을 추출한 후, 이 기준 영상특징(Desired Feature)과 현재의 영상특징(Actual Feature) 사이의 차이값, 그리고 현재의 영상특징을 이용하여 시각궤환 제어를 수행한다.

X,Y,Z 3차원 공간상에서의 위치를 나타내는 영상특징을 추출하기 위해서 다음과 같은 영상 특징을 선택하였다.

F_1 = 물체영상의 무게중심(center of gravity)의 X 좌표

F_2 = 물체영상의 무게중심(center of gravity)의 Y 좌표

F_3 = $\sqrt{\text{물체영상의 면적}}$

여기서는 영상특징의 중복(Feature Coupling) 없이 독립적인 위치를 표현할 수 있도록 3가지의 영상 특징을 설정하였다. 먼저 X-Y 2차원 공간을 표현하기 위해서 물체의 무게중심(Center of Gravity)의 X,Y 좌표를 각각 F_1, F_2로 설정하였으며, 거리정보를 표현하기 위해서 물체 영상의 면적을 이용하였다. 여기서 물체영상의 면적을 그대로 영상특징으로 설정한다면 거리정보의 변화율과 물체영상의 면적의 변화율 사이의 선형관계를 보장할 수 없게 된다. 따라서 물체영상의 면적에 Square Root를 취함으로써 선형관계에 근사한 영상특징 F_3을 얻을 수 있게 된다.

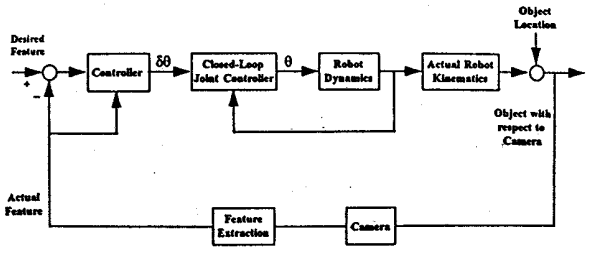

그림 9 비젼 센서를 이용한 시각 구동의 구조
Fig. 9 A Scheme of Visual Servoing using Vision Sensor

이때 기준 영상특징을 추출할 때는 Teach-by-Showing 방법을 이용한다. 즉, 시각구동의 대상 물체를 CCD 카메라와 대상 물체 사이의 원하는 위치 관계가 형성되는 장소에 놓고 CCD 카메라로 영상을 Capture 해서 기준 영상특징 F_{1d}, F_{2d}, F_{3d}를 추출한다.

따라서, 영상특징의 변화량을 로보트 좌표계상에서의 변화량으로 변환시켜 주는 Mapping Function을 $G(\cdot)$ 이라 하면 다음과 같은 관계가 형성된다. 즉, 로보트의 팔 끝에 부착되어 있는 CCD 카메라를 기준으로한 좌표계를 ^cX라 하면

$$\delta^cX = G(F_i, \delta F_i) \quad i=1,2,3 \tag{7}$$

여기서, $\delta F_i = F_{id} - F_i \quad i=1,2,3$

현재의 영상특징(Actual Feature)인 F_1, F_2, F_3를 추출한 후, 기준 영상특징(Desired Fea- ture)인 F_{1d}, F_{2d}, F_{3d}와의 차이를 이용해서 카메라 좌표 공간에서의 위치 변화량이 계산된다. 이를 구체적으로 보면 카메라의 시선을 물체의 중심에 일치시킨 후, 화면에 보이는 물체의 크기가 원하는 크기가 될 때까지 계속 시선 방향으로 다가가는 것이다. 이와 같이 시선 방향으로 물체를 추적하는 경우, 물체가 어디에 있던 간에 카메라로 물체를 볼 수만 있으면 항상 추적이 가능하다. 이때 로봇은 시선 방향으로 움직이는 기능만 있으면 되므로 기존의 일부 방법처럼 전체 작업 공간에 대한 정보가 필요 없게 된다.

본 논문에서는 움직이는 탁구공을 효과적으로 추적하기 위하여 카메라의 움직임과 영상 특징 변화량 사이의 비선형 관계인 $G(F, \delta F)$에 적합한 퍼지 제어기를 채용하였다. 이때 사용된 퍼지규칙은 (8)식과 같고, 여기서 Y_1은 물체영상 무게중심의 X 좌표와 화면 Center의 x_0좌표의 차이고, Y_2는 물체영상 무게중심의 Y좌표와 화면 Center의 y_0좌표의 차이다. 출력은 각 방향으로 Normalize된 속도이다. 이에 로봇의 최고 속도(1cm/80msec)에 해당하는 Scale Factor를 곱하면 실제 로봇에게 주어야 할 속도 명령이 된다. 이에 사용된 입·출력 퍼지 소속함수(Membership Function)는 앞절에서 언급한 영상 특징들을 이용하였다. 즉, 기준 영상특징(Desired Feature)과 현재의 영상특징(Actual Feature) 사이의 차이값을 정규화(Normalize)하여 입력에 대한 소속함수로 이용하였으며, 그 소속 함수는 NL, NM, NS, NT, ZE, PT, PS, PM, PL로 나누었다. 특히, 목표지점 부근의 정밀도를 높이기 위해서 그림 10과 같이 ZE 부근의 소속함수를 세분화하였다.

If Y_1 or Y_2 is near -200, then velocity is -1.0
If Y_1 or Y_2 is near -120, then velocity is -0.8
If Y_1 or Y_2 is near -60, then velocity is -0.4
If Y_1 or Y_2 is near -20, then velocity is -0.1 (8)
If Y_1 or Y_2 is near 0, then velocity is 0.0
If Y_1 or Y_2 is near 20, then velocity is 0.1
If Y_1 or Y_2 is near 60, then velocity is 0.4
If Y_1 or Y_2 is near 120, then velocity is 0.8
If Y_1 or Y_2 is near 200, then velocity is 1.0

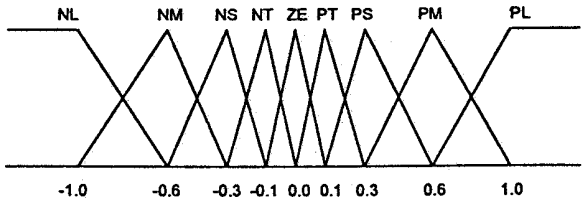

그림 10 입력변수 언어값의 소속함수
Fig. 10 Membership Function of Linguistic Values of Input Variables

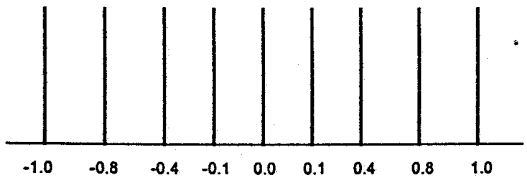

그림 11 출력변수 언어값의 소속함수
Fig. 11 Membership Function of Linguistic Values of Output Variables

한편, 출력에 대한 소속함수는 싱글톤(Singleton)으로 설정하였으며, 최종적으로 정규화된 소속함수는 다음과 같다.

출력에 대한 정확성을 보다 정확히 하기 위해서 신경망(Neural Network)을 이용하여 오차를 최소화하는 방향으로 신경망의 가중치를 조정함으로써 퍼지 소속함수의 출력에 대한 정밀도를 향상시킬 수 있었다.

3.3 이동 물체 추적 실험 : Visual Servoing

본 논문에서 제안한 시각구동 방법을 실제로 구현하기 위하여 실험에서 사용된 로봇은 그림 12와 같이 SPR-600 SCARA 로봇이며, 로봇의 팔끝에 CCD 카메라를 부착하였다. 전체 시스템의 구조는 제어를 담당하는 32-Bit CPU Board인 Force사의 CPU-30 2장, 비젼처리를 위한 흑백 프레임 그래버(Frame Grabber)인 DT-1451 등으로 이루어져 있다. 실험에 사용된 CCD 카메라는 TOSHIBA 사의 IK-M41MK CCD 신호 모듈과 IK-M30M 렌즈 (1:1.6, f=7.5) 이고, 시각구동의 대상 물체는 탁구공의 흰색 원형으로 하였으며, 비젼처리를 간단히 하기 위해서, 그림 12와 같이 대상물체를 검은색 배경에 놓고 실험을 진행하였다.

또한 S/W는 실시간으로 여러 개의 작업을 병렬로 처리할 수 있도록 하기 위하여 실시간 O.S.인 VxWorks를 사용하였다. 로봇제어를 위한 서보의 샘플링 타임은 5msec이며, 로봇의 효율적인 제어를 위해서 두 장의 CPU-30 Board를 사용하였다. 첫 번째 CPU Board에서는 로봇의 Motion만을 관장하며, 샘플링 타임을 40msec이다. 두 번째 CPU Board에서는 비젼처리를 제어하고, 제안한 알고리즘의 처리를 담당하며, 샘플링 타임은 80msec이다.

이때 비젼처리부의 샘플링 타임이 80msec이고, 로봇 제어부의 샘플링 타임이 40msec 이므로, 비젼처리를 한번 한 후 계산된 δX값을 선형 보간하여 로봇을 제어하게 하였다. 다음과 같이 앞에서 설명한 Moving Target Tracking을 위한 Language Set을 이용하여 구현하였다.

```
define inport2 vision with file = vision_file;
movj robot2 to loc3 with sp = 10;
move robot1 with position_eq until time < 45;
position_eq = robot1 + vision;
end
```

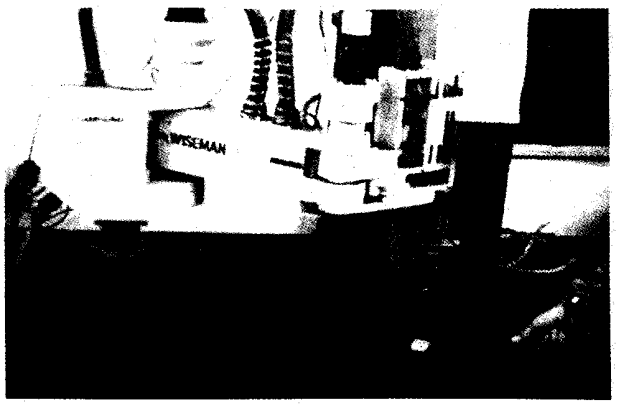

그림 12 Visual Servoing의 구현
Fig. 12 Implementation of Visual Servoing

4. Task-Level Robot Language의 개발

4.1 Task Level Robot Language의 개요

기존의 Robot-Level Language는 주어진 작업을 수행하는데 필요한 로보트 동작들로 프로그래밍을 하지만 Task-Level Language에서는 목표점에 이르기까지의 물체들의 상태들로 작업을 묘사한다. 예를 들면 Peg-in-hole 작업의 경우 Insertion을 수행하기 위해 필요한 일련의 로보트 동작들 대신에 INSERT PEG IN HOLE이라는 물체 중심의 동작 또는 조립 명령을 통하여 작업을 나타냄으로써 자동적으로 Task Planner에 의해 Task-Level 프로그램이 합성된 Robot-Level 프로그램으로 변환되어 테스크를 수행한다. 이때 합성된 Robot-Level 프로그램은 Sensor-Based Motion, 즉 센서 데이터를 이용한 동작계획 수정이 가능한 로보트 동작들로 구성되게 된다.

Task-Level 프로그래밍을 하기 위해서는 그림 13과 같이 Modeling, Task Specification, Robot Program Synthesis의 세 과정을 거치게 된다.

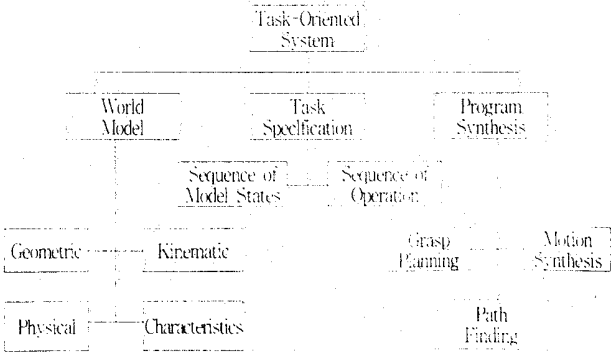

그림 13 Task-Level 프로그래밍 시스템의 구조
Fig. 13 Structure of Task-Level Programming System

(1) Modeling

Task의 World Model은 다음 정보를 가지고 있어야 한다.

- Task 환경내의 모든 Object와 로보트에 대한 기구학 표현
- Object들의 질량, 관성 등을 나타내는 물리적 표현
- Robot Kinematic 표현
- Joint Limit, 가속도 범위, 센서 해상도 등의 로보트 특성

2) Task Specification

Task는 Task Planner에 의해 Task를 완료하기까지의 연속된 World Model들로 표현된다. 이상적인 Task Planner의 경우에는 사용자가 Task의 초기상태와 완료 상태만을 지정해 주면 Model State를 자동으로 만들어 낸다.

3) Robot Program Synthesis

프로그램 Synthesis에서는 Grasping Planning, 동작 Synthesis, 그리고 Path-finding이 필요하다. 그리고 Synthesis 결과는 로보트 동작 명령어, Grasp 명령어, 센서처리 명령어, 에러 확인 등이 복합된 프로그램이 된다. 주변의 Object들과 충돌이 일어나지 않고 Grasp 위치 또한 물체가 안정되게 선정되도록 동작계획을 생성해 낸다. 그리고 동작 Synthesis에서는 현재 위치에서 출발하여 Task Step에 맞게 목표위치 까지 이동한 후 물체에 접촉하는 순간 Compliant Motion을 하여 최종 위치에 도달할 수 있도록 로보트 동작 명령어를 생성한다.

4.2 Task-Level Language 인터프리터의 설계

본 논문에서는 조립작업의 자동화를 목표로 사용자에게 보다 친숙한 언어인 Task-Level Language Set들로 조립작업을 기술하면 Task-Level Language 인터프리터에 의해 기본동작 제어언어와 Sensor-Based Robot Control 명령어로 표현되는 Robot-Level Language로 변환되는 인터프리터를 설계하였다. 본 연구에서 구성된 전체 제어 시스템의 소프트웨어 구조는 계층적 제어구조를 가지며 상위 레벨과 하위 레벨로 나누어진다. 상위 레벨에서는 Workstation을 중심으로 하여 Task-Level 프로그래밍의 개발을 위한 Task-Level 인터프리터가 있으며, 하위 레벨에서는 이미 개발된 Robot-Level 프로그래밍을 지원하기 위해 독립적인 제어시스템이 구축되어 있다. 따라서 상위 레벨에서 Task-Level 프로그래밍을 하여 컴파일 결과로 Robot-Level 프로그램을 생성하고 하위 레벨에서는 생성된 Robot-Level 프로그램을 Load하여 Robot-Level 인터프리터에 의해 동작 프로세서를 생성한 후 Joint 경로계획을 실시하고 서보 컴퓨터를 이용하여 계산된 경로 데이타값으로 직접 로보트 운동의 제어를 담당하게 된다. Task-Level Interpreter에서는 Task-Level 명령어를 하위 레벨의 로보트 제어기에서 제공하는 Robot-Level 명령어 및 Sensor-Based Robot 명령어로 표현된 일련의 작업 명령으로 번역하게 된다. 또한 사용자가 필요에 따라서 Robot-Level 명령어 및 Sensor-Based 로봇 명령어와 제공되는 Task-Level 명령어를 이용하여 새로운 Task-Level 명령어를 정의하는 기능을 담당하며 이에 대한 제어 시스템의 소프트웨어 구조는 그림 14와 같다. 이러한 Task-Level Language로 표현된 명령을 여러 단계의 Robot-Level Language로 변환시키기 위해서는 각 동작에서 사용되는 물체들의 형상정보 및 기하학적인 관계 정보들이 표현된 데이타 베이스를 사전에 구축하고, 계속되는 연구에서 이들 데이타 베이스를 이용하여 위치정보 및 조립 작업에 필요한 조립방향등을 계산하여 Task-Level Language에서 Robot-Level Language로 자동 변환하는 동작계획생성에 대해 연구를 하고자 한다.

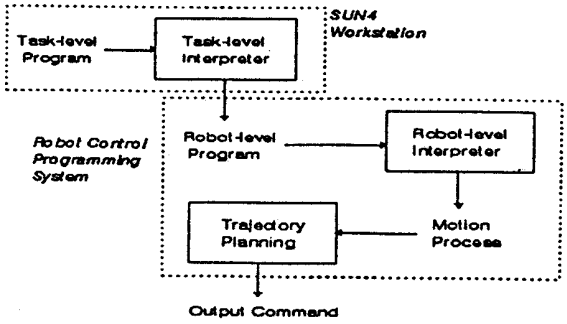

그림 14 제어 시스템의 전체 소프트웨어 구조

Fig. 14 Overall Software Structure of Control System

4.3 Object 모델링에 의한 Task Description

조립 계획을 생성하기 위한 첫 단계는 조립품을 분석하여 조립품 모델을 만드는 것이다. 조립품 모델은 조립 작업을 수행하기 위해 필요한 데이터를 포함하고 있어야 하는데, 부품의 특징 추출을 위한 형상정보와 조립순서 및 조립동작, 조립부품들 간의 Topology를 나타내는 기하학적 관계정보를 들 수 있다. 이와 같은 부품의 형상정보와 조립품의 관계정보 데이타 베이스를 이용하여 조립순서 및 필요한 동작을 생성하고자 한다. 이와 반면 [10]은 조립품의 모델링에 의한 조립순서 자동생성의 관점에서 연구를 수행하였다.

본 연구에서는 조립을 위한 로보트의 동작 생성에 중점을 둔 조립품 모델링에 의해서 데이타 베이스를 구축하고, 사용자가 보다 친숙한 언어를 이용하여 조립동작을 기술할 수 있는 Task-Level Robot Language 개발에 관한 연구를 수행하였다.

부품의 데이타 베이스는 본 연구에서 사용하는 콘트롤러의 메뉴 중 "등록"이라는 과정을 통하여 생성되며 그 구조는 이름, 형상정보, 관계정보로 구성되는데 이름은 부품의 고유명칭을 나타내고 형상정보는 물체인식을 위한 정보로써 비젼 카메라를 이용한 특징량 추출로 얻어지며, 물체인식을 위한 생김새 및 특징량, 물체에 부착된 좌표계, 로보트가 물체를 잡기 위한 위치 및 자세 정보가 포함된다. 관계정보는 조립작업에 이용시 다른 부품과의 접촉 관계를 저장하는 데이타 구조로써 물체에 접촉 및 조립되는 각각의 부품들에 대한 정보가 각각 따로 저장된다. 이러한 정보는 CAD 데이타 또는 비젼 센서로 부터 얻어내거나 사용자에 위해 입력되며 접촉면의 이름, 물체에 부착된 좌표계를 기준으로 표현된 접촉방향 및 접촉점 그리고 조립작업을 수행할 때 필요한 힘 정보 등이 있다. 접촉면의 이름은 한 개의 부품에 여러 개의 부품 접촉이 있을 경우 조립동작 명령에서 각 부품의 접촉면을 지정하기 위해 사용되는 이름이며, 접촉 방향은 부품이 조립될 방향을 나타내고, 접촉점은 두 부품간의 위치 정렬을 위한 기준점으로 사용된다.

또한 힘 정보는 힘센서를 이용한 조립 동작시 물체의 좌표계를 기준으로 각 방향으로 가해야 할 힘 또는 토크의 양을 정의하여 Peg-In-Hole과 같이 센서가 반드시 필요로 하는 동작도 수행 가능토록 하였으며, 다른 동작에서도 센서 데이타를 이용함으로써 외부 환경의 변화에 적응할 수 있도록 하였다. 이렇

게 정의한 데이타 베이스의 입력과정은 그림 15와 같고, 물체의 정보를 저장하는 데이타 베이스의 구조는 다음과 같다.

```
structure data_base {
   char name[ ] ;                    /* object name                */
   structure feature_data {          /* vision을 통한 형상정보     */
        char   centroid[3];          /* object의 무게중심          */
        char   feature_filed[ ];     /* feature data 저장화일      */
        int    object_origin;        /* object 좌표계의 원점       */
        float  grasp_point[3];       /* object를 잡기 위한 위치    */
        float  grasp_orientation[3]; /* object를 잡기 위한 자세    */
   } feature;
   structure relation_data {         /* 부품간의 관계정보          */
        char   contact_name[ ];      /* 접촉면의 이름              */
        float  contact_direction[3]; /* 부품의 접촉방향            */
        char   force_file[ ];        /* 힘정보 파일                */
   } relation[ ];
} object[ ];
```

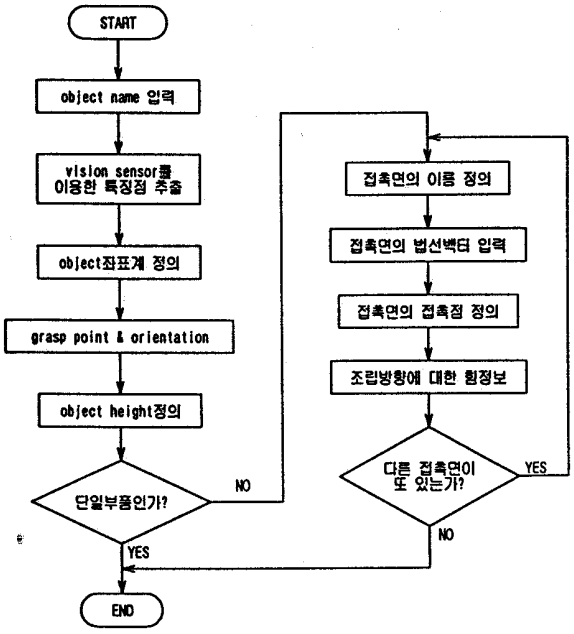

그림 15 데이타 베이스의 입력과정
Fig. 15 Input Procedure of Data Base

4.4 조립 작업을 위한 Task-Level Robot Language

본 논문에서는 인간의 언어와 유사한 물체중심의 작업 명령으로 로보트가 작업대에서 조립작업을 수행하는 것을 목표로 Task-Level Robot Language를 정의 및 구현하였으며 조립동작 명령과 기타 프로그램 제어명령 및 선언문과 연산자로 구분된다. 특히 조립 명령어에서는 작업수행에 필요한 여러가지 종류의 센서들을 선택적으로 사용할 수 있으며 문법 표현에서는 "[ ]"으로 나타난다. Task-Level Language Interpreter에서는 이러한 명령어를 해석하여 현재의 조립 동작에 사용할 물체의 상태 및 Task 수행에 사용할 센서의 종류에 따라 작업을 수행하기 위한 Robot-Level 프로그램을 생성하게 된다. 이렇게 생성된 Robot- Level 프로그램에는 기본동작 제어명령 및 Sensor-Based 동작명령이 포함되며 온-라인 상태에서 판단해야 할 상황에 대한 조건문 등이 포함된다.

4.4.1 조립작업을 위한 Task-Level 명령어

간단한 예를 들면, 전자회로 기판 조립작업의 경우 대부분의 작업이 부품을 집어 기판에 삽입하는 동작의 반복으로 구성된다고 가정할때, 기존의 Robot-Level Language에서는 하나의 부품삽입을 위해 6~10단계의 기본동작 제어명령(move, grasp, release,...)으로 구현되어야만 하였다. 하지만 본 논문에서 제안한 Task-Level Language의 조립명령에서는 로보트 중심의 작업명령이 아니라 조립작업에 사용될 부품 위주의 서술형식으로 조립작업을 기술, 프로그래밍 할 수 있게 하였다. 본 논문에서 정의한 기본적인 조립작업에 필요한 명령어의 문법 및 기능은 다음과 같다.

- pick <object> [using <vision> sensor];

데이타 베이스에서 물체에 해당하는 정보를 참조하여 현재 물체의 위치 및 Grasp 자세등을 계산하여 Grasp 동작을 수행한다.

- place <object> to <location>;

현재 잡고 있는 Object를 지정된 Location에 내려놓는 명령으로, 이때 Location이 2차원 좌표값일 경우 데이타 베이스로부터 물체의 높이를 참조한다.

- attach <contact_name#1> of <object#1> to <contact_name #2> of <object#2> [using <vision and/or force> sensor];

평면으로 구성된 부품들 간의 면의 접촉을 나타내는 조립동작으로 각 물체의 정의된 Contact_name에 해당하는 데이타 베이스를 참조하여 조립위치 및 조립방향과 조립 동작시 필요한 힘을 계산하여 동작을 수행한다.

- insert <contact_name> of <peg> into <hole_name> of <object> [using <vision and/or force> sensor];
- screw <contact_name> of <bolt> on <hole_name> of <object> [using <force> sensor];

홈이나 구멍에 물체를 삽입하는 동작 및 나사를 체결하는 Contact_name에 해당하는 데이타 베이스를 참조하여 동작을 수행한다.

이러한 문법에 따라 기술된 Task-Level 명령은 Task-Level 인터프리터에 의해 현재 사용 가능한 센서의 종류 및 물체의 상태에 따라 조립작업에 필요한 일련의 Robot-Level 명령어로 변환되며 한 예로 insert 명령을 Robot-Level 명령으로 변환하는 알고리즘은 그림 16과 같다. 단, 여기서 결합위치와 접근위치는 각각 조립부품을 대상부품에 결합했을 때의 위치와 동작중 대상 부품과의 충돌을 막기 위해 결합동작을 수행하기 전 대상부품의 접촉방향쪽으로 조립부품을 이동해야 할 위치를 나타낸다.

4.4.2 기타 프로그램 제어 명령어 및 연산자

본 논문에서는 조립동작 Task-Level 명령어 외에 변수선언 명령, 프로그램 제어명령(if, for, while statement), 논리연산자, 관계연산자, 산술연산자 등의 명령을 제공한다. 변수 선언문에서는 int, float, char 변수 뿐만 아니라 location 변수도 사용

가능하며 배열의 정의가 가능하다.

조건문 : if (<var> == <var/num>) then . . . . endif
반복문 : for <var>=<num> to <num> step=<num>
. . . next <var>
: while (<var> == <var/num>) . . . .
end_while
변수선언문 : int, float, char, location
논리연산자 : AND, OR, NOT, XOR
관계연산자 : < , <= , >= , > , == , !=
산술연산자 : + , - , * , /

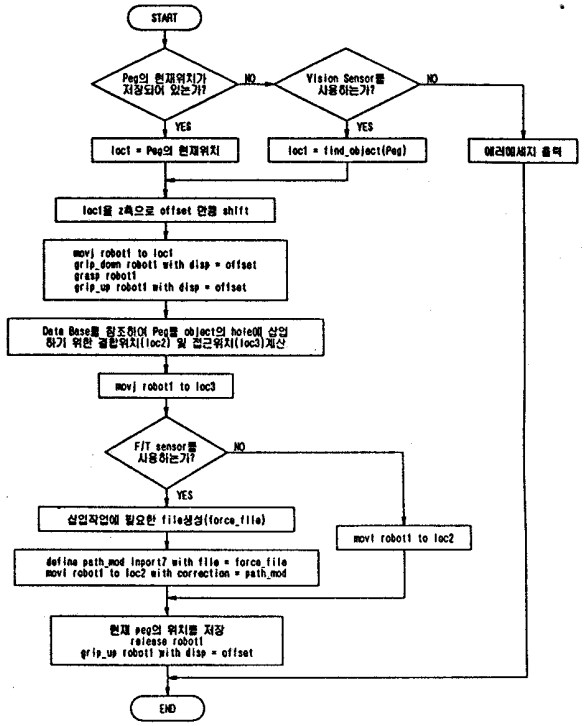

그림 16 Insert 동작을 위한 Task-Level Language 인터프리터 알고리즘
Fig. 16 Algorithm of Task-Level Language Interpreter for Insert Operation

4.5 세개의 부품으로 구성된 제품의 조립작업 예

본 논문에서 제안한 Task-Level Robot Language를 이용한 조립작업 기술의 효율성을 보이기 위하여 그림 17과 같은 부품의 조립작업을 가정하고 프로그램하였다.

조립작업은 먼저 part_a를 Base 부품으로 임의의 위치에 옮겨두고 part_a에 part_b를 접촉점 및 접촉방향에 맞게 조립하고, 볼트를 이용하여 part_b를 part_a에 부착하는 과정으로 이루어진다. 이러한 조립작업을 위해서는 part_a, part_b, bolt에 대한 모델링이 필요하다. 모델링 과정은 그림 18~20에서 보듯이 각 부품에서 접촉이 일어날 곳에 대한 접촉점 및 접촉방향을 정의함으로써 이루어지며 이러한 모델링에 의해 생성된 각 부품에 대한 데이타 파일의 구조중 관계정보에 대한 예는 다음과 같다.

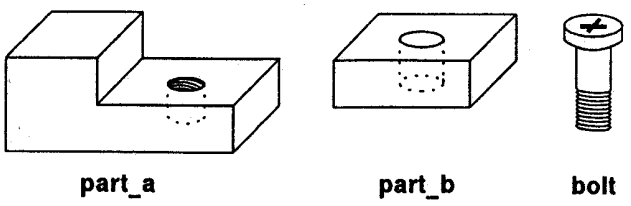

그림 17 3개의 부품으로 이루어진 조립제품의 예
Fig. 17 An Example of Assembly Task Consisting of 3 Parts

부품 "part_a"의 relation_data

```
contact_name      = "right";
contact_direction = {0,1,1};
contact_point     = {5,7,10};
contact_force     = {0,10,10};

contact_name      = "up";
contact_direction = {0,0,1};
contact_point     = {2.5,11,0};
contact_force     = {2,2,15};
```

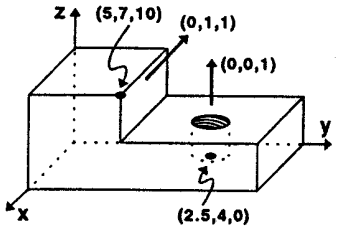

그림 18 Part_a의 모델링
Fig. 18 Modeling of Part_a

부품 "part_b"의 relation_data

```
contact_name      = "left";
contact_direction = {0,-1,-1};
contact_point     = {5,0,6};
contact_force     = {0,10,10};

contact_name      = "up";
contact_direction = {0,0,1};
contact_point     = {2.5,4,0};
contact_force     = {2,2,15};
```

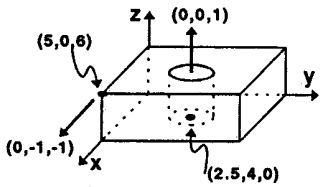

그림 19 Part_b의 모델링
Fig. 19 Modeling of Part_b

부품 "bolt"의 relation_data

```
contact_name      = "bottom";
contact_direction = {0,0,-1};
contact_point     = {0,0,0};
contact_force     = {2,2,15};
```

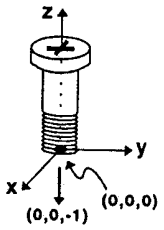

그림 20 Part_c의 모델링
Fig. 20 Modeling of Part_c

이러한 모델링이 끝난 후 앞에서 정의한 Task-Level Language들을 이용하여 이들 부품의 조립을 기술하게 되면 Task-Level 인터프리터에 의해 실제 로보트를 동작시킬 Robot-Level 프로그램이 자동으로 생성된다. 이러한 조립작업을 위한 Task-Level 프로그램과 생성된 Robot-Level 프로그램의 예는 다음과 같다. 단, 여기서의 모든 부품들은 공급 장치로부터 공급되거나 그 위치를 알고 있다고 가정하며 조립작업

에 사용되는 Tool에 대한 명령은 배제하였다.

<Task Level Robot Program>

```
#include example.dat
object     part_a, part_b, part_c;

pick part_a;
place part_a to loc10;
attach left of part_b to right of part_a using force sensor;
insert bottom of bolt into hole of part_b;
screw bottom of bolt on nut of part_a using force sensor;
```

<Interpreter에 의하여 생성된 Robot Level Program>

```
loc1 = cur_location("part_a");
location_shift loc1 with dz = 5.5;
movj robot1 to loc1;
grip_down robot1 with disp = 5.5;
grasp robot1;
grip_up robot1 with disp =5.5;

location_shift loc10 with dz = 5.5;
movj robot1 to loc10;
grip_down robot1 with disp = 5.5;
cur_location("part_a") = cur_robot_location(robot1);
release robot1;
grip_up robot1 with disp = 5.5;

loc1 = cur_location("part_b");
location_shift loc1 with dz = 5.5;
get_assemble_location("left","part_b","right","part_a", loc2, loc3);
movj robot1 to loc1;
grip_down robot1 with disp = 5.5;
grasp robot1;
grip_up robot1 with disp = 5.5;
movj robot1 to loc3;
define inport7 force_sensor with file = file1;
movl robot1 to loc2 with correction = force_sensor sp = 30;
cur_location("part_b") = cur_robot_location(robot1);
release robot1;
grip_up robot1 with disp = 5.5;

loc1 = cur_location("bolt");
location_shift loc1 with dz = 5.5;
get_assemble_location("bottom","bolt","hole","part_b", loc2, loc3);
movj robot1 to loc1;
grip_down robot1 with disp = 5.5;
grasp robot1;
grip_up robot1 with disp = 5.5;
movj robot1 to loc3;
define inport7 force_sensor with file = file2;
insert_peg robot1 toward loc2 with correction = force_sensor
sp = 20;

get_assemble_location("bottom","bolt","nut","part_a", loc2, loc3);
define inport7 force_sensor with file = file3;
drive_in robot1 toward loc2 with correction = force_sensor
sp = 30;
cur_location("bolt") = cur_robot_location(robot1);
release robot1;
grip_up robot1 with disp = 5.5;
end
```

5. 결 론

본 논문에서는, 전자제품조립 Task의 자동화를 목표로 하여 기존에 개발한 로보트 중심의 제어언어를 토대로 Sensor-Based 로봇 동작제어의 기능을 강화하고, 이들을 이용하여 각 작업에 필요한 Task Primitive를 구성함으로써 로보트의 작업을 편리하게 기술할 수 있는 좀 더 발전된 형태의 로보트 제어언어인 Task-Level Language를 제시하였다. 즉, 로보트의 지능화, 고기능화를 연구개발 목표로 사용자가 제품의 조립상태를 보고 두 부품간의 접촉방향, 조립방향, 조립순서 등의 구체적인 조립상태를 데이타 베이스 형식으로 입력하고, 조립순서에 따라 조립 테스크를 자동적으로 로봇의 기본 동작으로 변환하며, 동작계획을 수행하는데 필요한 부품의 형상, 위치 및 자세인식을 자동 수행하는 등의 내용을 포괄하는 로보트에 종속적인 요소를 배제시킨 대상물체 중심의 Task-Oriented Robot Language를 개발하였다.

또한, 앞으로의 연구과제는 대상물체의 CAD 데이타로 부터 조립부품들 간의 접촉방향 및 조립에 필요한 관계정보들을 자동생성하는 연구와 산업현장에서 적용할 수 있도록 Task-Level Robot Language 보완에 대한 연구가 수행되어야 할 것이라고 사료된다.

참 고 문 헌

[1] T.Henderson, E.Weitz, C.Hansen, and A.Mitiche, "Multi-sensor Knowledge Systems : Interpreting 3D Structure", *Int. J. Robotics Research*, Vol. 7, No. 6, pp. 114-137, 1988.

[2] R.C.Luo and M.G.Kay, "Multisensor Integration Fusion in Intelligent Systems", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 19, No. 5, pp. 901-931, 1989.

[3] L.I.Lieberman and M.A.Wesley, "AUTOPASS : An Automatic Programming System for Computer Controlled Mechanical Assembly", *IBM J. Research Development*, Vol.21, No. 4, pp. 321-333, 1977.

[4] J.W. Roach and M.N.Boaz, "Coordinating the Motion of Robot Arms in a Common Workspace", *IEEE J. of Robotics and Automation*, Vol. 3, No. 5, pp.437-444, 1987.

[5] M.A.Arbib, R.O.Eason and R.C.Gonzalez, "Autonomous Robotics Inspection and Manipulation Using Multisensor Feedback", *IEEE Trans. on Computer*, Vol.24, No.4, pp. 17-31, 1992.

[6] E.Rutten, *et. al.*, "A Task-Level Robot Programming Language and its Reactive Execution", *IEEE Int. Conf. on Robotics and Automation*, Vol.3, pp.2751-2756, 1992.

[7] Y.Kuniyoshi, H.Inoue, and M.Inaba, "Design and Implementation of a System that Generates Assembly Programs

from Visual Recognition of Human Action Sequence", *IEEE Int. Conf. on Intelligent Robotic Systems*, pp. 567-574, 1990.

[8] R.A.Volz, "Report of the Robot Programming Language Working Group : NATO Workshop on Robot Programming Languages", *IEEE J. Robotics and Automation*, Vol. 4, No. 1, pp. 86-90. 1988.

[9] D.Peng and K.G.Shin, "Modeling of Concurrent Task Execution in a Distributed Systems for Real-Time Control", *IEEE Trans. on Computer*, Vol. C-36, No. 4, pp. 500-516, 1987.

[10] 한민홍, 정무영, 조형석, 권대갑, "지능형 조립시스템의 구축을 위한 기초 연구", 한국 과학 재단 최종 보고서, 1993.

[11] P.K.Allen, B.Yoshimi, and A.Timcenko, "Real-time visual servoing", In Proc. of IEEE Int. Conf. Robotics and Automation, pp.851-856, 1991.

[12] J.T.Feddema and O.R.Mitchell, "Vision-guided servoing with feature-based trajectory generation", IEEE Trans. Robotics and Automation, Vol.5, No.5, pp.691-700, Oct. 1989.

[13] I.H.Suh, *et. al.*, "A Control System for Multiple-Robot Manipulators ; Design and Implementation", *Proc. of ISRAM'94*, Vol.5, pp.279-285, 1994.

[14] I.H.Suh, *et. al.*, "Fuzzy Adaptive Force Control of Industrial Robot Manipulators with with Position Servos", *Mechatronics Pergamon Press*, Vol.5, No.8, pp.899-918, 1995.

[15] I.H.Suh, *et. al.*, "Fuzzy Membership Function Based Neural Networks with Applications to the Visual Servoing of Robot Manipulators", *IEEE Tans. on Fuzzy Systems*, Vol.2, No.3, pp. 203-220, 1994.

[16] A.V.Aho *et. al.*, "Compilers Principles, Techniques, and Tools", Addison Wesley, 1988.

[17] *Operation Manual of Universal Force-Moment Sensor System*, Nitta Ind., Ltd, 1988.

[18] *User manual for DT1451*, Data Translation, Inc., 1988.

[19] *CPU-30 User's Manual*, FORCE COMPUTERS, Inc., 1990.

[20] *VxWorks Programmer's Guide*, Wind River Systems, Inc., 1990.

저 자 소 개

서 일 홍 (徐 一 弘)

전기학회논문지 제45권 제1호 참조

여 희 주 (呂 熙 珠)

1965년 5월 2일생. 1988년 한양대학교 전자공학과 졸업. 1990년 동 대학원 전자공학과 졸업(석사). 현재 한양대 대학원 전자공학과 박사과정