
차 례

학습모듈의 개요	1
학습 1. 작업 요구 사항 파악하기	
1-1. 작업 요구 사항 파악	3
• 교수·학습 방법	18
• 평가	19
학습 2. 작업 수행 지능 설계하기	
2-1. 작업 수행 지능 설계	21
• 교수·학습 방법	61
• 평가	62
참고 자료	65

작업지능 소프트웨어 개발 학습모듈의 개요

학습모듈의 목표

로봇 작업 수행을 위하여 작업 요구 사항을 파악하고 설계, 구현할 수 있다.

선수학습

C 프로그래밍, 로봇 소프트웨어 플랫폼(ROS) 프로그래밍, 로봇 공학, 모션 컨트롤러

학습모듈의 내용 체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 작업 요구 사항 파악하기	1-1. 작업 요구 사항 파악	1903080318_16v2.1	작업 요구 사항 파악하기
2. 작업 수행 지능 설계하기	2-1. 작업 수행 지능 설계	1903080318_16v2.2	작업 수행 지능 설계하기

핵심 용어

지능로봇, 소프트웨어, 알고리즘, 인공지능, 작업지능, 이동지능, 인지지능, HRI지능

1-1. 작업 요구 사항 파악

학습 목표

- 로봇 작업을 수행하기 위한 작업 요구 사항을 분석할 수 있다.
- 작업을 성공적으로 수행하기 위하여 로봇 성능 스펙을 도출할 수 있다.
- 도출된 결과에 따라 로봇 작업지능 소프트웨어의 목표 사양을 선정할 수 있다.

필요 지식 /

① 작업 요구 사항 분석(이만형, 1999)

로봇(robot)의 어원은 체코 극작가 카렐 차펙(KAREL CAPEK)이 1920년에 발표한 최초의 로봇 드라마, ROSSUM' S UNIVIERSAL ROBOTS에서 등장한 robota이며, 이 단어는 체코어로 '강제노동'을 뜻한다. 이는 사람이 로봇을 개발하는 첫 번째 목적이 사람을 대신해서 위험하거나 유해한 환경에서 규격화된 작업을 장시간으로 정확하게 하는 것임을 의미한다.

산업용 로봇이 가장 많이 활용되고 있는 자동차 제조 공정에는 스폿(spot) 용접용 로봇, 아크(arc) 용접용 로봇, 도장(painting)용 로봇, 조립용 로봇, 핸들링용 로봇, 이·적재용 로봇, 부품 가공용 로봇, 검사용 로봇, 최종 조립용 로봇 등이 있다.

1세대 로봇이면서 계속해서 성능 향상 및 가격 하락이 이뤄지고 있는 산업용 로봇(Industrial Robot)은 제조공정에서 다음의 네 가지 분야 작업에 응용되고 있다.

- 자재취급: 부품배치, 적재 및 하역, 기계에 장착 및 탈착 등
- 공정처리: 용접, 도장, 절삭 및 디버링(deburring) 작업, 도포 작업 등
- 조립: 자재취급과 공구의 조종을 포함한 일괄 조립(batch assembly) 또는 적은 단위의 조립(low volume assembly)
- 검사: 측각 게이지, 선형변위변환기(linear variable differential transformer), 로봇 비전 등을 이용해서 부품 또는 제품의 불량 여부 판정

제조공정의 네 가지 작업 분야에 대해 보다 상세히 설명하고 작업별 로봇 관련 요구 사항을 분석하면 다음과 같다.

1. 자재취급

로봇이 제조공정에서 자재나 부품을 취급하기 위해서는 다음의 기능들을 갖춰야 한다. 로봇은 자재를 안전하게 들어 올리고 정확한 위치에 놓을 수 있어야 하고, 자재를 취급하기 위한 충분한 힘·토크를 낼 수 있어야 하며, 작업의 사이클에 비하여 충분한 속도를 낼 수 있어야 한다.

(1) 부품배치

로봇이 어떤 위치에서 1개의 부품을 들어 올려서 목적 위치에 정확히 놓는 작업을 의미하며 영어로 픽-플레이스(pick-and-place)라고 한다.

이 작업에는 들어 올릴 부품의 형상과 위치를 인식하기 위한 영상인식, 역기구학 계산, 정밀제어, 동작 정숙성이 필요하다.

(2) 적재 및 하역

로봇이 1개의 부품 또는 제품을 운반대 혹은 다른 부품 위에 적재(palletizing)하거나 또는 하나씩 차례로 하역(depalletizing)하는 작업을 의미한다.

이 작업은 부품배치와 마찬가지로 적재 또는 하역할 부품이나 제품의 형상과 위치를 인식하기 위한 영상인식, 역기구학 계산이 필요하며, 무거운 부품이나 제품일 경우에는 큰 작업토크도 필요하다.

(3) 기계에 장착 및 탈착

로봇이 부품들을 생산 기계에 붙이거나 공급(loading)하는 작업을 장착이라고 하고 생산 기계로부터 작업 완성품을 떼어내거나 회수(unloading)하는 작업을 탈착이라고 한다. 장착 및 탈착의 예는 다이캐스팅, 플라스틱 사출, 단조, 프레스 작업 등이 있다.

이 작업에는 영상인식, 역기구학 계산과 더불어, 생산 기계와 로봇 간의 작업 사이클 시간의 차이를 고려할 수 있는 능력이 필요하다. 이를 위해서는 공정 전체를 담당하는 제어기가 있어야 하며, 제어기와 로봇 간에 유·무선 통신이 필요하다.

또 생산 기계의 작업이 진행되는 동안 로봇이 공급 또는 회수를 할 수 있도록 로봇의 작업 속도가 빨라야 한다.

2. 공정처리

공정처리 작업은 로봇이 주어진 부품이나 자재에 용접, 도장, 절삭 가공 및 디버링(deburring, 버 제거), 도포 작업 등을 수행하는 것을 의미한다. 이 작업의 특징은 공정에 적합한 공구(tool)를 로봇의 말단부(end effector)에 설치해야 한다는 것이다.

(1) 용접

용접에는 두 장의 얇은 금속판을 접촉점에서 녹여서 접합하는 점용접(spot welding)과

경로를 따라서 연속적으로 녹여서 접합하는 아크용접(arc welding)이 있다.

점용접 작업에는 공압 또는 모터로 제어되는 무거운 용접총(spot gun)이 로봇에 장착 되므로 큰 적재하중 능력을 갖춰야 하며, 용접 지점간(point-to-point) 제어 기능이 중요하다. 아크용접 작업에는 연속 경로제어 능력이 중요하다.

(2) 도장

현재 많이 사용되는 도장(painting) 방식으로는 분무 도장(air spray painting), 스프레이 도장(airless spray painting), 정전기 도장(electrostatic painting), 정전기 분말도장(electrostatic powder painting) 등이 있다.

이 중에서 분무 도장은 도료를 압축 공기와 함께 물품에 뿌리며 도장하는 방법으로, 작업 능률이 좋고 넓은 부분에 균일하게 도장할 수 있다. 분무 도장은 인체 건강에 유 해한 대표적인 작업이므로, 로봇을 적용하면 작업환경과 생산성을 크게 향상시킬 수 있다.

로봇은 분무 도장에서 요구되는 부드러운 작업 순서를 달성할 수 있도록 연속 경로 제어 능력이 필요하다.

(3) 절삭가공 및 디버링

절삭가공이란 만들고자 하는 제품보다 경도가 큰 공구를 사용하여 공작물로부터 칩(chip)을 깎아서 원하는 형상을 만들어내는 작업을 의미하며, CNC(computer numerical control)를 이용한 선반(공작물을 회전시키면서 공구로 가공) 작업, 밀링(공구가 회전하 면서 공작물 가공) 작업, 드릴링(공작물에 회전하는 드릴을 눌러대어 구멍 뚫기) 작업 등이 있다.

로봇이 절삭가공을 수행하기 위해서는 위치와 자세를 동시에 제어하고 고강성을 유지 하면서 고정밀도를 충족시켜야한다.

디버링이란 절단이나 절삭 등의 모든 가공 공정 후 필연적으로 발생하는 날카로운 부 분이나 작은 조각인 버(burr)를 제거하는 작업을 의미한다.

디버링 작업을 위해서는 로봇과 작업면 간의 상호작용력(interaction force) 제어가 중 요하다.

(4) 도포

도포란 점착제나 도료를 대상 제품의 표면에 균일하게 바르는 작업을 의미하며, LNG(liquid natural gas) 탱크의 단일 판넬에 에폭시 점착제를 도포하거나, 자동차 부품 에 점착제를 도포하거나, 신발 제조 공정에서 신발 갑피의 모양을 고려하여 점착제를 최적의 경로로 바르는 작업 등을 예로 들 수 있다.

신발 제조 공정의 경우 신발 갑피의 모양을 고려하여 점착제 도포용 경로점을 생성하 는 방법과 경로점들을 연결한 유클리디안 거리가 최소가 되도록 경로를 최적화하는 방법 등이 필요하다.

3. 조립

조립은 부품들을 체결 및 해체하여 제품을 만드는 작업으로서 전통적으로 노동 집약적인 산업이다. 조립은 반복적이고 지루한 작업이지만 일반적으로 그 내용이 조립제품에 따라 방대하고 복잡하다. 또 인쇄회로기판 조립과 같이 정밀한 작업 수행 시 반복 정밀도가 높아야 한다.

로봇으로 조립작업을 하기 위한 고려사항으로는 로봇의 말단장치가 부품들을 쉽게 들어 올리고 조립할 수 있도록 하는 것이다.

최근에는 인간과 로봇이 함께 조립을 할 수 있는 협동로봇(collaborative robot) 기술이 개발되고 있으며 복잡한 조립 작업을 수행하기 위한 양팔 로봇 기술이 대두되고 있다.

4. 검사

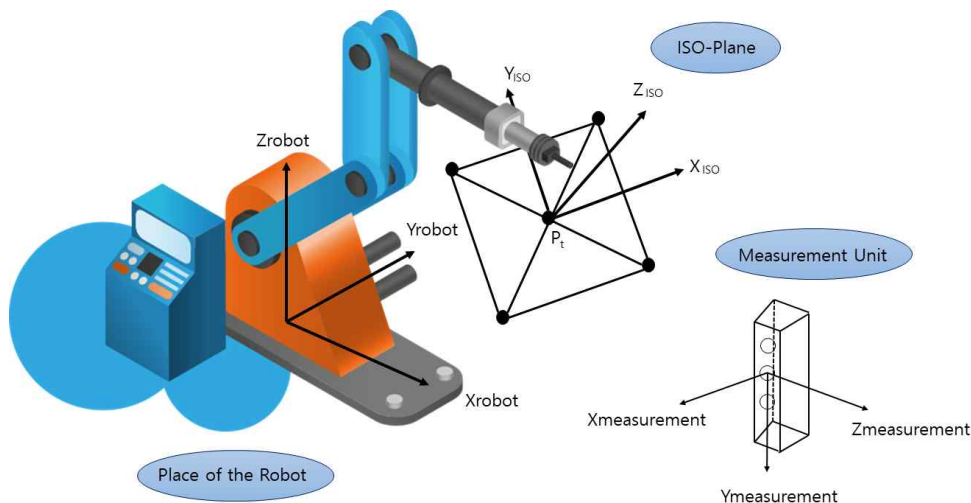
검사 작업에서는 부품이 움직이거나 검사용 장치가 움직이며 부품 또는 제품의 불량여부를 판단하며, 고도의 정밀성을 요구한다.

검사 로봇 중 비파괴검사 로봇은 초음파를 이용하여 강관, 강재 수문, 선박, 압력 용기 등의 용접 부위의 합격 여부, 부식 결함 등을 원격·자동으로 검사할 수 있어, 기존에 사람이 접근하기 힘든 곳에서 정교하게 안전진단을 할 수 있다.

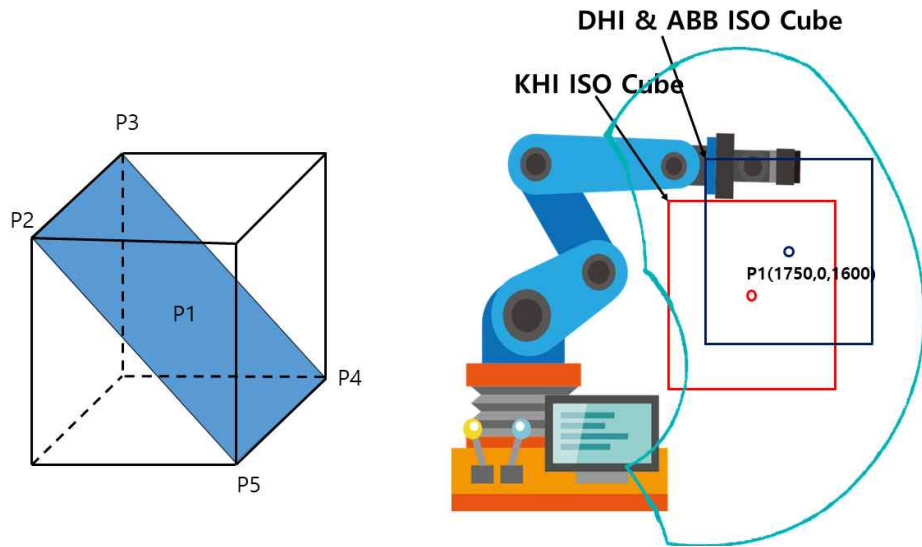
② 로봇 성능 스펙 도출

로봇의 주요 설계 사양은 가반 하중(payload), 최대 속도(max speed), 위치 반복 정밀도(pose repeatability), 작업 영역(work space) 등이다.

로봇 성능 시험은 ISO 9283에 의거하여 [그림 1-1]과 같은 측정 시스템으로 다양한 항목을 시험한다. 공간상의 위치 측정을 위해서는 광학식 거리 측정 시스템을 사용하거나, 레이저 센서를 이용한 거리 측정 시스템을 사용한다.



[그림 1-12] 로봇 성능 측정 시스템 장치



[그림 1-13] ISO 테스트 큐브

ISO 9283의 로봇 성능 측정 항목은 다음과 같다.

1. 자세 특성 시험

(1) 자세 정밀도/반복성(pose accuracy/repeatability)

로봇의 위치 결정 정밀도 및 반복 정밀도 평가. [그림 1-2]의 ISO 규정점 P1에서 출발하여 P5→P4→P3→P2→P1 순서로 연속으로 움직이도록 프로그래밍하고 각 점에서 위치 측정

(2) 거리 정밀도/반복성(distance accuracy/repeatability)

Off-Line, MDI(manual data input) 대응능력 평가. ISO 규정점 P4에서 출발하여 P2와 P4를 왕복하도록 프로그래밍하고 각 점에서 위치 측정

(3) 다방향 자세 편차(multi-directional pose deviation)

접근 방향에 따른 정밀도 평가. ISO 규정점 P1 → P2 → P4 순서로 움직이도록 프로그래밍하고 각 점에서 200mm씩 떨어진 x, y, z축 좌표에서 위치 측정

(4) 안정화 시간(stabilization time)

목표점에서 위치결정 시간 평가. 진폭 한계를 $\pm 0.1\text{mm}$ 로 규정하고 규정치 이내로 들 때까지의 시간 측정

(5) 오버슈트(overshoot)

목표점에서 벗어난 편차 평가. ISO 규정점 P1에서 출발하여 P5 → P4 → P3 → P2 → P1 순서로 연속으로 움직이도록 프로그래밍하고 각 점에서 편차 측정

자세 편차(pose drift): 시간에 따른 정밀도 천이 평가. 주전원을 인가한 후 8시간 워업(warmup) 상태를 지속하며 10분 간격으로 ISO P1점에서 위치 측정

2. 경로 특성 시험

(1) 직선경로 정밀도/반복성(linear path accuracy/repeatability)

직선을 이루는 두 점을 교시(teaching)하고 직선동작과 반복동작이 이뤄지도록 프로그램을 만든 후 교시된 경로와 실제 경로를 비교하여 측정

(2) 곡선경로 정밀도/반복성(circular path accuracy/repeatability)

원호를 이루는 두 점을 교시(teaching)하고 원호 보간(interpolation) 시험을 위한 프로그램을 만든 후 교시된 경로와 실제 경로를 비교하여 측정

(3) 경로 속도 정밀도/반복성(path velocity accuracy/repeatability)

작업영역 내에서 등속·변속 구간이 있는 속도 프로파일을 포함하여 P4에서 출발하여 P2와 P4를 충분히 왕복하도록 프로그래밍하고 각 점에서 위치 측정

(4) 코너링 특성(cornering characteristics)

P5와 P2의 중간을 의미하는 P52에서 출발하여 P52 → P2 → P23, P23 → P3 → P34, P34 → P4 → P45, P45 → P5 → P52를 연속하여 움직이도록 프로그램을 만든 후 각 점(P1, P2, P3, P4)에서 3회씩 측정

3. 최소 위치정립 시간 시험

로봇이 위치정립(positioning) 시 동특성을 평가하기 위한 목적이며, 측정점 P1에서 X, Y, Z 방향으로 동일한 거리만큼 떨어진 10개의 점(이동점) D1-D10을 정해 D1(P1) → D2 → D3 → D4 → D5 → D6 → D7 → D8 → D9 → D10 → D1(P1) 순서로 연속 동작하도록 프로그램을 만든 후 왕복 운동에 걸리는 시간 측정

③ 작업지능 소프트웨어 목표 사양 선정

작업을 수행할 로봇의 동작을 시뮬레이션 할 수 있는 소프트웨어에는 여러 가지가 있다. 대표적인 것으로 ROS(robot operating system)의 MoveIt!, PETER CORKE가 MATLAB으로 구현한 오픈소스인 Robotics Toolbox(<http://www.petercorke.com/RVC/top/toolboxes>), SimLab의 Robotics Lab 등이 있다.

소프트웨어의 목표 사양을 정할 때 작업을 수행할 로봇에 대해 다음의 사양을 고려해야 한다.

- 로봇의 종류와 구조, 형상수치 선정
- 로봇의 인식 기능 선정
- 로봇의 자율화 지수(자율성) 선정
- 로봇의 성능 지수 선정

1. 로봇의 종류와 구조 선정

목표 작업을 수행하기 위한 로봇이 정해져 있을 경우 작업에 필요한 가반 하중, 최대 속도, 위치 반복 정밀도, 작업 영역을 만족하는지 확인한다.

로봇이 정해지지 않은 경우 <표 1-1>을 참조하여 적절한 로봇의 구조와 이동 수단을 가진 로봇의 종류를 선택하고, 관절과 링크의 기구학적 구조를 설계한다.

<표 1-1> 로봇 타입 분류

구분	로봇 종류
매니퓰레이터 구조에 따른 분류	직각좌표형 로봇(cartesian type robot), 원통좌표형 로봇(cylindrical type robot), 수평관절형 로봇(SCARA), 수직관절형 로봇(vertical articulated type robot), 극좌표형 로봇(polar coordinate type robot), 병렬 로봇(parallel robot) 등
이동 수단에 따른 분류	고정형 로봇, 바퀴형 로봇(mobile robot), 족형 로봇(legged robot), 공중 로봇, 수중 로봇 등

매니퓰레이터 구조에 따른 로봇들을 작업 좌표계 관점에서 비교하면 다음 페이지의 [그림 1-3]과 같다.

제조공정의 네 가지 작업 분야에 공통적으로 사용될 수 있는 로봇은 매니퓰레이터 구조상 수직관절형 로봇이며 이동 수단에 따른 분류에 의하면 고정형 로봇이다. 각 작업 분야별로 적절한 구조를 갖춘 로봇 종류를 예시하면 다음과 같다.

(1) 자재취급

- 부품배치: 직각좌표형 로봇, 원통좌표형 로봇, 수평관절형 로봇, 병렬 로봇
- 적재 및 하역: 원통좌표형 로봇, 수평관절형 로봇, 바퀴형 로봇
- 기계 장착 및 탈착: 원통좌표형 로봇

(2) 공정처리

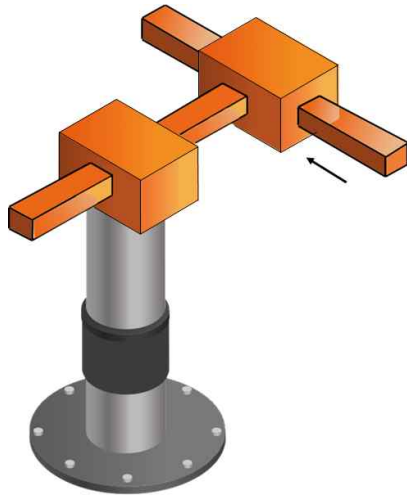
- 용접·도장: 수직관절형 로봇, 극좌표형 로봇
- 절삭가공 및 디버링: 직각좌표형 로봇
- 도포: 직각좌표형 로봇

(3) 조립

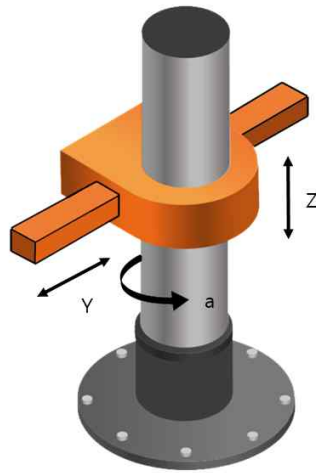
- 직각좌표형 로봇, 원통좌표형 로봇, 수평관절형 로봇

(4) 검사

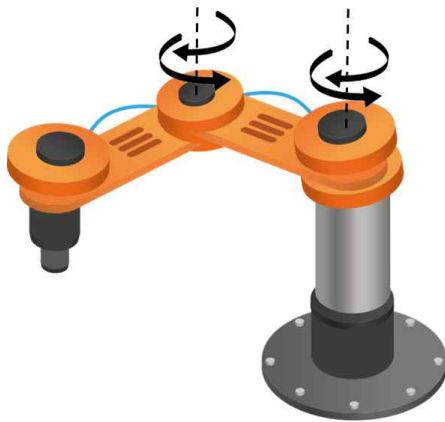
- 직각좌표형 로봇, 수평관절형 로봇, 바퀴형 로봇



(a) 직각좌표형 로봇



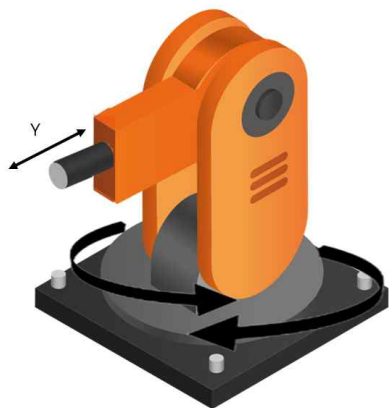
(b) 원통좌표형 로봇



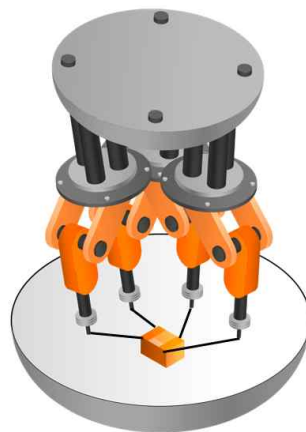
(c) 수평관절형 로봇



(d) 수직관절형 로봇



(e) 극좌표형 로봇



(d) 병렬 로봇

[그림 1-3] 매니퓰레이터 로봇 구조 비교

2. 작업별 로봇 인식 기능 도출

앞에서 나열한 제조공정의 네 가지 작업 분야에서 관련된 로봇 인식 기능(시각, 청각, 촉각, 토크 측정, 거리 측정) 중 카메라를 이용한 시각 기능은 기본적으로 필요하며 추가적으로 필요한 인식 기능은 다음과 같다.

(1) 자재취급

적재 및 하역, 기계에 장착 및 탈착: 거리 측정

(2) 공정처리

용접 · 절삭가공 및 디버링 · 도포: 촉각, 토크 측정

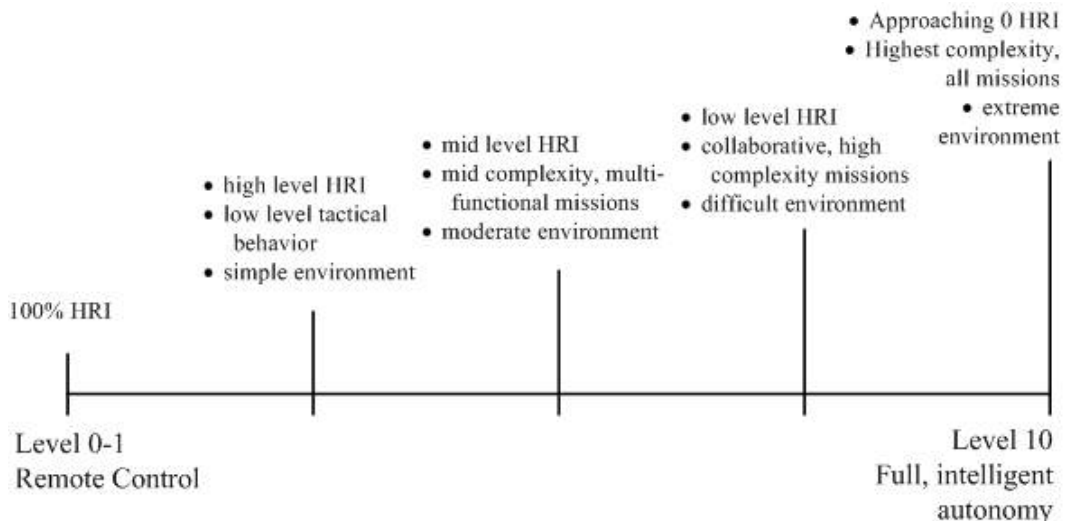
(3) 조립

촉각, 토크 측정, 거리 측정

3. 자율화 지수 선정

앞에서 로봇의 종류와 구조, 인식 기능이 선정되었으면 로봇의 자율화(autonomy) 지수를 선정한다.

미국 국립표준기술연구소(NIST)에서는 ALFUS(autonomy levels for unmanned systems)를 표준으로 제정하여 임무의 복잡도, 환경의 복잡도, 인간-로봇 상호작용(human robot interaction, HRI) 정도에 따라 [그림 1-4]처럼 시스템의 자율화 지수를 0(완전 수동식)에서 10(완전 자율화) 사이로 정의하였다.



출처: Huang, Pavak, and Albus et al.(2005), Autonomy Levels for Unmanned System (ALFUS) Framework: An Update, Proceedings of SPIE Defense and Security Symposium

[그림 1-4] 미국 NIST의 무인 시스템 자율화 지수인 ALFUS

제조공정의 네 가지 작업 분야를 임무의 복잡도, 환경의 복잡도, 인간-로봇 상호작용 정도 관점에서 상증하를 분류하면 다음과 같다.

(1) 임무의 복잡도

- 상: 조립
- 중: 자재취급
- 하: 공정처리, 검사

(2) 환경의 복잡도

- 상: 자재취급
- 중: 조립, 공정처리
- 하: 검사

(3) 인간-로봇 상호작용 정도

- 상: 조립
- 중: 자재취급, 검사
- 하: 공정처리

4. 작업별 로봇 성능 스펙 도출

제조공정의 네 가지 작업 분야에서 관련된 로봇 성능 스펙을 도출하면 다음과 같다.

(1) 자재취급

- 부품배치: 자세/거리의 정밀도/반복성, 다방향 자세 편차, 안정화 시간, 오버슈트
- 적재 및 하역: 자세/거리/경로의 정밀도/반복성, 다방향 자세 편차, 안정화 시간
- 기계에 장착 및 탈착: 자세/속도의 정밀도/반복성, 안정화 시간

(2) 공정처리

- 용접: 점용접의 경우 자세/거리의 정밀도/반복성, 아크용접의 경우 경로 정밀도/반복성, 오버슈트
- 도장: 경로 정밀도/반복성
- 절삭가공 및 디버링: 자세/거리/속도의 정밀도/반복성, 힘 제어
- 도포: 경로 정밀도/반복성

(3) 조립

- 자세 및 거리 정밀도/반복성, 다방향 자세 편차, 안정화 시간, 오버슈트

(4) 검사

- 자세 및 거리 정밀도/반복성, 다방향 자세 편차, 안정화 시간

5. 로봇 작업지능 소프트웨어 목표 사양 선정

앞의 과정에 따라 작업지능 소프트웨어의 목표 사양을 선정한다.

수행 내용 / 작업 요구 사항 파악하기

재료 · 자료

- 작업 요구서
- ISO 9283 해설자료
- 로봇 개요 설명 책자

기기(장비 · 공구)

- 컴퓨터, 프린터
- 문서 작성용 소프트웨어
- 로봇 동작 시뮬레이션 소프트웨어

안전 · 유의 사항

- 로봇의 종류와 제품이 매우 다양하므로 작업 요구 사항을 잘 숙지하여 적절한 로봇 종류와 구조를 선정하는 것이 중요하다.
- 공개 소프트웨어를 사용할 경우 라이선스를 읽고 무료로 사용해도 문제가 없는지를 확인해야 한다.
- 로봇의 종류와 구조, 인식 기능, 자율화 지수(자율성), 성능 스펙을 모두 고려해서 소프트웨어의 목표 사양을 선정해야 한다.

수행 순서

① 로봇 작업을 수행하기 위한 작업 요구 사항을 분석한다.

1. 로봇으로 수행할 작업의 공정순서표를 입수하거나 작업자의 협조를 통해 <표 1-3>과 같은 형태로 작성한다.

<표 1-3> 공정순서표 예: 마운트 부품 용접 자동화

순서	작업공정	용접공정	비고
1	Jig에 소재 투입		작업자
2	제품 Clamping		Jig
3	로봇 이동 및 용접	공정 1~3	로봇

순서	작업공정	용접공정	비고
4	Positioner 회전		Positioner
5	로봇 이동 및 용접	공정 4	로봇
6	Positioner 회전		Positioner
7	로봇 이동 및 용접	공정 5~6	로봇
8	제품 Unclamping		Jig
9	용접 완료제품 취출		작업자
10	로봇 스테이션 Change		로봇
11	반복		

2. 로봇이 수행해야 할 작업이 제조공정의 어느 분야에 해당하는지를 확인하고 세부 작업 내용을 파악하기 위해 <표 1-4>를 작성한다.

<표 1-4> 로봇 작업 분야 및 요구 인식기능

제조공정 및 인식기능	분야	작업내용	비고
자재취급	부품배치		
	적재 및 하역		
	기계에 장착 및 탈착		
공정처리	용접		
	도장		
	절삭가공 및 디버링		
	도포		
조립			
검사			
기타			
인식기능	시각		
	촉각		
	토크 측정		
	거리 측정		
	기타		

② 작업을 성공적으로 수행하기 위하여 로봇 성능 스펙을 도출한다.

1. 작업에 필요한 가반 하중, 최대속도, 위치 반복 정밀도, 작업 영역 등을 조사하고, 로봇 성능시험 기준인 ISO 9283을 참고하여 다음과 같은 표를 작성한다.

<표 1-5> 로봇 성능 요구 사항표

항목	단위	목표값	비고
가반 하중			
최대속도			
위치 반복 정밀도			
자세 정밀도			
자세 반복성			
안정화 시간			
오버슈트			
직선 경로 정밀도			
직선 경로 반복성			
곡선 경로 정밀도			
곡선 경로 반복성			
기타			

2. <표 1-6>을 참고하여 작업을 수행할 로봇의 종류를 선정하고, 관절과 링크의 기구학적 구조를 설계한다.

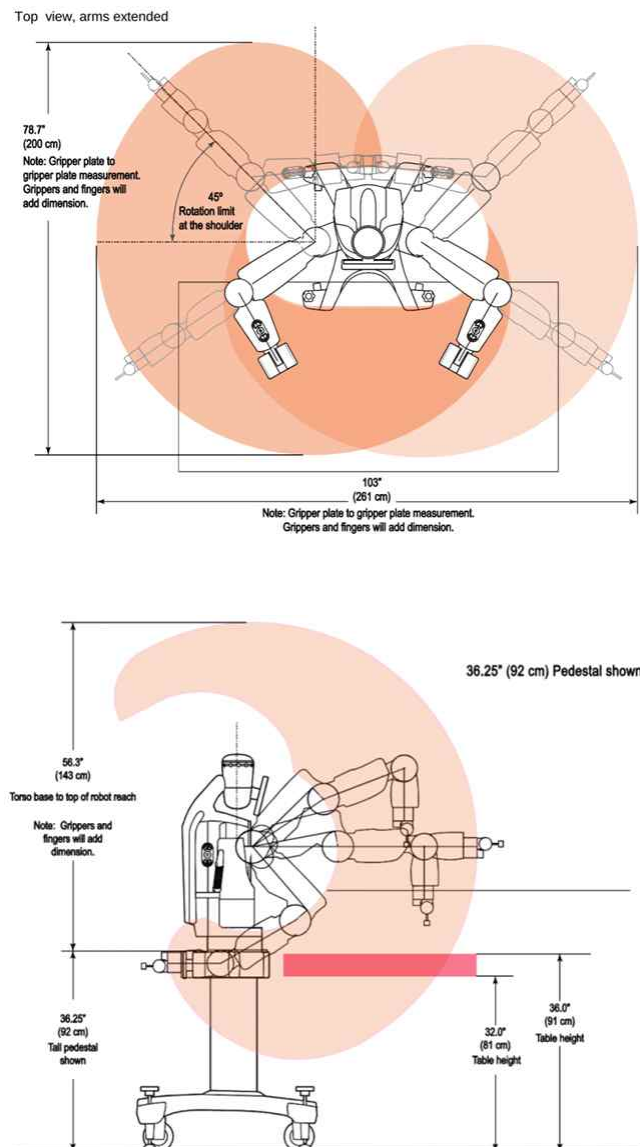
<표 1-6> 제조 분야별 적절한 로봇종류

제조공정	분야	로봇 종류
자재취급	부품배치	직각좌표형, 원통좌표형, 수평관절형, 병렬로봇
	적재 및 하역	원통좌표형, 수평관절형, 바퀴형 로봇
	기계에 장착 및 탈착	원통좌표형 로봇
공정처리	용접·도장	수직관절형, 극좌표형 로봇
	절삭가공 및 디버링	직각좌표형 로봇
	도포	직각좌표형 로봇
조립	직각좌표형, 원통좌표형, 수평관절형	
검사	직각좌표형, 수평관절형	

3차원 공간에서 임의의 위치에 있는 물체를 임의의 방향으로 잡기 위해서는 최소한 6자유도의 매니퓰레이터 로봇이 필요하다.

제한된 작업 공간에서 정해진 위치에 있는 물체를 인식하고 집어서 목표 위치에 놓는 작업을 하는 데에는 3자유도 직각좌표형 로봇이 필요하다.

3. 작업에 필요한 로봇의 인식 기능(시각, 촉각, 토크 측정, 거리 측정 등)을 알아보고 해당 기능의 센서를 로봇 설계에 추가한다.
4. 설계된 로봇이 작업에 필요한 요구 사항을 만족시키는지 확인한다. 작업을 수행할 로봇으로서 연구용 협동로봇인 RETHINK ROBOTICS사의 백스터(BAXTER)를 선택했다면 웹사이트나 매뉴얼에서 작업 반경이 [그림 1-5]와 같음을 확인할 수 있다.



[그림 1-5] 백스터의 작업 반경

③ 계획된 작업을 수행하기 위한 로봇 작업지능 소프트웨어의 목표 사양을 선정한다.

1. 로봇 작업지능 소프트웨어를 선정한다.

ROS는 오픈소스 로봇용 소프트웨어 플랫폼으로서 2007년 스탠포드 대학 AI Lab에서부터 개발이 시작되고 WILLOW GARAGE라는 회사가 이를 이어 받아 2010년에 ROS 1.0을 개발한 후 매년 새로운 버전이 알파벳 순서의 이름으로 발표되고 있다. 현재 ROS는 비영리 단체인 Open Robotics에 의해 운영되고 있으며, ROS는 BSD 라이선스를 기반으로 하고 있어 누구든지 무료로 다운로드하여 수정, 재사용, 재배포가 가능하다.

ROS를 사용하기 위해 컴퓨터 운영체제로 ROS가 공식적으로 지원하는 Ubuntu를 설치한다. 시중의 ROS 교재에 있는 예제 코드를 사용하는 경우 ROS Kinetic Kame를 설치한다. 이외에도 로봇 작업지능 소프트웨어로서 OpROS, Matlab, Robotics Lab 등이 있다.

2. ALFUS를 기준으로 로봇의 자율화 지수(자율성)를 선정한다. <표 1-7>에서 항목별 상중하를 참고하여 지수를 0~10 사이에 선정한다. 임무의 복잡도와 환경의 복잡도는 높을수록 지수가 높아지고, 인간-로봇 상호작용 정도는 낮을수록 지수가 높아지는 것에 유의한다.

<표 1-7> 로봇의 자율화 지수 선정

정도	임무의 복잡도	환경의 복잡도	인간-로봇 상호작용 정도
상	높음	매우 복잡함	없음
중	보통	복잡함	중간 정도
하	낮음	단순함	100%

3. 지금까지의 내용을 토대로 로봇 작업지능 소프트웨어의 목표 사양을 선정한다.

교수 방법

- 제조공정의 네 가지 분야의 세부 내용들을 설명한다.
- 로봇의 종류와 동작 원리, 장단점에 대해 그림과 함께 설명한다.
- 로봇의 성능 스펙에 대해 ISO 9283에 의거하여 그림과 함께 설명한다.
- 로봇에 사용되는 센서의 종류와 관련된 인식 능력을 사진과 함께 설명한다.

학습 방법

- 제조공정의 네 가지 분야 중 관심 있는 작업 분야를 상세히 조사한다.
- 로봇의 종류와 동작 원리, 장단점을 이해하고 관심 있는 작업 분야에 맞는 로봇을 조사한다.
- 관심 있는 작업 분야와 관련된 ISO 9283 성능스펙을 선택한다.
- 관심 있는 작업 분야와 관련된 센서를 선택하고 해당 센서의 인식 능력을 조사한다.

학습1 평 가

평가 준거

- 평가자는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습 내용	평가 항목	성취수준		
		상	중	하
작업 요구 사항 파악	- 로봇 작업을 수행하기 위한 작업 요구 사항을 분석할 수 있다.			
	- 작업을 성공적으로 수행하기 위하여 로봇 성능 스펙을 도출할 수 있다.			
	- 도출된 결과에 따라 로봇 작업지능 소프트웨어의 목표 사양을 선정할 수 있다.			

평가 방법

- 포트폴리오

학습 내용	평가 항목	성취수준		
		상	중	하
작업 요구 사항 파악	- 로봇과 관련된 제조공정의 네 가지 분야 자료 조사			
	- 로봇의 종류와 장단점에 대한 보고서			
	- ISO 9283 요약보고서			
	- 로봇에 사용되는 센서의 종류와 관련된 인식 능력 조사 보고서			
	- 사용자 요구 분석 결과에 따른 로봇 작업지능 소프트웨어의 목표 사양 선정 과제			

• 서술형 시험

학습 내용	평가 항목	성취수준		
		상	중	하
작업 요구 사항 파악	- 로봇과 관련된 제조공정의 네 가지 분야에 대해 설명			
	- 로봇의 종류와 동작 원리, 장단점에 대해 설명			
	- 로봇의 성능 스펙에 대해 ISO 9283에 의거하여 설명			
	- 로봇에 사용되는 센서의 종류와 관련된 인식 능력을 설명			

• 사례 연구

학습 내용	평가 항목	성취수준		
		상	중	하
작업 요구 사항 파악	- 로봇과 관련된 제조공정의 각 분야 별 사례 연구			
	- 관심 있는 로봇의 종류가 적용된 사례 연구			
	- 관심 있는 로봇에 사용되는 센서의 종류와 인식 능력 사례 연구			
	- 사용자 요구 분석 결과에 따른 로봇 작업지능 소프트웨어의 목표 사양을 선정 사례 연구			

피드백

1. 포트폴리오

- 포트폴리오를 평가하고 부족한 부분이나 문제점을 기재한 후 수정 및 보완 사항을 확인한다.

2. 서술형 시험

- 각 질문에 대한 답변의 구체성과 적절성을 확인하고 부족한 점을 지도한다.

3. 사례 연구

- 사례 연구로서 조사한 내용을 발표함으로써 내용을 서로 공유하고 부족한 부분은 보완할 수 있도록 지도한다.

2-1. 작업 수행 지능 설계

학습 목표

- 소프트웨어의 목표 사양을 만족하는 작업지능을 분석할 수 있다.
- 로봇과 주변장치 인터페이스를 분석할 수 있다.
- 분석된 결과에 따라 알고리즘을 도출할 수 있다.
- 로봇 작업 지능 알고리즘을 기반으로 한 프로그램을 작성할 수 있다.

필요 지식 /

① 인공지능

인공지능(artificial intelligence, AI)은 인간의 지능(학습 능력과 추론 능력, 지각 능력, 자연 언어의 이해력 등)을 컴퓨터 프로그램으로 구현하는 기술로서, 분야별로 보면 지식 표현, 문제 해결, 지식시스템, 자연어 처리, 학습, 인지모델, 지능로봇 등으로 구분할 수 있다.

인공지능 시스템은 세 가지 기능을 할 수 있어야 하는데 첫째는 지식(컴퓨터 입장에서는 데이터)을 기호체계를 이용해서 저장할 수 있어야 하며, 둘째는 이러한 지식을 추론(reasoning) 과정을 통해 문제 해결에 적용할 수 있어야 하고, 셋째는 실험을 통해 얻은 새로운 지식을 수용하여 학습할 수 있어야 한다.

로봇 지능 소프트웨어의 핵심인 인공지능은 궁극적으로 사람과 유사한 수준으로 판단하고 행동하는 능력을 갖추어야 하는데, 이에 해당하는 인공지능 시스템은 최근 연구가 활발하게 진행되고 있는 인지 에이전트(cognitive agent)와 심층신경망(deep neural network), 강화학습(reinforcement learning), 최적화 알고리즘(optimization method)이다.

1. 인지 에이전트

인지 아키텍처(cognitive architecture)는 메모리, 프로그램 실행과 제어 컴포넌트, 데이터 표현, 입출력 장치들로 구성된 컴퓨터 아키텍처 구조 위에 환경과 태스크에 대한 지식의 표현과 습득, 활용을 지원하며, 목표 기반 동작을 통해 주어진 목표를 달성할 수 있게 한다. 이 과정을 공식으로 표현하면 “아키텍처+지식=행위”가 된다.

인지 아키텍처에서 코딩되는 지식은 일반지식(general knowledge)과 태스크지식(task

knowledge)을 포함하는데, 일반지식은 언어처리, 계획, 회고적 추론(retrospective reasoning) 등 일반적인 인지 능력을 뒷받침하는 지식을, 태스크지식은 문제와 기술 영역(domain)에 국한된 지식을 각각 의미한다.

인지 아키텍처는 지능시스템에 요구되는 주요 기능인 지각(perception), 추론(reasoning), 계획(planning), 언어처리(language processing), 학습(learning)을 잘 조정하는 역할을 수행해야 한다.

인지 아키텍처는 동적인 환경에서 태스크들을 잘 수행하도록 지식을 학습, 코딩, 활용하는 것을 지원하는 구조를 갖추고 있어야 하며, 의사 결정과 환경과의 상호작용 또한 지원해야 한다.

인지적 에이전트 아키텍처로는 Soar, ACT-R, LIDA, Clarion, EPIC, Icarus 등이 발표되었다. 이 중에서 Soar는 인간처럼 임의의 환경에서도 학습과 경험으로 얻게 된 지식을 저장하고, 불러오며, 처리함으로써 다양한 방식으로 작업을 수행하고 문제를 풀 수 있는 인간 수준 AGI(artificial general intelligence)를 목표로 1981년부터 ALLEN NEWELL, HERBERT SIMON, JOHN LAIRD에 의해 공동으로 개발된 인지적 에이전트이다(LAIRD, 2006).

Soar는 MICHIGAN대학교 JOHN LAIRD 교수의 주도로 현재 9.6버전이 매뉴얼과 다양한 예제와 함께 전용 웹사이트에 오픈소스로 공개되어 있다.

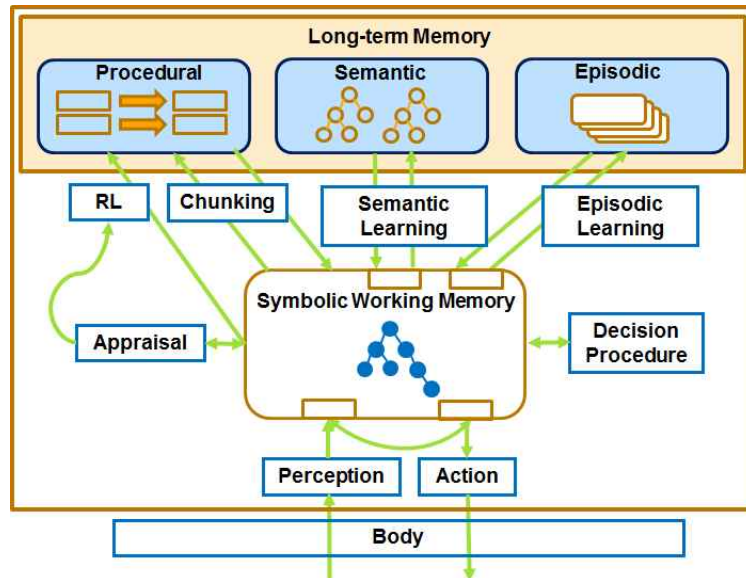
(1) Soar(state, operator, and result)

Soar는 지식집약적 추론(knowledge-intensive reasoning)과 반응적 실행(reactive execution), 계층적 추론(hierarchical reasoning)과 계획(planning), 학습(learning) 등이 통합되어 있는 범용 인지 아키텍처다.

Soar는 주어진 문제와 하위문제(subproblem)에 대해 광범위한 종류의 타입과 레벨을 사용할 수 있어서, 미리 작성되었거나 경험을 통해 새로 학습된 사용가능 지식을 동적으로 융합하여 잘 동작할 수 있다.

Soar의 구성 요소는 다음과 같다.

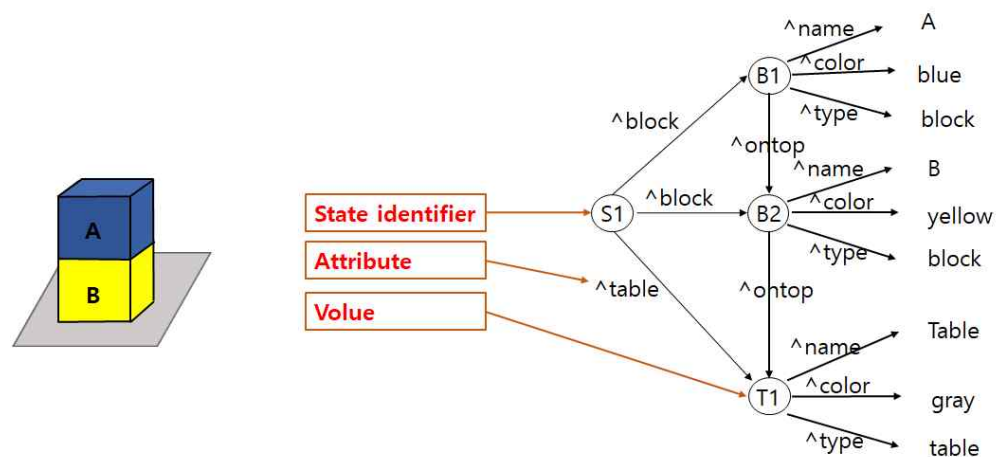
- 복수개의 장기기억(long-term memory) 시스템: 절차적(procedural) · 삽화적(episodic) · 의미적(semantic) 메모리
- 복수개의 단기 지식(short-term knowledge): 작업 메모리 요소(working memory element), 상징적(symbolic) · 도식적(diagrammatic) · 형상기반(imagery-based) 표현
- 복수개의 학습 메커니즘: 강화학습(reinforcement learning), 청킹(chunking), 삽화적 · 의미적 학습



출처: Laird, J. E. (2006). The Soar Cognitive Architecture. MIT Press. p.17.
[그림 2-1] Soar의 전체 구조

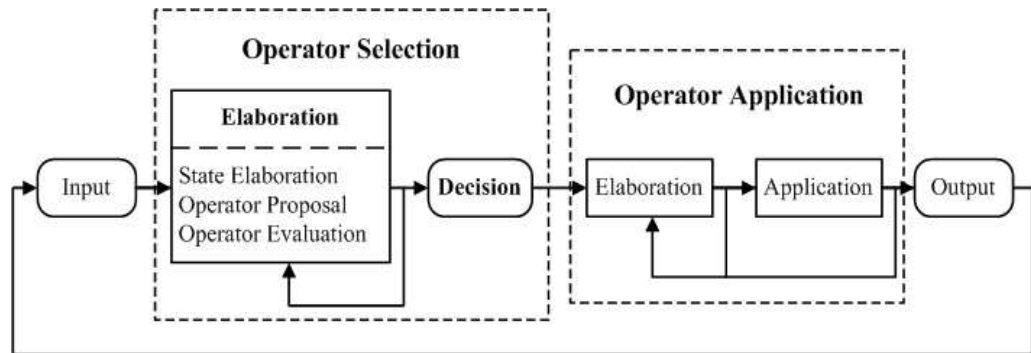
Soar의 작업 메모리 요소는 Soar 에이전트의 외부환경(world)과 내부 추론(reasoning)에 관한 모든 동적인 정보, 즉 센서 데이터, 중간 계산값, 현재 연산자, 그리고 목표(goal) 등을 담고 있다. 작업 메모리 요소는 노드와 링크를 이용하여 식별자(identifier)-속성(attribute)-값(value)의 삼중구조로 구성되어 있다. 식별자는 자신으로부터 링크가 나오는 노드이며 속성을 가질 수 있다. 속성은 작업메모리의 특성(색, 크기, 무게)을 나타내며, 값은 속성이 가리키는 노드를 의미한다.

작업 메모리 요소의 객체(object)는 동일한 식별자를 공유하는 작업 메모리 요소들의 집합을 의미하며, 블록, 벽, 음식조각, 보드 위에 있는 셀 등의 실세계와 관련된 것들을 표현한다. [그림 2-2]는 작업메모리의 예로서 “테이블 위에 두 개의 블록이 있는데, 하나는 다른 하나 위에 있고, 하나는 테이블 위에 있다.” 는 상황을 표현한다.



[그림 2-2] Soar의 작업 메모리 요소로 표현된 테이블-블록 상황

[그림 2-3]은 Soar의 결정 사이클(decision cycle)을 나타낸 것으로, 전체적으로 연산자 선택(operator selection)과 연산자 적용(operator application)으로 구분할 수 있다.



출처: Laird, J. E. (2006). The Soar Cognitive Architecture. MIT Press. p.81.

[그림 2-3] Soar의 프로세스 사이클

연산자 선택에는 상태 정교화(state elaboration), 연산자 제안(operator proposal), 연산자 평가(operator evaluation)의 세부 단계가 있고, 연산자 적용에는 연산자 정교화(operator elaboration)와 연산자 적용(operator application)의 세부 단계가 있다.

상태정교화란 다른 작업 메모리 요소의 조합을 추상화한 것을 생성하고, 상태 공간에서 이를 새로운 증가물(augmentation)로 표현하는 작업을 의미한다. 상태정교화 규칙은 상태에서 복수개의 작업 메모리 요소 구조를 생성하며 병렬로 발화하는 복수개의 인스턴스들을 가질 수 있는데, 만약 어떤 작업 메모리 요소 구조가 현재 상태를 변화시키지 못하면 철회(retraction)되어서 모두 제거된다.

Soar 연산자의 선호도(preference)에는 acceptable, reject, better/worse, best, worst, indifferent 등의 기호 선호도와 0에서 1까지의 수치적 선호도가 있으며, 결정(decision) 단계에서 연산자 후보들의 선호도에 기반하여 하나의 연산자를 선택한다. 기본적으로 연산자는 Soar가 현재의 상태로 테스트 한 후 acceptable 선호도를 얻어야 제안(propose)되는데 연산자제안은 그 연산자의 이름과 파라미터 등을 선언하는 데 필요한 추가적인 작업 메모리 요소를 생성한다. 이렇게 후보 연산자가 제안되면 다른 연산자와 우선순위를 비교해서 선호도를 생성하는데 이 과정이 연산자평가이다.

이때 현재 연산자를 포함하는 작업 메모리 요소의 구조가 변화되는데, 만약 불충분하거나 충돌하는 선호도를 가진 연산자가 있다면 Soar는 교착상태(impasse)를 발생시키고 하위상태(substate)를 생성한다. 이 교착상태는 이후에 추가된 하위상태의 선호도에 의해 해소될 수 있으며, 청킹(chunking)은 이렇게 만들어진 규칙을 교착상태와 교체하여 새로운 규칙을 만든다.

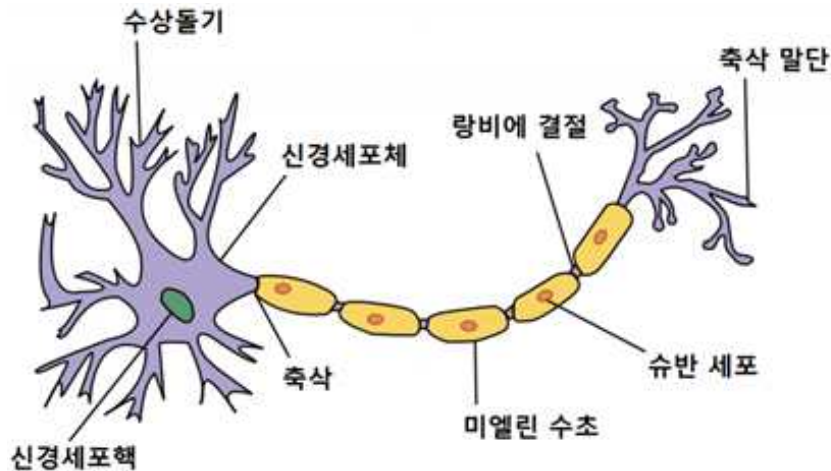
연산자적용 시 연산자가 선택되었는지를 테스트하고 그 후 연산자에 추가적인 구조를 생성하여 관련된 파라미터를 연결함으로써 연산자적용을 준비하는 연산자정교화와, 그

결과로서 상태에 지속적인 변화를 가하며 내부적 혹은 외부적 행위를 일으키는 연산자 적용 단계가 있다. 이 단계가 완성되면 출력(output)단에서 그 명령을 실행하고, 그 결과로서 환경 인식이 변화하면 이후의 연산자선택 단계에서 현재 연산자가 철회되고 새로운 연산자가 선택될 것이며 이 과정은 계속 반복된다.

Soar에 대한 보다 상세한 내용은 <http://soar.eecs.umich.edu/downloads/Documentation/SoarManual.pdf>에 접속해서 매뉴얼을 다운로드하면 매뉴얼 안에 잘 설명되어 있다.

2. 심층신경망

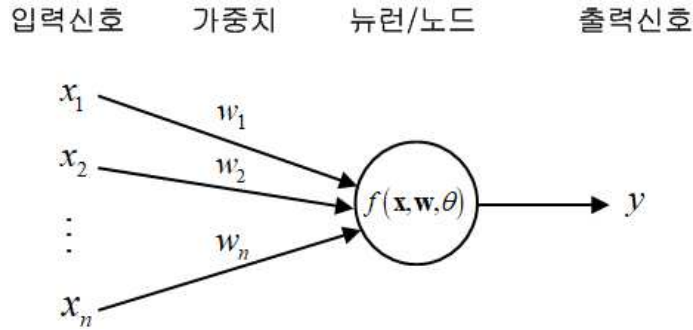
심층신경망(deep neural network, DNN)은 인공신경망(artificial neural network)을 확장한 것이다. 인공신경망은 인간의 뇌를 기반으로 한 추론모델이며, 인간의 뇌는 약 1,000억개의 신경세포(neuron)와 각 뉴런을 연결하는 6조개의 시냅스(synapse)로 구성되어 있다. 각각의 뉴런은 [그림 2-4]처럼 신경세포체(soma), 수상돌기(dendrite), 축삭돌기(axon), 축삭종말(axon terminal)로 구성되어 있다.



[그림 2-4] 인간의 신경세포 구조

인간은 새로운 내용을 학습할 때 뇌에 있는 신경세포(뉴런)들이 축삭돌기를 뻗어 인접한 뉴런의 수상돌기에 접근함으로써 연결 부위인 시냅스를 형성한 후 전기적 신호를 전달하고, 동일한 학습을 반복하면 이 연결 강도가 더욱 강해져서 오랜 시간이 지나도 그 내용을 다시 기억해 낼 수 있다.

이러한 신경생리학적 과정을 수학적으로 가장 간단하게 모사한 신경망(neural network)이 1958년 로젠블라트(ROSENBLATT)가 발표한 단층 퍼셉트론(perceptron)이다. 단층 퍼셉트론은 [그림 2-5]처럼 입력층과 뉴런/노드, 출력층으로만 구성되어 있으며, 입력값이 어느 영역에 속하는지 분류하는 목적으로 개발되었다.

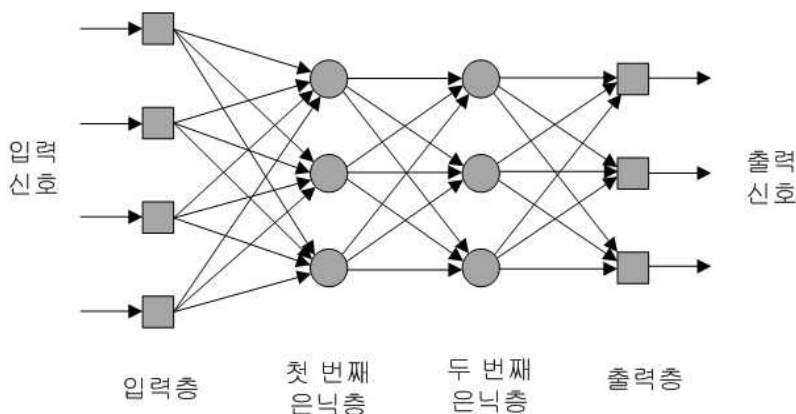


[그림 2-5] 인간의 신경세포 구조를 모사한 단층 퍼셉트론

뉴런/노드는 입력 신호의 가중치 합을 계산하여 임계값 θ 이상이 되면 **계단 함수나 시그모이드(sigmoid) 함수 같은 활성화 함수(activation function)**를 출력신호로 보낸다. 시그모이드 함수를 활성화 함수로 쓴 경우를 식으로 표현하면 다음과 같다.

$$y = f\left(X = \sum_{i=1}^n x_i w_i - \theta\right) = \frac{1}{1 + e^{-X}} \quad \text{식 (1)}$$

1962년 로젠블라트는 **센서(sensor) 유닛, 연계(association) 유닛, 반응(response) 유닛의 3개 유닛**으로 확장 구성된 퍼셉트론 이론을 발표했는데, 이를 **다층 퍼셉트론(Multi-Layer Perceptron)**이라고 한다. 이는 입력층(input layer), 은닉층(hidden layer), 출력층(output layer)으로 구성된 피드포워드 신경망(feedforward neural network)이라고도 불리며, DNN의 기본구조를 이룬다. [그림 2-6]은 2개의 은닉층이 있는 다층 피드포워드 신경망을 보인다.



[그림 2-6] 두 개의 은닉층이 있는 다층 퍼셉트론(신경망)

피드포워드 신경망에서 입력층은 입력 데이터를 받아들이는 기능을 하며, **입력층의 노드 (뉴런에 대응) 개수는 입력 데이터의 특성값의 개수와 동일하다.**

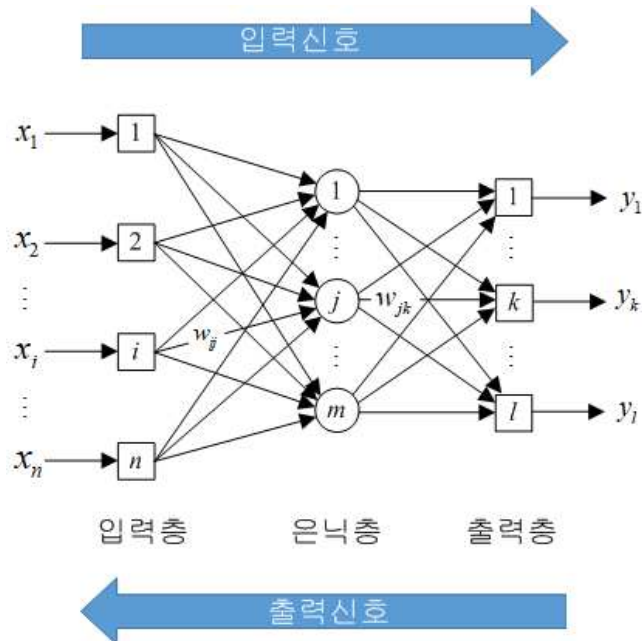
피드포워드 신경망 은닉층에 있는 노드는 두 가지 역할을 하는데, 첫 번째는 앞 층에서

전달받은 데이터를 가중치를 고려해 선형적으로 합산하고 식 (1)의 활성화 함수(activation function)로 계산한 값을 다음 층의 뉴런으로 전달한다. 은닉층은 여러 층이 될 수 있으며, 노드 수가 너무 많으면 오버피팅(overfitting)이 발생할 수 있고 너무 적으면 입력 데이터를 충분히 표현하지 못할 수 있으므로 적절한 개수를 정해야 한다.

피드포워드 신경망의 출력층은 풀고자 하는 문제의 해의 구성요소와 관계되어 있는데, 필기체 숫자 인식의 경우 출력층의 노드는 0부터 9까지 숫자에 매핑되는 10개의 노드로 구성된다.

피드포워드 신경망에서 원하는 결과를 얻기 위해서는 각 노드를 잇는 링크가 갖는 값인 가중치(weight)를 적절한 값으로 조정해야 하는데 이것을 학습(learning)이라고 한다.

역전파(back propagation)는 라벨이 붙은 학습 데이터를 가지고 여러 개의 은닉층을 가지는 피드포워드 신경망을 학습시키는 데 사용되는 대표적 지도학습(supervised learning) 알고리즘이다. 역전파는 [그림 2-7]에서 볼 수 있듯이 두 단계로 구성되는데, 첫 번째 단계에서는 학습 데이터를 신경망의 입력층에 전달하고 각 은닉층의 노드와 링크들을 거치며 계산된 값을 출력층 노드에서 출력한다. 두 번째 단계에서는 출력층의 출력 값과 라벨 값과의 오차를 최소화하도록 출력층부터 역방향으로 오차제곱합(sum squared error)의 경사도(gradient) 정보를 이용하여 가중치를 조정한다.

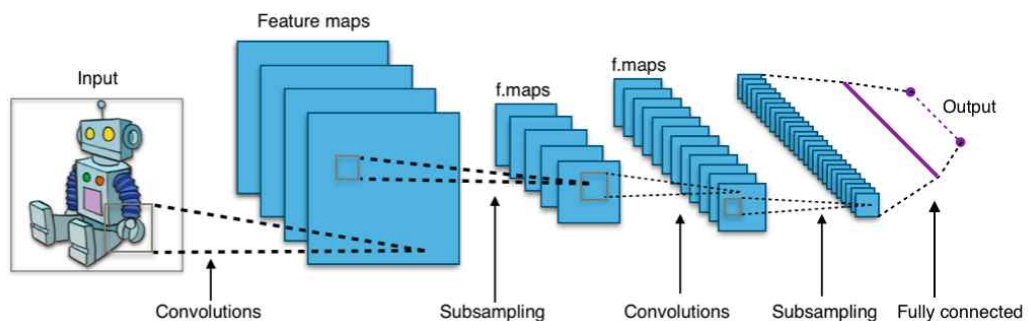


[그림 2-7] 세 개의 층이 있는 역전파 신경망

딥러닝(deep learning)의 한 분야로 영상인식에 많이 사용되고 있는 컨벌루션 신경망(convolutional neural network)의 핵심기술들을 알아보면 다음과 같다.

- 사람의 시각인지 과정을 모방한 컨벌루션(convolution) 기법을 이용한 필터 커널(kernel) 사용
- 컨벌루션이 완료되면 각 픽셀에 있는 데이터를 활성화 함수에 적용해 판별력 강화. 활성화 함수로 ReLU(rectified linear unit) 함수가 많이 사용됨
- 처리할 데이터를 줄이기 위해 이웃한 데이터 간 대비율(contrast)을 높이고 데이터 수를 줄이는 풀링(pooling) 또는 서브샘플링(sub-sampling) 시행. 풀링에는 최대 풀링(max pooling), 평균 풀링(average pooling), 확률적 풀링(stochastic pooling)이 있음
- 상기 **필터링 - 활성화 - 풀링** 과정을 반복적으로 거치면서 좀 더 확대된 형태의 이미지 구성
- 마지막 풀링이 끝나면 데이터가 1차원 벡터 형태로 정렬되고 이 값들은 전체 연결 신경망(fully-connected network)의 입력층으로 보내짐. 이후에는 몇 개의 추가적인 다층 신경망을 통해 피드포워드 및 역전파 과정을 거치며 학습이 진행됨

[그림 2-8]은 상기 과정을 통해 CNN이 입력 영상을 학습하는 과정을 보인다.



출처: https://en.wikipedia.org/wiki/Convolutional_neural_network

[그림 2-8] 컨벌루션 신경망을 이용한 이미지 인식 과정

2012년 ImageNet의 영상인식 경연대회에서 압도적인 물체 인식률(오류율 16.4%)로 우승하며 AI기술의 혁신을 일으킨 제프리 힌튼(GEOFFREY HINTON) 교수 팀의 SuperVision이 사용한 신경망 모델은 7개의 은닉층, 65만 개의 뉴런과 6,000만개의 변수, 6억 3,000만 개의 네트워크 연결로 구성되어 있다. 최근 이미지 인식 연구에서 마이크로소프트의 연구팀이 만든 ResNet은 무려 150개의 은닉층을 사용해서 예측 정확도를 높일 정도로 DNN의 구조는 점점 복잡해지고 있다.

DNN이 로봇에 적용된 사례는 다음과 같다.

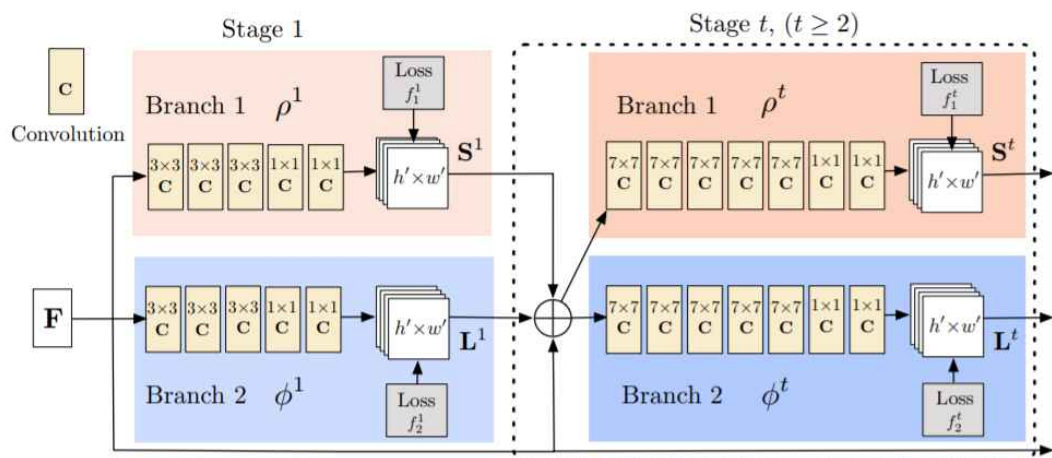
- **영상인식과 음성인식**
- **실시간으로 여러 사람의 자세 추정**
- **실시간 행동분석**
- **임의 형상을 가진 물체의 픽-플레이스**

[그림 2-9]는 카네기멜론 대학교에서 openCV와 CNN을 기반으로 개발한 OpenPose가 동영상으로부터 실시간 자세 추정을 하는 장면을 보이고, [그림 2-10]은 OpenPose의 CNN구조를 보인다.



출처: <https://www.youtube.com/watch?v=WoFuxJZgKSA>

[그림 2-9] OpenPose 패키지를 이용한 여러 명의 실시간 자세 추정



출처: Cao, Simon and Wei et al.(2017), Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields, arXiv:1611.08050v2.

[그림 2-10] OpenPose를 구성하는 CNN 구조

3. 강화학습

강화학습(reinforcement learning)은 **행동심리학의 강화(동물이 시행착오를 통해 학습하는 방법)**를 인공지능에 접목한 알고리즘으로서 환경에 대한 사전지식이 없어도 환경과 상호 작용하면서 학습할 수 있다. 스키너(SKINNER)의 실험에서, 굶긴 쥐가 상자를 돌아다니다가 우연히 지렛대를 밟았는데 먹이가 나오고(보상) 이 과정이 몇 번 반복되면 쥐는 먹이와 지렛대 사이의 관계를 학습하게 되는데 이 과정이 바로 강화(reinforcement)이다.

강화학습은 에이전트와 환경으로 구성되는데, 에이전트는 강화학습을 통해 스스로 학습하는 주체를 의미하며, 환경은 에이전트에게 보상을 주고 다음 상태를 알려준다.





보상은 양수로 설정하면 상이고 음수로 설정하면 처벌이 된다. 에이전트는 적절한 상벌을

통해 무엇을 해야 하는지 더 명확하게 알 수 있다.

순차적으로 행동을 결정하는 강화학습 문제를 정의할 때 사용하는 방법인 MDP(markov decision process)은 상태, 행동, 보상함수, 감가율, 정책으로 구성된다.

- 상태(state): 에이전트의 상황에 대한 관찰 값. [그림 2-11]에 보인 그리드 월드(grid world)의 상태 집합은 식 (2)와 같음. 예를 들어 시간 t 에 에이전트가 (1,3)에 있으면 $S_t = (1,3)$ 으로 표현함

$$S = \{(1,1), (1,2), (1,3), \dots, (5,5)\} \quad \text{식 (2)}$$

 (1,1)	(2,1)	(3,1)	(4,1)	(5,1)
(1,2)	(2,2)	(3,2)	(4,2)	(5,2)
(1,3)	(2,3)	 (3,3)	(4,3)	(5,3)
(1,4)	 (2,4)	 (3,4)	(4,4)	(5,4)
(1,5)	(2,5)	(3,5)	(4,5)	(5,5)

[그림 2-11] 5개의 행과 열로 구성된 그리드 월드

(1) 행동(action)

에이전트가 상태 S_t 에서 할 수 있는 가능한 행동의 집합. 그리드 월드의 행동 집합은 식 (3)과 같음. 시간 t 에 에이전트가 오른쪽으로 이동하면 $A_t = right$ 로 표현함

$$A = \{up, down, left, right\} \quad \text{식 (3)}$$

(2) 보상함수(reward function)

환경이 에이전트에게 주는 정보. 식 (4)는 시간 t 일 때 에이전트의 상태가 $S_t = s$ 이고 행동이 $A_t = a$ 일 때 에이전트가 받을 보상값이며, E는 기대값(expectation)을 의미함. 에이전트는 다음 시간인 $t+1$ 에 환경으로부터 보상값 R_{t+1} 을 받는 것에 주의해야 함. [그림 2-11]의 그리드 월드에서는 사각형으로 표현된 에이전트가 원이 있는 상태로 가는 행동을 했을 때는 +1의 보상을, 삼각형이 있는 상태로 가는 행동을 했을 때는 -1의 보상을 받음

$$R_s^a = E[R_{t+1} | S_t = s, A_t = a] \quad \text{식 (4)}$$

(3) 감가율(discount factor)

나중에 받게 될 보상일수록 현재보다 가치가 줄어드는 것을 반영하기 위한 계수. 감가율은 0과 1 사이의 값인 γ 로 표기하며, $\gamma^{k-1} R_{t+k}$ 은 현재로부터 시간이 k 만큼 지난 후에 받는 보상값 R_{t+k} 를 현재 시점에서 계산한 값임

(4) 정책(policy)

모든 상태에서 에이전트가 할 행동의 집합이며, 상태가 입력으로 들어오면 행동을 출력으로 보내는 일종의 함수. 에이전트는 강화학습을 통해 최적 정책을 학습해야 함. 식 (5)는 시간 t 에 에이전트의 상태가 $S_t = s$ 일 때 가능한 행동 중에서 $A_t = a$ 를 할 확률을 의미함

$$\pi(a|s) = P[A_t = a | S_t = s] \quad \text{식 (5)}$$

(5) 가치함수(value function)

어떠한 상태에 있으면 앞으로 얼마의 보상을 받을 것인지에 대한 기댓값. 식 (6)은 정책을 고려한 가치함수로서 현재 상태의 가치함수와 다음 상태의 가치함수 간의 관계를 알려주는 벨만 기대방정식(Bellman expectation equation)이다.

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \quad \text{식 (6)}$$

(6) 큐함수(Q function)

어떤 상태에서 어떤 행동이 얼마나 좋은지 알려주는 함수이며 에이전트가 정책을 업데이트할 때 많이 사용함. 큐함수는 상태, 행동이라는 두 가지 변수를 가지며 식 (7)에서 보듯이 가치함수의 조건문에 행동이 추가된다는 것이 다르다.

$$q_\pi(s, a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] \quad \text{식 (7)}$$

(7) 벨만 최적 방정식

에이전트가 최적의 정책을 따라갔을 때 받은 보상의 합

$$v^*(s) = \max_a E[R_{t+1} + \gamma v^*(S_{t+1}) | S_t = s, A_t = a] \quad \text{식 (8)}$$

4. 최적화 알고리즘

최적화 알고리즘(optimization method)은 최소화해야 할 비용 함수(cost function)의 해를 수학적 방법이 아닌 컴퓨터 알고리즘으로 탐색하는 방법으로, 컴퓨터 CPU 연산 속도의 비약적인 향상과 효율적인 탐색 알고리즘의 발전으로 로봇의 지능 소프트웨어에 점점 더 많이 적용되고 있다. 본 학습 모듈에서는 대표적 전역 최적화 알고리즘 중 코드 길이가 가장 짧으면서도 우수한 전역해 탐색 성능을 보이는 PSO를 사용한다.

(1) PSO(particle swarm optimization)

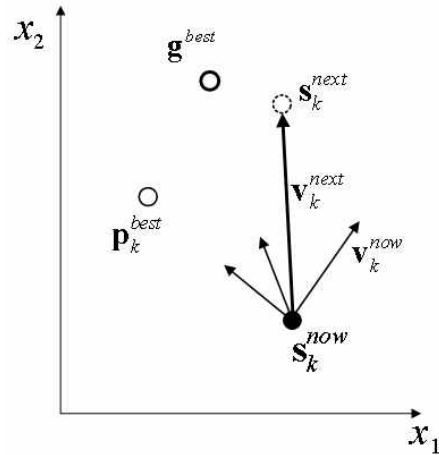
PSO는 KENNEDY와 EBERHART가 1995년 제안한 전역 최적화 알고리즘으로서 조류나 어류의 무리가 서로의 정보를 공유하면서 가장 먹이가 많은 곳을 찾아가는 과정을 컴퓨터 코드로 구현한 것이다.

PSO의 탐색 원리를 간단히 설명하면, 탐색변수가 n 개이고 총 개체 수가 N 인 한 집단에서 i 번째 반복(iteration)차수의 탐색 시 $k(k=1, \dots, N)$ 번째 개체의 위치벡터를 $\mathbf{s}_k^i \in R^n$ 이라고 하면, 그 개체는 지금까지 탐색했던 위치벡터 중 가장 우수한(비용함수를 최소화하는) 벡터를 \mathbf{p}_k^{best} 값으로 저장한다. 또 매 반복차수에서 전체 개체의 위치를 업데이트 한 후 가장 좋은 해의 위치벡터를 \mathbf{g}^{best} 라고 하면, PSO는 그 다음 업데이트 차수에서 k 번째 개체의 방향벡터 \mathbf{v}_k^{next} 와 위치벡터 \mathbf{s}_k^{next} 를 다음 식에 의해 결정한다.

$$\mathbf{v}_k^{next} = w\mathbf{v}_k^{now} + c_1 r_1 (\mathbf{p}_k^{best} - \mathbf{s}_k^{now}) + c_2 r_2 (\mathbf{g}^{best} - \mathbf{s}_k^{now}) \quad \text{식 (9)}$$

$$\mathbf{s}_k^{next} = \mathbf{s}_k^{now} + \mathbf{v}_k^{next}, k=1, \dots, N \quad \text{식 (10)}$$

식에서 \mathbf{v}_k^{now} 와 \mathbf{s}_k^{now} 는 현재 k 번째 개체의 방향벡터와 위치벡터를 각각 나타내며, 위치벡터에 방향벡터를 더함으로써 다음 차수의 위치벡터를 결정한다. 식에서 w 는 방향벡터의 관성 가중치(inertial weight)를, c_1 과 c_2 는 학습요인(learning factor)의 가중치 계수를 나타내며, r_1 과 r_2 는 주변점을 랜덤하게 탐색하기 위해 0과 1 사이에서 발생시킨 균일분포난수(uniformly distributed random number)를 의미한다. [그림 2-12]는 2차원 탐색 공간에서 PSO의 k 번째 개체에서 식 (9)와 식 (10)을 이용하여 다음 위치벡터를 결정하는 원리를 벡터도로 표현한 것이다.



[그림 2-12] 2차원 탐색 공간에서 PSO의 k 번째 개체가 현재 위치벡터(s_k^{now})로부터 다음 위치벡터(s_k^{next})로 이동하는 원리

② 작업지능

로봇의 작업지능은 ‘변화하는 작업 환경에서 계획된 작업을 로봇이 자율적으로 수행할 수 있는 능력’으로 정의할 수 있다.

로봇의 작업지능에는 전통적인 매니플레이션(manipulation) 기술을 기반으로 공유자율화(shared autonomy) 기술, 인간 조작 모사 기술, 클라우드 로봇 기술 등이 필요하다.

1. 매니플레이션

조작기술이라고도 불리는 매니플레이션은 로봇의 링크와 관절모터, 작업툴(end effector, 그립퍼 또는 핸드)로 구성되어 있는 매니플레이터(manipulator)가 작업 환경과 상호작용을 하면서 사용자가 목표한 대로 대상 물체나 환경에 물리적인 변화(물체의 이동, 결합, 조립, 형상 변경 등)를 가하는 것을 의미한다. [그림 2-13]은 6개의 액추에이터로 구성된 6자유도 매니플레이터를 나타낸다.



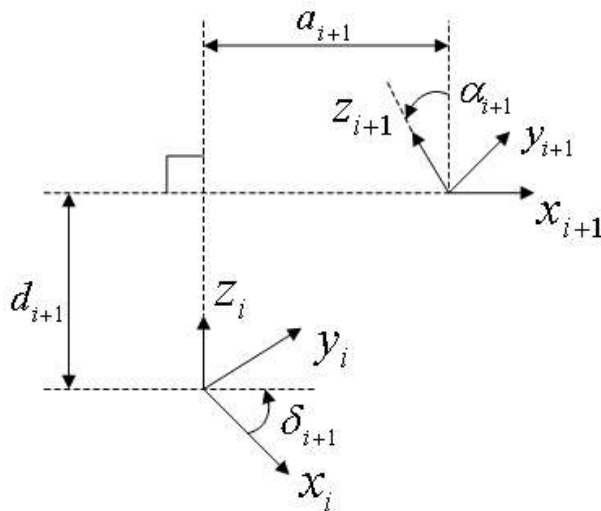
[그림 2-13] 6자유도 매니플레이터

매니플레이션은 주로 산업용 로봇에게 핵심적으로 요구되어 온 기능이며, 사람과 공존하고 협업하는 로봇(케어 로봇, 서비스 로봇, 협동로봇 등)에도 필수적으로 중요한 기술이다.

매니플레이션에는 물체를 잡고 목표 지점으로 이동하기 위한 작업 계획(planning) 수립, 이를 구현하기 위한 관절모터의 궤적 생성 기술, (필요 시)여유자유도 제어 기술, 그리고 이 과정에서 힘/토크 센서 데이터의 신호처리 및 센서융합(sensor fusion) 기술 등이 필요하다.

(1) 작업 계획

매니플레이터로 원하는 작업을 수행하기 위해서는 순기구학(forward kinematics)과 역기구학(inverse kinematics)에 대해 먼저 알아야 한다. 순기구학은 매니플레이터를 구성하는 링크의 길이와 관절의 각도값들이 주어졌을 때 작업툴 좌표계의 3차원 위치와 방향을 계산하는 것이고, 반대로 역기구학은 매니플레이터 작업툴 좌표계의 3차원 위치와 방향이 주어졌을 때 이렇게 만들기 위한 모든 관절의 각도값을 계산하는 것이다. 기구학을 적용하기 위해서는 먼저 로봇의 구조를 수학적으로 모델링해야 하는데 일반적으로 가장 많이 사용하는 방법은 Denavit-Hartenberg(DH)법으로서 4개의 파라미터로 인접한 두 링크 간의 좌표계 관계를 표현할 수 있다. [그림 2-14]는 지역 좌표계인 $x_i y_i z_i$ 좌표계와 $x_{i+1} y_{i+1} z_{i+1}$ 좌표계의 변환 관계를 관절각(δ_{i+1}), 링크 오프셋(d_{i+1}), 링크 길이(a_{i+1}), 링크 트위스트(α_{i+1})로 표현한 것이다. 이 파라미터들의 아래첨자가 $i+1$ 인 이유는 이 값들이 변화함으로써 $x_i y_i z_i$ 좌표계 기준으로 $x_{i+1} y_{i+1} z_{i+1}$ 좌표계가 회전 또는 이동하기 때문이다.



[그림 2-14] DH법을 이용한 인접 좌표계 간의 기하학적 관계

어떤 매니퓰레이터가 n 개의 관절(관절각 벡터는 $\mathbf{q} = [q_1 \cdots q_n]^T$)로 구성되어 있고, 작업툴 좌표계의 3차원 위치와 방향 벡터가 $\mathbf{x} = [\mathbf{p}_e \ \phi_e]^T$ 로 표현된다면, 순기구학은 다음과 같이 \mathbf{q} 에 대한 함수로 표현된다.

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \quad \text{식 (11)}$$

이에 반해 역기구학은 작업툴 좌표계의 벡터 \mathbf{x} 가 주어졌을 때 그에 상응하는 관절각 벡터 \mathbf{q} 를 계산하는 것이므로 다음의 식으로 표현된다.

$$\mathbf{q} = \mathbf{g}(\mathbf{x}) = \mathbf{f}^{-1}(\mathbf{x}) \quad \text{식 (12)}$$

순기구학과 달리 역기구학의 해는 존재하지 않거나 해가 존재하더라도 하나 이상의 해가 존재할 수 있다. 역기구학은 크게 수학적식을 이용하는 해석적인(analytic) 방법, 매니퓰레이터의 기하학적(geometric) 특성을 이용하는 방법, 그리고 최적화 알고리즘을 이용하는 수치해석(numerical analysis)적 방법으로 나눌 수 있다.

(가) 해석적인 방법

해석적인 방법은 일반적으로 많이 사용되며, 정완균·박재영·최영진 외(2012)에 의하면 파이퍼(pieper)의 조건(인접한 세 좌표계의 회전축이 한 점에서 교차하면 분석적인 방법의 역기구학의 해가 언제나 존재한다)을 만족하지 않으면 역기구학 해가 존재하지 않을 수 있으며, 6자유도의 PUMA 로봇처럼 자유도가 높아질수록(관절 수가 많아질수록) 수식으로 유도하기가 점점 복잡해진다.

(나) 기하학적 특성을 이용한 방법

기하학적 특성을 이용한 방법은 각 관절이 작업툴 좌표계의 위치와 방향에 미치는 기하학적 관계와 영향을 분석하고 이로부터 해당 관절의 적절한 각도를 계산하는 원리를 이용한다. 예를 들어 PUMA 로봇의 경우 어깨로부터 팔꿈치에 해당하는 3개의 관절(그룹 1)과 팔꿈치부터 손목까지의 3개의 관절을 두 그룹(그룹 2)으로 분리하고, 그룹 1은 작업툴 3차원 좌표계의 위치를 정하는 역할을 하고, 그룹 2는 작업툴 3차원 좌표계의 방향을 조정하는 역할을 부여한 후 기하학적 조건들을 따져 총 6개의 관절각을 계산한다.

(다) 수치해석적인 방법

수치해석적인 방법은 컴퓨터 CPU 성능의 향상과 최적화 알고리즘의 발전으로 로봇의 기구학 모델의 특징과는 상관없이 해가 존재한다면 짧은 시간 안에 찾아낼 수 있는 장점이 있다. 전역 최적화 알고리즘 중 코드 구조가 간단하면서도 전역해를

안정적으로 찾아내는 PSO가 많이 사용된다.

(2) 관절모터의 궤적 제작 기술

로봇이 어떤 동작을 안정적이면서도 정확하게 구현하기 위해서는 제어기가 모든 관절 모터에 매 샘플 시간마다 적절한 목표각(reference angle) 값을 전송하고, 각 모터는 이 각도 값을 정확히 추종할 수 있어야 한다. 이러한 목표각의 일정시간 간격 값들을 배열(array) 형태로 로봇 제어기의 메모리에 저장해 둘 경우 메모리 사용량이 상당히 클 뿐만 아니라, 샘플 시간이나 로봇 모션의 속도 변화 시 그에 맞춰 목표각 배열값들을 다시 계산해야 한다. 그러나 소수의 파라미터로 궤적의 모양을 조절할 수 있는 다항함수(polynomial function)로 각 관절모터의 회전각 궤적을 표현하면 목표각 데이터 저장 및 변경에 효율적으로 대응할 수 있다.

혼합다항함수법(blending polynomial method)은 어떤 관절의 전체 회전각 궤적을 몇 개의 세그먼트(segment)로 분할하고 각 세그먼트의 시작점과 끝점인 두 경유점(via point)에서 이웃한 세그먼트의 각도와 각속도를 동일하게 유지함으로써 전체적으로 부드러운 궤적을 이어 붙이는 방법이다. 각 세그먼트는 각도와 각속도의 연속성을 만족시키기 위해 3차 이상(각가속도의 연속성까지 만족시키기 위해서는 5차)의 다항식으로 표현되어야 하므로, 어떤 궤적이 m 개의 세그먼트로 분할될 경우 생성된 $m+1$ 개의 경유점에 다음과 같은 조건들이 주어진다.

$$\begin{aligned} q(t_0) &= q_0, q(t_1) = q_1, \dots, q(t_m) = q_m, \\ \dot{q}(t_0) &= \omega_0, \dot{q}(t_1) = \omega_1, \dots, \dot{q}(t_m) = \omega_m, \\ \ddot{q}(t_0) &= \alpha_0, \ddot{q}(t_1) = \alpha_1, \dots, \ddot{q}(t_m) = \alpha_m \end{aligned} \quad \text{식 (13)}$$

여기서 q_k, ω_k, α_k 는 $k(k=0, \dots, m)$ 번째 경유점에서 관절이 회전하는 각도, 각속도, 각가속도를 각각 나타낸다.

혼합다항함수 궤적의 세그먼트 경유점에서 각도와 각속도의 연속성만 만족해도 되는 경우 $i(i=1, \dots, m)$ 번째 세그먼트(해당 경유점 시간은 t_{i-1} 과 t_i)의 수식은 다음과 같이 3차 다항식으로 표현할 수 있다.

$$q_i(t) = a_0^i + a_1^i(t - t_{i-1}) + a_2^i(t - t_{i-1})^2 + a_3^i(t - t_{i-1})^3, \quad t_{i-1} \leq t \leq t_i \quad \text{식 (14)}$$

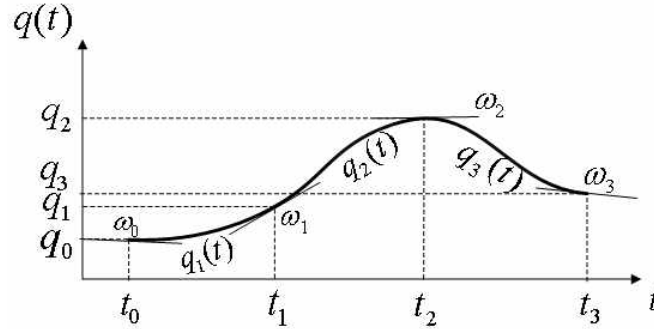
여기서 4개의 계수 $a_j^i, j=0, \dots, 3$ 은 식 (13)에 나타낸 각도 관련 조건식들을 모두 만족시키는 미지수이므로 식 (13)과 식 (14)를 이용해서 다음 관계식을 도출할 수 있다.

$$a_0 = q_{i-1}, \quad a_1 = \omega_{i-1},$$

$$a_2 = \frac{3(q_i - q_{i-1}) - (2\omega_{i-1} + \omega_i)(t_i - t_{i-1})}{(t_i - t_{i-1})^2}, \quad a_3 = \frac{2(q_{i-1} - q_i) + (\omega_{i-1} + \omega_i)(t_i - t_{i-1})}{(t_i - t_{i-1})^3}$$

식 (15)

[그림 2-15]는 3개의 세그먼트로 구성된 혼합다항함수 궤적을 예시한 것으로, 로봇 작업
툴의 위치 제어를 위한 관절 회전각 궤적을 만들어내는 데 큰 문제없이 사용할 수 있다.



[그림 2-15] 3개의 세그먼트로 구성된 혼합 다항식 궤적

(3) 여유자유도 제어

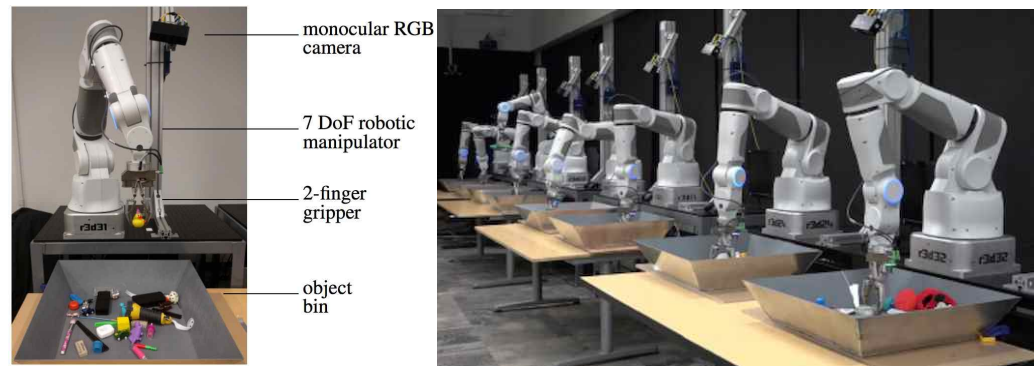
어떤 매니퓰레이터의 작업툴 좌표계가 3차원 공간에서 원하는 위치(3자유도)에 놓이고
원하는 방향(3자유도)으로 향하게 만들기 위해서는 매니퓰레이터의 관절이 최소 6개여
야 한다. 같은 원리로 2차원 공간에서 작업툴의 원하는 위치와 방향을 모두 만족시키
는 자세를 만들기 위해서는 매니퓰레이터의 관절은 최소 4개 이상이어야 한다.

3차원 매니퓰레이터의 관절이 6개보다 많은 경우 전체 관절 수에서 6(2차원 매니퓰레
이터에서는 4)을 뺀 수가 여유자유도다. 여유자유도가 1 이상이면 복수 개의 역기구학
해를 얻을 수 있어서 작업툴의 위치나 방향을 변화시키지 않은 채 관절을 회전시킬
수 있기 때문에, 장애물을 회피하면서 매니퓰레이터가 작업을 수행할 수 있게 하거나
작업 조건이 복수 개라도 이를 최대한 만족시키면서 작업을 수행할 수 있게 하는 장
점이 있다. 여유자유도는 영 공간(null space)과 관련되어 있다.

(4) 딥러닝(deep learning) 기법을 이용한 파지(grasp) 작업 학습

Google-X 연구소에서는 손-눈 협응(hand-eye coordination)이라고 불리는 일종의 시각
제어(visual servoing) 기법을 이용한 자율적 만능 파지(grasp)기술 개발을 위해 [그림
2-16]과 같이 DNN(deep neural network)의 일종인 CNN(convolutional neural network)
을 사용해서 단안(monocular) RGB 카메라와 7자유도 관절을 가진 매니퓰레이터 14대
로 다양한 작업환경과 조명, 작업툴 마모 상태에서 총 80만 번(로봇 당 약 3,000시간에
해당)의 파지 훈련을 시켰다. 그 결과 로봇들은 대부분의 물건들을(심지어는 처음 접하
는 물체들도) 사람처럼 능숙하고 안정적으로 잡아서 옮겼으며, 물건 더미에서 대상 물

체를 분리하는 예비파지(pre-grasp) 기능까지 갖춘 것으로 보였다. 그러나 이러한 접근법은 구조가 다른 로봇 매니퓰레이터에게 바로 적용할 수 없는 단점이 있다.



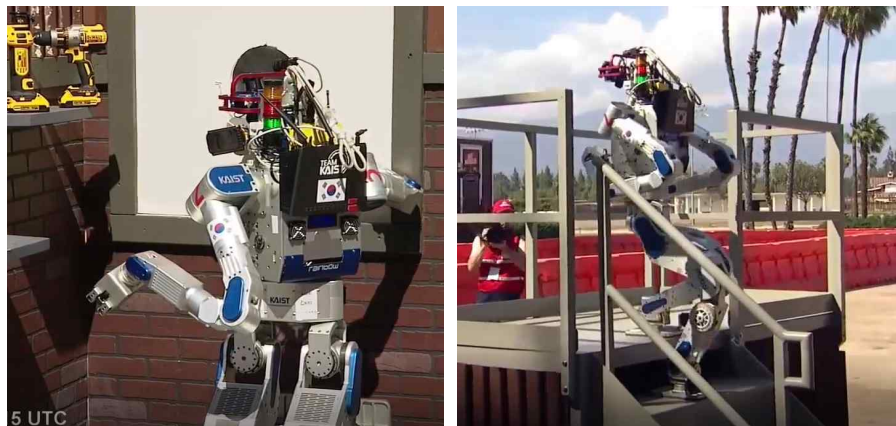
출처: Evan Ackerman(2016. 03. 28). How Google Wants to Solve Robotic Grasping by Letting Robots Learn for Themselves. <http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/google-large-scale-robotic-grasping-project.html>에서 2016. 10. 03. 검색.

[그림 2-16] Google의 로봇들이 만능 파지 기술을 학습하는 모습

2. 공유자율화

공유자율화(shared autonomy)는 로봇의 완전자율화(full autonomy)로 가는 전단계로서 로봇이 복잡한 작업을 수행하거나 인지적 판단이 필요할 때 인간의 명령이나 도움을 최소화 하면서 로봇의 수행 능력을 극대화하여 인간과 로봇이 효율적으로 분업 및 협업하는 기술이다.

최신형 텔레프레즌스(telepresence) 로봇의 경우 사용자가 로봇에게 목적지를 정해주면 로봇이 장애물 회피를 자동으로 수행하는데, 이는 공유자율화를 구현한 예라고 할 수 있다. 또 2015년 6월에 미국 캘리포니아주에서 개최된 DRC(DARPA robotics challenge)도 공유자율화 기술을 이용해서 인간 오퍼레이터와 로봇이 협업을 하여 8가지의 재난 현장 미션을 주어진 시간(60분) 안에 수행할 수 있었다. [그림 2-17]은 2015년 열린 DRC 결승전에서 우승한 한국의 DRC-HUBO가 미션을 수행하는 모습을 보인다.



[그림 2-17] 한국 KAIST의 DRC-HUBO가 DRC에서 미션을 수행하는 모습

공유자율화를 위해서는 원격 조작을 위한 휴먼 인터페이스 기술, 이를 위해 직관적이고 스마트한 방식으로 원격 조작 입출력을 매핑(mapping)하는 기술, 인간의 조작 명령을 로봇 매니퓰레이터의 역기구학 해로 자동 변환하는 기술, 주종(master-slave) 시스템 간의 전송/피드백 시간 지연 최소화 기술 등이 함께 개발되어야 한다.

공유자율화의 정도와 필요 기술을 작업 수행 단계별로 알아보기 위해, 여러 가지 물건들이 불규칙적으로 놓여 있는 테이블에서 특정 물체를 인식한 후 분류하거나 집어서 다른 곳으로 옮기는 작업에 대해 생각해 보자. 이 작업은 인간에게는 매우 쉽지만 로봇에게는 여전히 어려운 작업이다. 공유자율화의 각 수행 단계를 알아보면 다음과 같다.

(1) 정보 획득

카메라를 이용한 영상 획득, 레이저 거리측정장치(laser range finder, LRF)를 이용한 물체와의 거리 측정, 압력 센서를 이용한 접촉 감지, 로봇의 자세 측정 등의 모든 로봇 내외부 정보를 그대로 획득하는 단계로서, 로봇의 자율화 정도가 높은 편이다.

(2) 정보 분석

로봇이 획득한 센서 원본 데이터를 이용해서 물체를 인식하거나 해당 범주에 분류하는 작업이 정보 분석에 해당된다. 센서 데이터가 불완전하거나 잡음이 심한 경우, 또 대상 물체가 다른 물체에 가려져 있는 경우 정확하고 강인한(robust) 물체 인식 또는 분류가 상당히 어렵다. 그러므로 이 단계에서는 로봇의 자율화 정도가 낮은 편이며 인간의 개입이 필요하다.

(3) 판단과 행동 선택

이 단계에서는 로봇이 분석한 정보에 따라 상황을 판단하고 몇 가지 행동 후보들을 제안한다. 이동로봇의 조작 태스크를 예로 들면 주행과 관련해서는 어느 방향으로 갈 것인지, 조작과 관련해서는 어느 물체를 잡아야 할지, 잡을 물체가 정해졌다면 그 물체의 형상 특성에 따라 어떻게 잡아야 할지를 로봇이 결정하거나 인간에게 질의하는 것에 해당된다. [그림 2-18]은 백스터 로봇이 물체를 잡는 모습을 보인다.

많은 로봇들이 컵이나 병 같은 단순한 물체들은 잘 잡을 수 있지만, 가정이나 사무실 처럼 비정형적이고 복잡한 환경에서 특정 물건을 잡아야 하는 경우 인간이 개입하여 대상 물건과 이를 적절하게 잡는 방법을 알려 주는 것이 필요한 실정이다.

(4) 작업 구현

선택된 작업을 로봇이 수행하는 단계로서 작업의 난이도와 로봇의 지능 정도에 따라 로봇의 자율화 정도가 크게 다르다.

예를 들어 몰리 로보틱스(MOLEY ROBOTICS)사는 다양한 요리 도구들을 이용해서 음식물을 조리할 수 있는 양팔 로봇을 개발하기 위해 최고 수준의 인간 요리사의 요리 과정을 학습시키고 있다.



[그림 2-18] 백스터 로봇이 물체를 잡는 모습

3. 클라우드 로봇 기술

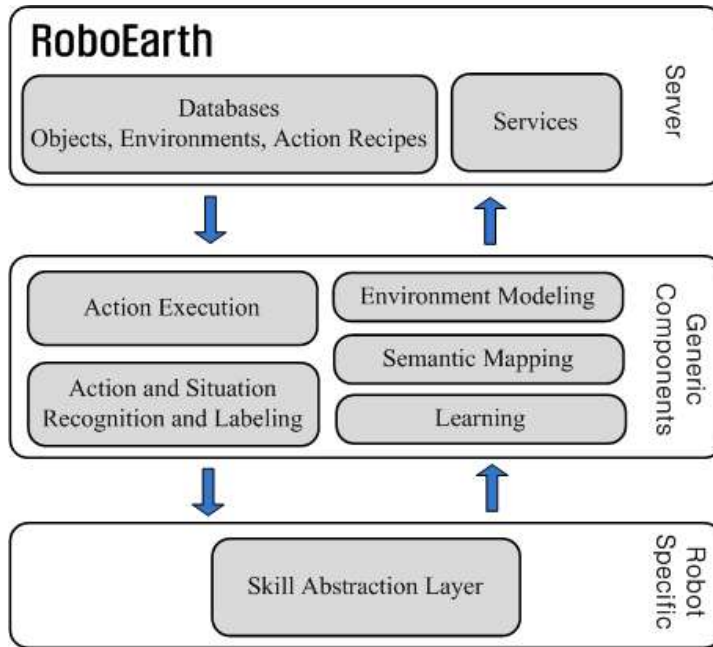
전 세계의 사람들이 인터넷과 모바일 시스템의 발달로 서로 연결되어 정보와 기술을 공유하고 소통함으로써 편리하고 윤택한 삶을 살고 있듯이, 로봇들도 서로 네트워크로 연결되어 필요한 데이터와 작업 지식들을 공유하고 재활용한다면 그 효율성은 크게 증대할 것이다.

RoboEarth는 2009년 말부터 4년간 유럽연합(EU)에서 지원을 받아 유럽의 6개 대학과 PHILIPS사가 공동으로 수행한 오픈소스 프로젝트로서 그 목적은 다음과 같다(<http://roboearth.org/>).

첫째, 로봇이 복잡한 태스크를 수행하는 데 있어서 필요한 학습과 적응(adaptation) 프로세스의 속도를 높이기 위해 네트워크로 연결된 정보 저장소(repository)를 만드는 것

둘째, 이러한 저장소에서 로봇들이 액션 레시피(action recipe)들을 공유함으로써 미리 프로그래밍 되어 있지 않은 새로운 작업을 용이하게 할 수 있음을 보이는 것

[그림 2-19]는 3층 구조의 RoboEarth 아키텍처를 보이는 것으로, 서버층(server layer)-일반 구성 요소층(generic component layer)-로봇 종속층(robot specific layer)으로 구성되어 있다.



출처: Waibel, Beets and Civera et al.(2011), Roboearth, IEEE Robotics & Automation Magazine, p. 71.
 [그림 2-19] 세 개의 층으로 구성된 RoboEarth의 아키텍처

(1) 서버층

RoboEarth의 서버층은 데이터베이스와 서비스로 구성되어 있다.

(가) 데이터베이스

RoboEarth의 데이터베이스는 로봇에게 필요한 데이터의 공유와 재사용성을 위해 아파치 하둡(Apache Hadoop) 기반으로 제작되었다. 데이터는 병렬로 저장되고, 데이터 간의 복잡한 의미론적(semantic) 관계는 별도의 그래프 데이터베이스에 W3C(world wide web consortium)-표준 OWL(web ontology language)로 코딩된 정보로 저장되었다.

데이터베이스는 전역 세계 모델(global world model)을 기본적으로 포함하고 있으며, 물체(objects)에 대해서는 물체의 종류, 3차원 크기, 영상, 포인트 클라우드, CAD 모델 등의 정보가 저장되어 있고, 환경(environments)에 대해서는 지도 영상, 물체의 위치와 자세, 물체를 마지막으로 본 시간 등의 정보가, 액션 레시피에 대해서는 다양한 작업에 대해 사람이 이해할 수 있는 계층 구조의 작업 절차 설명과 필요 기술(skill), 작업 파라미터(물체, 위치, 파지 방법의 종류) 등이 저장되어 있다.

데이터베이스의 인터페이스로서 로봇뿐만 아니라 사람도 접속하여 데이터에 접근하고 로봇이 인식한 물체에 라벨을 붙일 수 있는 HTML(hypertext mark-up language)과 서버에 정보를 저장하지 않은 채 요청의 일부로서 모든 필요한 정보들을 전송할 수 있는 REST(representational state transfer)를 사용한다.

(나) 서비스

데이터베이스 서비스는 액션 레시피에 대한 고급 수준의 설명에 대해 기본적인 학

습과 추론의 기능을 통해 해당 로봇이 가진 기술을 잘 매핑할 수 있게 하거나 로봇 타입에 따라 어떤 데이터가 안전하게 재사용될 수 있는지를 알려주는 기능을 한다.

KnowRob는 논리적 추론을 위해 데이터베이스의 시맨틱 정보를 이용하는데, 예를 들어 로봇 B가 로봇 A의 액션 레시피를 이용해서 작업하기 위해 필요한 요구 조건들을 만족시키는지, 동일한 물체를 인식하는 데 필요한 센서들을 갖추고 있는지 등을 판단한다.

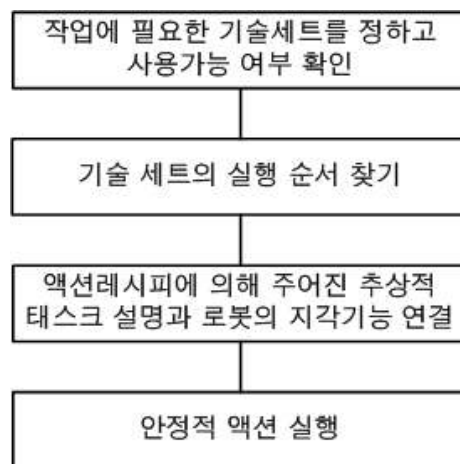
RoboEarth의 데이터베이스 서비스는 데이터베이스에 저장된 지식들을 분석해서 새로운 액션 레시피를 제작하고 이전의 정보들을 업데이트하는 기능을 가지고 있다. 예를 들어 컵이 컵 받침대 위에 있을 확률과 식탁 위에 있을 확률을 계산하여 로봇이 컵을 찾으러 갈 때 확률이 좀 더 높은 쪽을 향하게 한다.

(2) 일반 구성 요소층

일반 구성 요소층은 액션 실행, 환경 모델링, 시맨틱 매핑, 학습, 액션/상황의 인식 및 라벨링 기능으로 구성되어 있다.

(가) 액션 실행

액션 레시피에 따라 로봇은 [그림 2-20]의 순서로 작업을 실행한다. 그림에서 ‘기술’은 skill을 의미한다.



[그림 2-20] 액션 실행 순서

(나) 환경 모델링

환경 모델링은 RoboEarth 데이터베이스를 통해 사용 가능한 사전 정보와 로봇의 현재 환경에 대한 센서 입력을 조합한다. 이를 위해 지역(local)과 전역(global) 세계 모델 레벨을 사용한다. 지역 세계모델 레벨은 수십 Hz 이상의 높은 업데이트 주파수로 로봇의 3차원 인지 데이터와 추종 데이터를 융합하고 연결한다. 반면에 전역 세계모델 레벨은 RoboEarth 데이터베이스에 저장되어 있는 물체의 적절한 사전 정

보를 추출함으로써 물체의 지속성에 대해 간단히 판단하고 지역 세계 모델을 지속적으로(최대 1 Hz) 업데이트 하는 역할을 한다.

(다) 시맨틱(semantic) 매핑

RoboEarth 데이터베이스는 로봇의 기술들과 환경, 상호작용하고자 하는 물체들로 정의된 일반적 액션 레시피를 제공한다. RoboEarth는 로봇의 현재 위치 확인과 지도 작성을 동시에 수행하는 **SLAM(simultaneous localization and mapping)** 기법을 이용하여 환경에서 부분적으로 관찰된 물체를 데이터베이스로부터 다운로드한 인식 물체와 결합하는 일반화된 매핑 기능을 구현한다. 그 결과 기하학적으로 정확하게 작성된 SLAM용 지도에 RoboEarth로부터 다운로드 받은 인식 물체의 정보를 추가한 시맨틱 지도(semantic map)를 얻을 수 있다.

(라) 학습

RoboEarth의 학습 기능은 유사한 구조의 로봇 플랫폼에서 액션 레시피와 기술들을 반복적으로 사용함으로써 태스크 수행의 신뢰성과 성능을 향상시키게 한다. 또 학습 기능은 로봇이 컵이나 병을 부적절한 방법으로 잡으려 하는 경우 사람이 병을 잡는 방법에 대해 피드백 함으로써 로봇의 파지 성능을 향상시킬 수도 있다. 그리고 로봇은 RoboEarth의 데이터베이스에 저장된 다른 로봇의 태스크 수행 경험을 참조하여 작업 능력을 향상시킬 수도 있다.

(마) 액션/상황의 인식 및 라벨링

활용성이 높고 변환 가능한 액션 레시피와 이의 재활용은 일반적 기술 추상화층부터 고수준의 액션실행층까지 전체 로봇 액션 레벨들을 모두 다뤄야 하므로 상당히 어려운 기술이다. 인식 및 라벨링의 구성 요소는 다음의 두 계층으로 구성된다.

- 로봇의 센서로부터 얻은 신호를 이용하여 시간과 장소의 연관성 없이 기술을 인식하는 저수준층
 - 이미 인식된 저수준층의 프리미티브들을 이용해서 은닉 마르코프 모델(hidden markov model, HMM)로 액션과 상황의 시공간 관계를 학습하는 고수준층
- 예를 들어, 로봇 A가 어떤 방에 들어가려고 하는데 문이 닫혀 있어서 학습 기능으로 사람의 도움을 받아 문을 열고 들어갔고, 이를 RoboEarth 데이터베이스에 라벨을 붙여 저장했다고 한다. 로봇 B가 유사한 구조의 방에서 닫혀있는 문을 열려고 할 때 로봇 A의 상황과 유사하다는 사실을 인식하면 로봇 A가 학습한 액션 레시피를 변형하고 업데이트함으로써 로봇 B도 손쉽게 문을 열 수 있게 된다.

(3) 로봇 종속층

로봇 종속층은 로봇의 특정한 하드웨어 종속 기능들에 대한 일반적 인터페이스를 제공하므로 기술 추상화층이라고도 한다. 이 층은 MoveTo, MoveToObject, Grasp, Detect 등과 같이 표준화된 로봇 작업 기술의 부분집합을 제공하기 위해 로봇에 내재된 하드

웨어를 추상화한다. 이렇게 특정 로봇에만 가능한 기술의 부분집합은 어떤 태스크를 수행하기 위해 서로 다른 액션 레시피 중에서 하나를 선정할 때 중요한 요소가 된다. 예를 들어, 로봇 A는 전방향 휠을 가지고 있고, 로봇 B는 4개의 다리를 가지고 있다면 MoveTo라는 태스크를 수행하기 위해 두 로봇의 액션 레시피는 서로 달라야만 한다.

③ 매니퓰레이터 힘 제어

매니퓰레이터의 운동은 다음과 같이 크게 두 가지로 나뉜다.

- 자유공간에서 움직이는 운동: 용접, 도장, 픽-플레이스 등
- 외부환경과의 접촉을 유지하며 움직이는 운동: 연마, 디버링, 표면처리 등이며 접촉력 제어와 유연제어(compliant control)를 필요로 하는 작업으로 구분됨

로봇이 접촉을 유지하며 움직이는 작업을 잘 수행하기 위해서 필요한 기술이 바로 힘 제어(force/torque control)이다. 힘 제어를 수행하기 위해서는 다음의 단계가 필요하다.

- 작업의 목적을 명확하게 이해하기
- 작업을 수행하기 위한 기본 궤적 생성
- 작업 수행 중에 접촉 발생 시 힘 및 위치 피드백
- 피드백 데이터를 이용한 궤적 수정

1. 힘 제어의 분류

매니퓰레이터가 외부환경에 접촉운동을 수행할 경우 매니퓰레이터는 외부환경에 힘을 가하고, 고정된 외부환경으로부터 상응하는 힘이 반작용으로 발생한다. 이때 매니퓰레이터와 외부환경 사이에 발생하는 힘을 제어하는 방법에 따라 직접 힘 제어(direct force control)와 간접 힘 제어(indirect force control)로 분류된다.

(1) 직접 힘 제어

매니퓰레이터 말단에 힘/토크 센서를 부착하여 접촉운동 시에 발생하는 외력을 직접 측정하고, 매니퓰레이터가 생성하는 힘/토크가 측정된 외력과 동일해질 때까지 각 관절의 토크를 조절하거나 말단부의 위치를 조절하여 접촉력을 일정하게 유지하는 제어 기법. 힘 제어와 복합 위치/힘 제어가 이 범주에 속함

(2) 간접 힘 제어

매니퓰레이터에 외력이 작용할 경우, 외력에 비례하여 매니퓰레이터의 위치 및 속도를 제어하는 기법. 매니퓰레이터의 목표위치를 외부환경의 내부로 침투시키면 접촉 때문에 더 이상 목표위치를 추종하지 못하므로 해당 위치오차에 비례하는 힘이 외부환경에 작용하게 되어 간접적으로 힘 제어 가능. 강성 제어(stiffness control), 댐핑제어(damping control), 임피던스 제어(impedance control) 등이 이에 해당됨

2. 힘 제어를 구현하기 위한 매니퓰레이터

힘 제어를 구현하기 위한 매니퓰레이터는 다음과 같은 기능 및 특성이 필요하다.

- 매니퓰레이터의 관성 파라미터(질량, 질량중심, 관성모멘트) 및 동역학 모델을 알아야 함
- 관절 토크를 정밀하게 제어할 수 있어야 함
- 관절 위치 및 속도를 정밀하게 측정할 수 있어야 함
- 매니퓰레이터가 생성하는 힘을 측정할 수 있어야 함
- 로봇 제어기는 1kHz 이상의 충분한 대역폭을 가져야 하며, 실시간성 보장
- 실시간 연산 가능

각 관절의 토크를 제어할 수 있는 매니퓰레이터를 토크제어 기반 매니퓰레이터라고 하며, 이 경우 토크제어 알고리즘을 알아보자.

매니퓰레이터가 환경과 접촉하는 경우 매니퓰레이터의 운동방정식은 다음과 같이 표현된다.

$$M(q)\ddot{q} + n(q, \dot{q}) + J^T(q)f_e = \tau \quad \text{식 (16)}$$

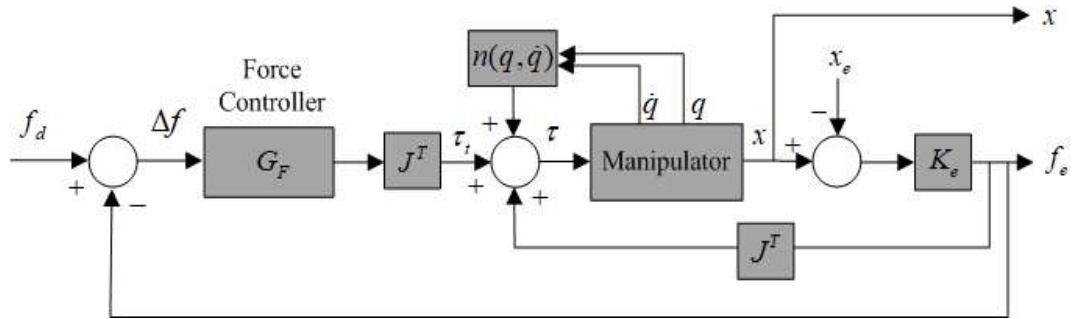
식에서 $M(q)$ 는 관성행렬, $n(q, \dot{q})$ 는 코리올리 및 구심력, 중력 및 마찰력 등을 포함하는 비선형 성분 벡터, τ 는 관절토크 벡터, f_e 는 매니퓰레이터 말단부와 환경 간의 접촉력을 각각 나타낸다.

만약 매니퓰레이터 말단부의 위치 x 가 환경의 위치 안쪽인 x_e 에 있게 하면 매니퓰레이터가 환경에 힘을 가하게 되어 환경의 변형이 발생한다. 환경의 강성을 K_e 라고 하면 접촉력 f_e 는 다음과 같이 표현된다.

$$f_e = K_e(x - x_e) \quad \text{식 (17)}$$

토크제어 기반의 힘 제어는 가장 직관적인 힘 제어 방법으로 [그림 2-21]과 같이 토크제어 기반 매니퓰레이터를 이용한다. 그림에서 f_d 는 외부환경에 인가하고자 하는 접촉력이며, 힘 오차 Δf 는 f_d 와 힘/토크 센서로 측정된 힘 f_e 사이의 오차이다. Δf 는 힘 제어기 G_F 및 자코비안 전치행렬 $J^T(q)$ 를 지나면서 접촉력 제어를 위한 토크 τ_t 로 전환되며, 목표 토크 τ_t 의 수식은 다음과 같이 표현된다.

$$\tau_t = J^T(q)G_F(f_d - f_e) \quad \text{식 (18)}$$



[그림 2-21] 토크제어 기반의 매니퓰레이터를 이용한 힘 제어기 블록선도

[그림 2-21]은 토크제어 기반의 매니퓰레이터를 이용한 힘 제어기의 블록선도를 나타낸다. 힘 제어기 G_F 는 다양한 제어기로 설계할 수 있으며, 비례적분(PID) 제어기도 사용 가능하다.

수행 내용 / 작업 수행 지능 설계하기

재료 · 자료

- 사용자 요구 분석서
- 프로그램 구조 설계서
- 알고리즘 기술서
- 프로그램 코드

기기(장비 · 공구)

- 컴퓨터, 프린터
- 프로그램 개발용 소프트웨어
- 문서 작성용 소프트웨어

안전 · 유의 사항

- 로봇 하드웨어로 사용되는 부품과 장치들은 고가이므로 작업을 수행하기 전에 시뮬레이터로 충분히 테스트하는 과정을 거쳐야한다. 이를 위해 로봇을 선택할 때 개발 환경 소프트웨어나

시뮬레이터가 제공되는지 여부를 확인한다.

- MATLAB의 경우 유료 소프트웨어이므로 교육기관에서 라이선스를 구매하여 네트워크 버전으로 학생들이 사용해야 한다.

수행 순서

① 소프트웨어의 목표 사양을 만족하는 작업지능을 분석한다.

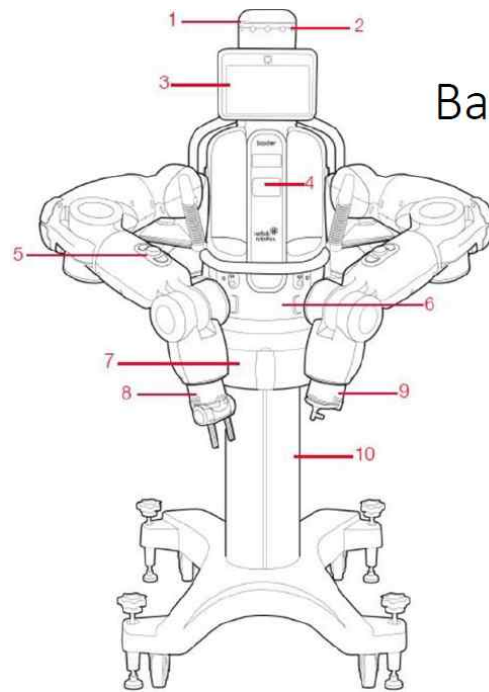
1. 수행해야 할 작업의 특성과 조건을 분석하고 적절한 제어 알고리즘을 결정한다.
 - (1) 수행해야 할 작업의 특성이 자유공간에서 움직이는 자유공간 작업(단순한 집기와 놓기, 용접, 도장 등)인지, 외부환경과의 접촉을 적절히 유지하며 접촉력 제어를 필요로 하는 접촉유지작업(연마, 표면처리 등)인지, 접촉력을 최소화하며 유연한 운동을 필요로 하는 유연접촉작업(부품 조립)인지를 구분한다.
 - (2) 자유공간 작업일 경우 위치 제어 알고리즘을 적용하며, 접촉유지작업이나 유연접촉 작업일 경우 토크 제어 알고리즘을 위치 제어 알고리즘에 추가하여 적용한다.
2. 수행해야 할 작업에 요구되는 지능을 분석한다.
 - (1) [그림 1-4]에 나타난 ALFUS를 참고하여 수행해야 할 작업의 복잡도, 환경의 복잡도, 인간-로봇 상호작용 정도별로 자율화 지수가 0~10 중에서 어느 정도인지를 파악한다.
 - (2) 전체 자율화 지수가 0에 가까우면 로봇은 작업자의 직접적 교시(teaching)에 의해 제작된 작업 계획을 수행하고, 자율화 지수가 10에 가까우면 로봇은 최적화 알고리즘이나 DNN을 이용하여 현재 작업에 대한 작업 계획을 자율적으로 생성하면서 작업을 실시한다.

② 로봇과 주변장치 인터페이스를 분석하고 그 결과에 따라 알고리즘을 도출한다.

1. 로봇과 주변장치 인터페이스에는 각종 센서, 제어기, 디스플레이 등이 있다. 백스터에 대해 하드웨어와 소프트웨어를 분석한다.
 - (1) 백스터 하드웨어

[그림 2-22]를 통해 백스터의 주변장치와 사용자 인터페이스를 분석한다(자세한 자료는 sdk.rethinkrobotics.com/wiki/Hardware_Specifications 사이트 참고).

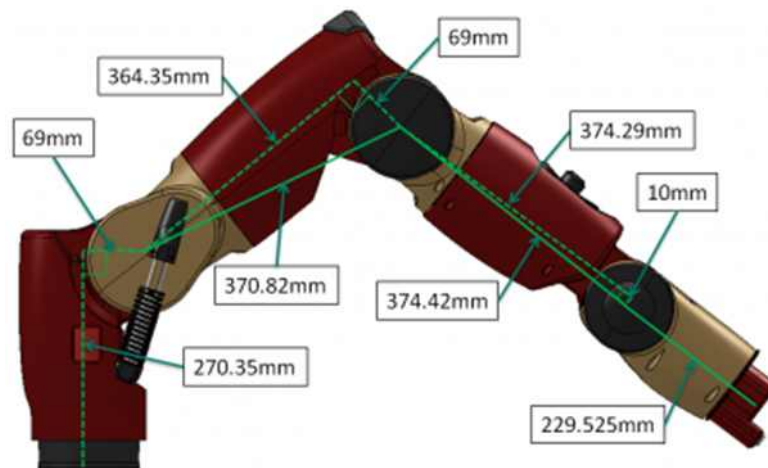
 - 센서: 초음파 센서, 3개의 카메라(헤드 디스플레이에 1개, 양쪽 그리퍼에 각 1개), 힘/토크 센서, 위치 센서
 - 모션 티칭/생성: ROS 기반 모션 프로그래밍 패키지 개발 또는 네비게이터(5번) 버튼 또는 손잡이(knob)
 - 상태 출력: 헤드 디스플레이



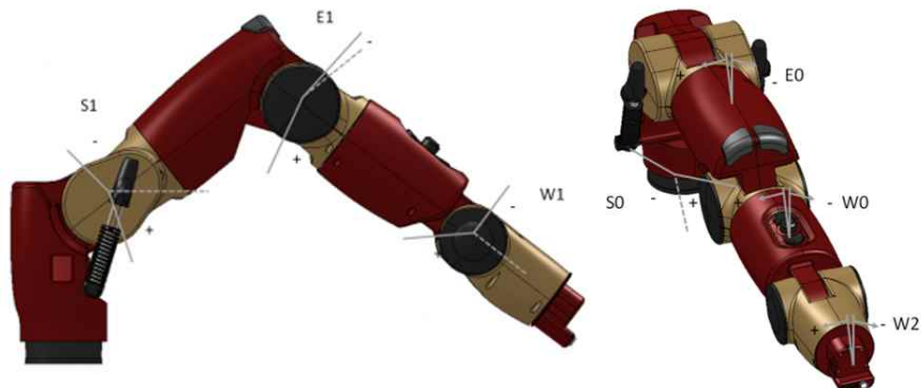
Baxter Hardware

1. Condition ring
2. Attention ring
3. Display
4. Work lights
5. Navigator
6. Lower front panel
7. Waist
8. Training cuff with parallel gripper
9. Training cuff with vacuum gripper
10. Pedestal

[그림 2-22] 백스터의 하드웨어 개요



[그림 2-23] 백스터 팔의 형상수치



[그림 2-24] 백스터 팔의 관절명

〈표 2-1〉 로봇 관절별 사양

관절명	최소각	최대각	범위	최대속도
S0(shoulder roll)	-97.50°	+97.50°	195.00°	114.6° /s
S1(shoulder pitch)	-123.0°	+60.0°	183.0°	114.6° /s
E0(elbow roll)	-174.99°	+174.99°	349.98°	114.6° /s
E1(elbow pitch)	-2.86°	+150.0°	152.86°	114.6° /s
W0(wrist roll)	-175.25°	+175.25°	350.50°	229.2° /s
W1(wrist pitch)	-90.0°	+120.0°	210.0°	229.2° /s
W2(wrist roll)	-175.25°	+175.25°	350.50°	229.2° /s

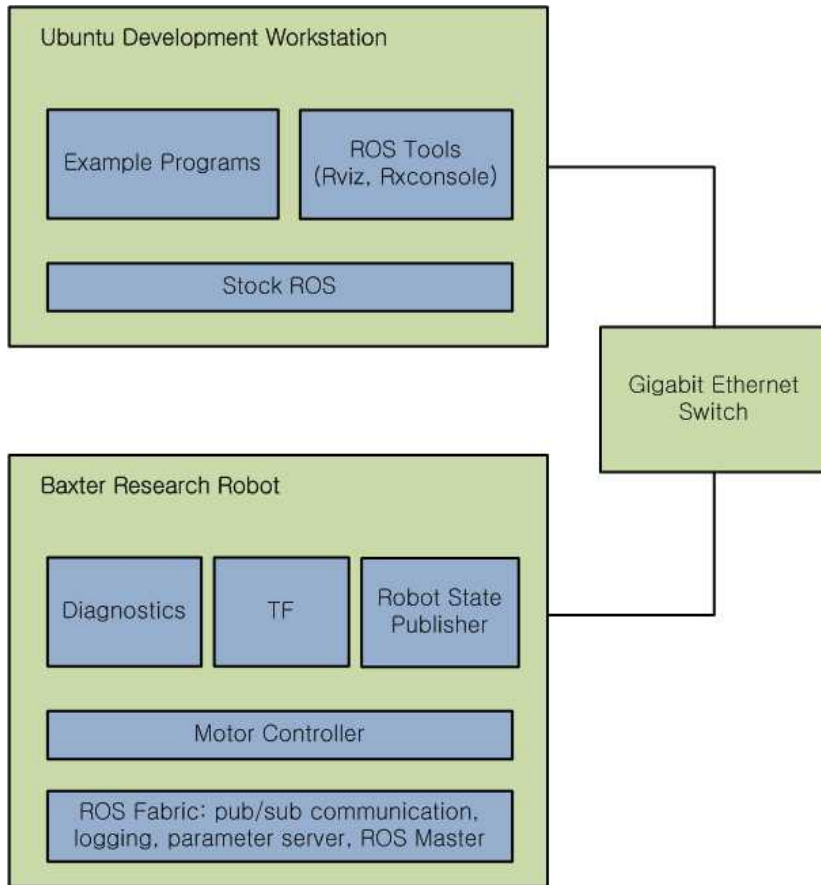
[그림 2-23]과 [그림 2-24]는 백스터 팔의 형상수치와 관절명을 나타내고, 〈표 2-1〉은 로봇의 관절별 회전각의 최소각, 최대각, 범위, 그리고 각속도 관련 정보를 나타낸다. 〈표 2-2〉는 백스터의 작업성능사양을 나타낸다.

〈표 2-2〉 백스터 작업성능 사양

항목	성능 사양
스크린 해상도	1024 x 600 픽셀
위치 정확도	±5mm
최대 페이로드(end effector 포함)	2.2kg
파지력(gripping force) (max)	35N
초음파센서 범위	4~40cm

(2) 백스터 소프트웨어

[그림 2-25]는 백스터의 소프트웨어 구조를 보인다(자세한 자료는 [sdk.rethinkrobotics.com/wiki/Baxter_Research_Robot_Software_Developers_Kit_\(SDK\)](http://sdk.rethinkrobotics.com/wiki/Baxter_Research_Robot_Software_Developers_Kit_(SDK)) 사이트 참고). 백스터는 독립형(stand-alone) ROS 마스터(master)를 가지고 있으면서 어떤 개발 워크스테이션도 여기에 연결되어 SDK(software development kit)를 통해 다양한 작업지능 ROS 패키지를 개발할 수 있다.



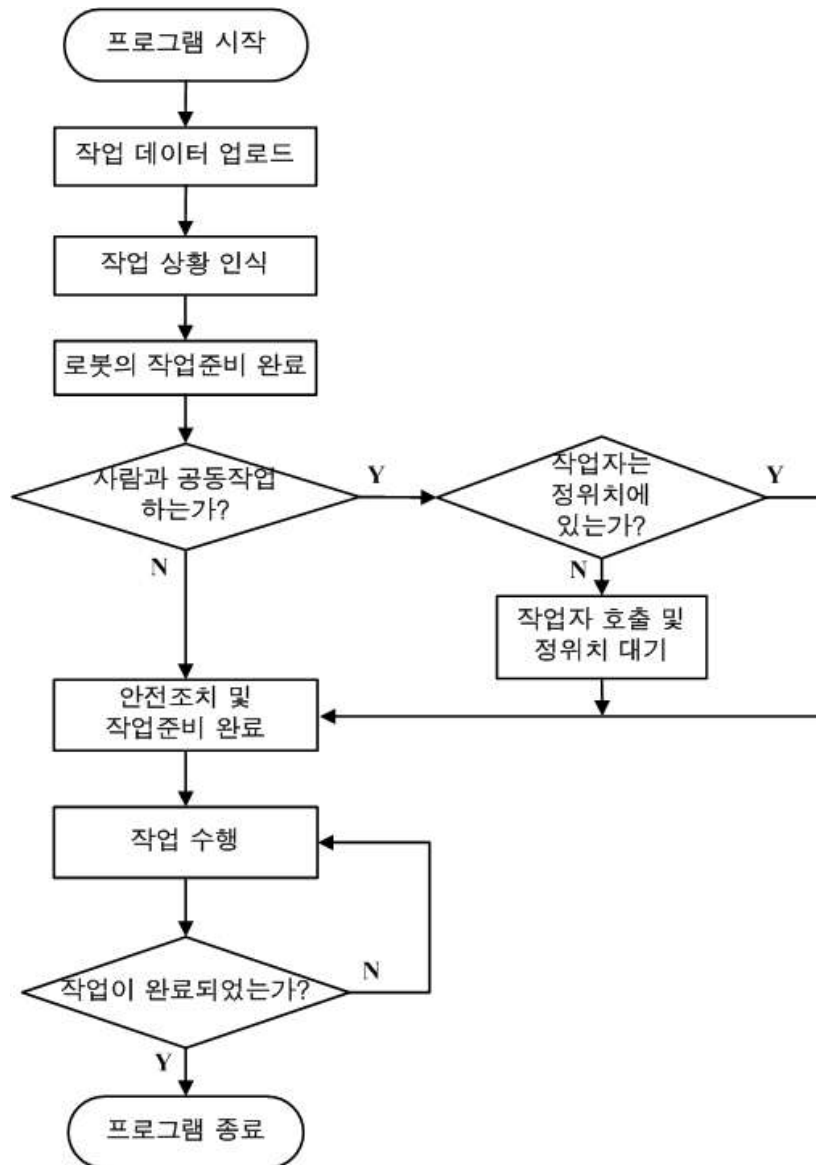
[그림 2-25] 백스터의 소프트웨어 개요

③ 로봇 작업지능 알고리즘을 기반으로 한 프로그램을 작성할 수 있다.

1. 작업지능 소프트웨어의 프로그램 구조를 설계한다.

작업지능 소프트웨어의 일반적인 프로그램 구조는 [그림 2-26]과 같다.

작업 데이터 업로드는 로봇 데이터베이스에서 작업에 필요한 액션 레시피와 액션 실행을 위한 목표 관절 궤적을 검색해서 로봇의 작업 메모리에 업로드한다. 만약, 다른 로봇 타입의 목표 관절 궤적이 데이터베이스에 있다면 RoboEarth의 ‘액션/상황의 인식 및 라벨링’ 기능과 유사한 방법을 사용하고 필요 시 최적화 알고리즘을 사용하여 현재 로봇에 맞게 수정한다.



[그림 2-26] 작업지능 소프트웨어의 프로그램 흐름도

2. 최적의 매니플레이션 궤적을 제작한다.

- (1) 현재 로봇 선진국들의 매니플레이션 자율화 지수는 5~6 정도 수준이며 스마트 팩토리(smart factory) 산업 분야에서 작업자와의 안전하고 조화로운 협업을 위해 최적의 매니플레이터 관절 궤적을 생성하는 알고리즘을 도출하는 연구가 활발히 진행되고 있다.
- (2) 식 (14)의 혼합다항함수법을 이용하여 특정 관절의 목표 회전각 궤적을 2개의 세그먼트로 구성하는 Matlab 코드를 작성한다.

```

function y = plot_two_seg_blend_poly(ti, tm, tf, thi, thm, thf, vi, vm, vf, st)

for i=1:floor((tf-ti)/st)+1
    t(i) = (i-1)*st + ti;
    if t(i) < tm    % 첫 번째 세그먼트
        t0 = ti;    t1 = tm;
        q0 = thi;    q1 = thm;
        v0 = vi;    v1 = vm;
    else           % 두 번째 세그먼트
        t0 = tm;    t1 = tf;
        q0 = thm;    q1 = thf;
        v0 = vm;    v1 = vf;
    end
    a0 = q0; a1 = v0;
    a2 = (3*(q1-q0)-(2*v0+v1)*(t1-t0))/(t1-t0)^2;
    a3 = (2*(q0-q1)+(v0+v1)*(t1-t0))/(t1-t0)^3;
    mth(i) = a0 + a1*(t(i)-t0) + a2*(t(i)-t0)^2 + a3*(t(i)-t0)^3;
    mth_dot(i) = a1 + 2*a2*(t(i)-t0) + 3*a3*(t(i)-t0)^2;
    mth_2dot(i) = 2*a2 + 6*a3*(t(i)-t0);
end
y = [mth' mth_dot' mth_2dot'];

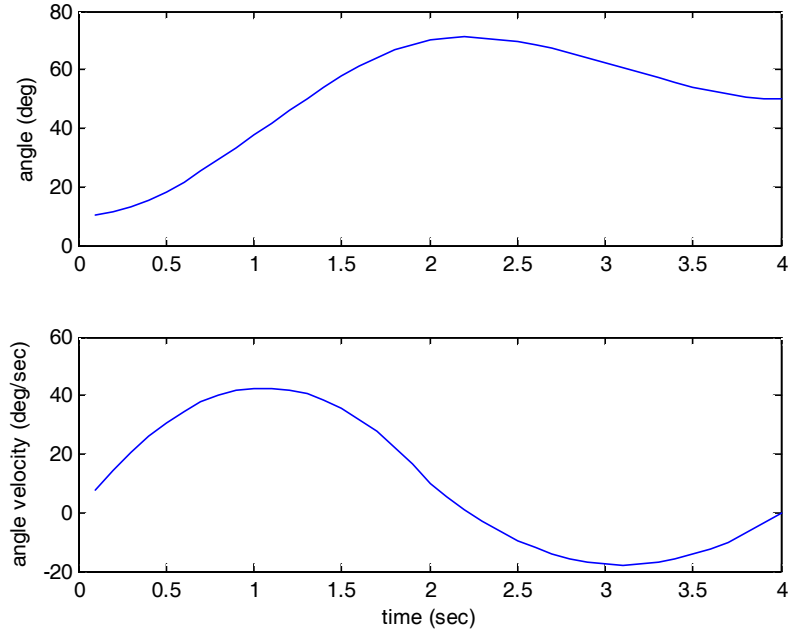
figure(1)
subplot(2,1,1)
plot(t, mth)
ylabel('angle (deg)');
subplot(2,1,2)
plot(t, mth_dot)
ylabel('angle velocity (deg/sec)');
xlabel('time (sec)');

```

목표 회전각 궤적의 경유점을 다음과 같이 설정하기 위한 Matlab 코드를 아래와 같이 작성하고 실행하면 [그림 2-27]과 같은 궤적을 얻을 수 있다.

$$\begin{aligned}
 t_0^1 &= 0, t_f^1 = t_0^2 = 2, t_f^2 = 4, \\
 q_0^1 &= 10^\circ, q_f^1 = q_0^2 = 70^\circ, q_f^2 = 50^\circ, \\
 \omega_0^1 &= 0, \omega_f^1 = \omega_0^2 = 10, \omega_f^2 = 0
 \end{aligned}
 \tag{19}$$

```
>> ti=0; tm=2; tf=4; thi=10; thm=70; thf=50; wi=0; wm=10; wf=0; st=0.1;
>> plot_two_seg_blend_poly(ti, tm, tf, thi, thm, thf, wi, wm, wf, st);
```



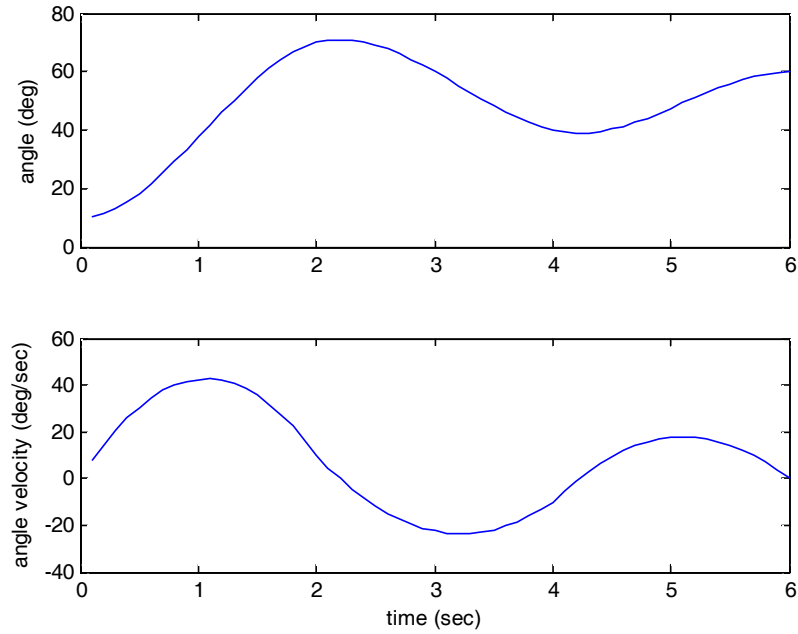
[그림 2-27] 2개의 세그먼트를 갖는 혼합다항식의 각도와 각속도 궤적

- (3) 식 (14)의 혼합다항함수법을 이용하여 특정 관절의 목표 회전각 궤적을 세 개의 세그먼트로 구성하는 Matlab 코드를 작성해 보자. plot_two_seg_blend_poly 함수에서 한 세그먼트를 더 추가하면 된다.

목표 회전각 궤적의 경유점을 다음과 같이 설정하고, 궤적을 생성하는 Matlab 코드를 아래와 같이 작성하고 실행하면 [그림 2-28]과 같은 궤적을 얻을 수 있다.

$$\begin{aligned} t_0^1 &= 0, t_f^1 = t_0^2 = 2, t_f^2 = t_0^3 = 4, t_f^3 = 6, \\ q_0^1 &= 10^\circ, q_f^1 = q_0^2 = 70^\circ, q_f^2 = q_0^3 = 40^\circ, q_f^3 = 60^\circ \\ \omega_0^1 &= 0, \omega_f^1 = \omega_0^2 = 10, \omega_f^2 = \omega_0^3 = -10, \omega_f^3 = 0 \end{aligned} \quad \text{식 (20)}$$

```
>> ti=0; tm1=2; tm2=4; tf=6; thi=10; thm1=70; thm2=40; thf=60;
    wi=0; wm1=10; wm2=-10; wf=0; st=0.1;
>> plot_three_seg_blend_poly(ti, tm1, tm2, tf, thi, thm1, thm2, thf, wi, wm1, wm2, wf, st);
```



[그림 2-28] 세 개의 세그먼트를 갖는 혼합다항식의 각도와 각속도 궤적

- (4) 매니퓰레이터가 장애물과 충돌하지 않으면서 목표 위치에 목표한 방향으로 말단부를 위치하도록 하기 위해서는, 몇 개의 세그먼트로 각 관절모터의 회전각 궤적을 만들지를 정하고 PSO 같은 최적화 알고리즘을 이용하여 요구 조건을 만족시키는 식 (19)와 식 (20)의 경유점 시간, 각도, 각속도 값의 최적값을 도출한다.

3. ROS의 모션 플래닝 패키지를 이용한다.

(1) RViz

RViz는 ROS의 3차원 시각화 도구로서 ROS 네트워크 상의 각종 데이터(거리측정센서의 장애물과의 거리 데이터, 3차원 거리센서의 포인트 클라우드 데이터, 카메라의 컬러 이미지 데이터 등)를 3차원 그래픽으로 보여주는 역할을 한다. 또 RViz 상에서 로봇 모델링 포맷인 URDF(Unified Robot Description Format)를 사용해서 3차원으로 표현된 로봇은 각 링크를 구동할 수 있어서 모션 시뮬레이션이나 실시간 제어에도 사용할 수 있다. [그림 2-29]는 6축 매니퓰레이터를 RViz와 연동하여 제어하는 모습을 보인다.



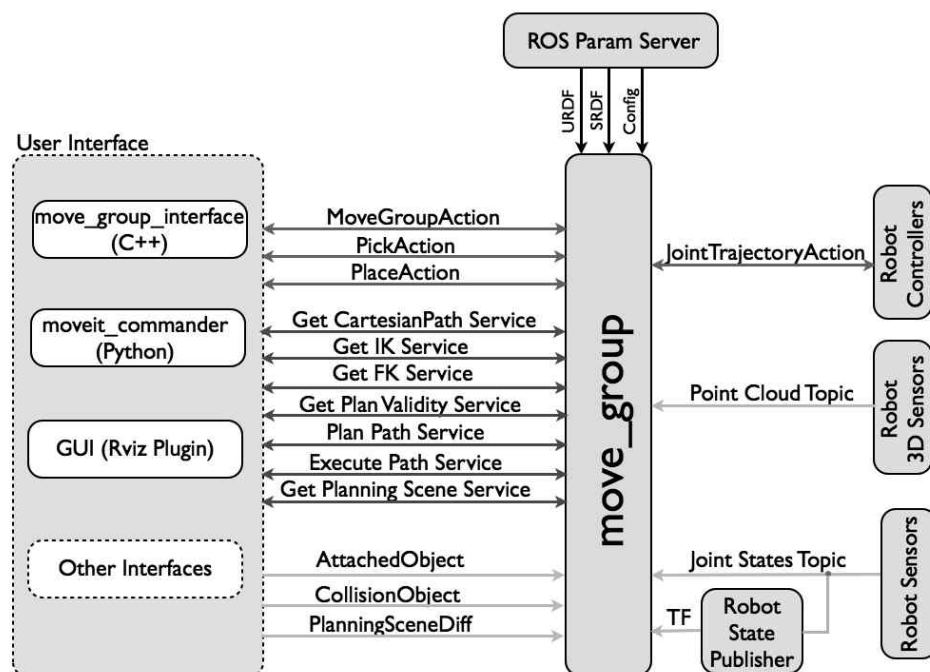
[그림 2-29] RViz를 이용한 6축 매니퓰레이터 실시간 제어

(2) MoveIt! 패키지

MoveIt! 패키지는 ROS에서 모바일 조작을 위한 툴들의 세트이다. 메인 웹 페이지 (<http://moveit.ros.org>)에는 잡기(grasp), 집어서 놓기(pick and place), 혹은 역기구학으로 간단한 모션 계획 등의 조작 태스크를 위해 MoveIt!을 사용하는 로봇 팔(또는 로봇)용 데모 예제와 함께 문서, 튜토리얼, 저장 교육법들이 담겨 있다.

이 라이브러리에는 빠른 역기구학 솔버(solver), 매니플레이션을 위한 최신 알고리즘, 3차원 지각(perception), 기구학, 제어, 그리고 네비게이션 기능 등이 있다. 백엔드 외에도 MoveIt!과 RViz 플러그인으로 새로운 로봇 팔을 구성하여 직관적인 방법으로 모션 계획 작업을 개발할 수 있고 사용하기가 쉬운 GUI를 제공한다.

[그림 2-30]은 MoveIt! 구조 다이어그램을 보인다.



[그림 2-30] MoveIt!의 소프트웨어 구조

MoveIt! 소프트웨어의 가운데 부분에 있는 move_group 성분은 표준 ROS 툴들과 YAML(YAML Ain't Markup Language), URDF, SDF(Simulation Description Format)와 같은 정의(definition) 언어를 사용하여 움직이는 동작을 하는 관절 그룹과 관련 요소들을 정의하는 역할을 한다. move_group은 다음의 기능을 수행한다.

- 모션 계획: move_group이 어떤 장애물과 충돌하지 않거나 관절 한계값을 위반하지 않으면서 목표자세에 도달하게 하도록 각 관절의 궤적 생성
- 계획 씬(scene): 로봇의 상태뿐만 아니라 로봇 주변의 공간 표현
- 환경 모니터: 점유격자지도와 octomap을 이용하여 환경과 물체 표현
- 기구학: 역기구학에 대해 연산적 자코비안-기반 솔버를 사용하는 플러그인 제공
- 충돌 확인: Flexible Collision Library 패키지를 사용하여 객체 충돌 여부 확인

ROS Kinetic Kame 버전을 우분투 16.04에 설치한 후 다음 명령을 실행하여 MoveIt!을 실행한다.

```
$ sudo apt-get install ros-kinetic-moveit-full
```

MoveIt!은 새로운 로봇 팔을 붙이기 위해 셋업 도우미라는 사용자 편의 그래픽 인터페이스를 제공한다. 셋업 도우미를 론치하기 위해, 터미널에서 다음 명령을 실행할 필요가 있다. 그러면 [그림 2-31]과 같은 셋업 도우미 창이 나타난다.

```
$ roslaunch moveit_setup_assistant setup_assistant.launch
```



[그림 2-31] MoveIt! 셋업 도우미의 초기 화면

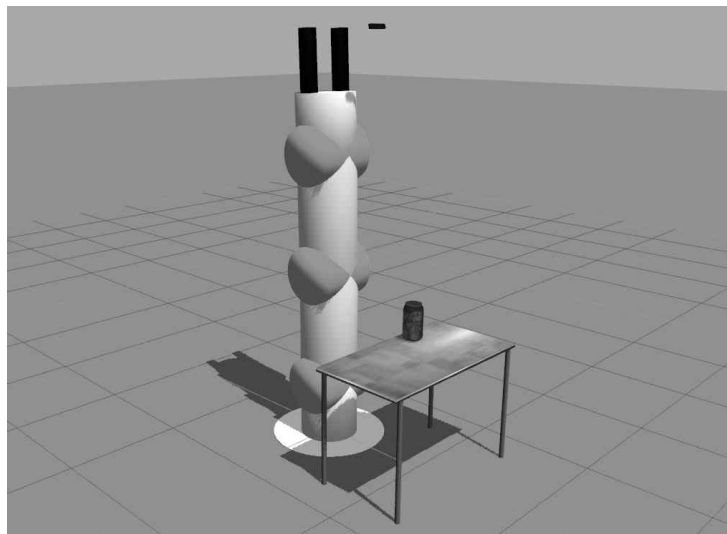
4. 대표적인 매니퓰레이터 작업인 픽-플레이스(pick and place) 태스크를 MoveIt!을 이용하여 시뮬레이션한다.

(1) 실제 로봇이 없는 경우를 대비해 가짜(fake) 제어기를 이용하여 태스크를 MoveIt! 데모 모드에서 실행한다.

가장 대표적인 매니퓰레이터 작업인 픽-플레이스(pick and place) 태스크는 목표 물체를 잡아서 집어 들고 목표 장소에 놓는 것으로 구성된다. 물체는 테이블처럼 평평한 표면 위에 놓여 있다고 가정하고 잡아야 할 물체는 상자로 근사화된 실린더로 모델링한다. 자세한 내용은 시중에 판매되고 있는 ROS 교재(김종욱, 2018)를 참고한다.

[그림 2-32]는 픽-플레이스의 계획 씬을 Gazebo로 표현한 것으로서 그립퍼와 RGB-D 센서가 있는 팔과 코카콜라 캔이 책상 위에 놓여 있다. 이 씬은 다음 명령으로 실행할 수 있다.

```
$ roslaunch rosbook_arm_gazebo rosbook_arm_grasping_world.launch
```



[그림 2-32] Gazebo에서 매니퓰레이터와 물체가 있는 환경

(2) 잡아야 할 목표 물체와 테이블에 관한 정보를 MoveIt!에게 알려줘야 한다. 목표 물체는 코카콜라 캔이므로 패키지의 scripts 폴더의 pick_and_place.py 코드에 다음과 같이 기본 프리미티브를 작성한다.

```
# Retrieve params:
self._grasp_object_name = rospy.get_param('~grasp_object_name', 'coke_can')
# Clean the scene:
self._scene.remove_world_object(self._grasp_object_name)
# Add table and Coke can objects to the planning scene:
self._pose_coke_can = self._add_coke_can(self._grasp_object_name)
```

다음의 _add_coke_can 메소드로 캔의 자세와 형상 규격을 정의함으로써 씬에 물체를

추가한다. 테이블에 대해서도 같은 과정을 반복한다.

```
def _add_coke_can(self, name):
    p = PoseStamped()
    p.header.frame_id = self._robot.get_planning_frame()
    p.header.stamp = rospy.Time.now()
    p.pose.position.x = 0.75 - 0.01
    p.pose.position.y = 0.25 - 0.01
    p.pose.position.z = 1.00 + (0.3 + 0.03) / 2.0
    q = quaternion_from_euler(0.0, 0.0, 0.0)
    p.pose.orientation = Quaternion(*q)
    self._scene.add_box(name, p, (0.15, 0.15, 0.3))
    return p.pose
```

- (3) 캔을 잡아서 들어올리기 위해서 먼저 잡기 이전/이후 자세를 만들어야 한다. 다음의 명령을 실행하여 moveit_simple_grasps 패키지에서 잡기 생성 서버를 사용하고 빌드한다.

```
$ wstool set moveit_simple_grasps --git https://github.com/davetcoleman/moveit_simple_grasps.git -v kinetic-devel
$ wstool up moveit_simple_grasps
```

- (4) 잡는 자세들을 정했으면 MoveIt!의 /pickup 액션 서버를 사용해서 목표값을 보낸다. 그 다음에 성공할 때까지 코크 캔을 최대한 많이 집는 것을 시도한다. _pickup 메소드 안에서 잡는 포즈를 생성한 직후에 MoveIt!을 위한 픽업 목표값을 생성한다. 목표값이 보내지고, 로봇 매니퓰레이터가 물체를 집어 올리는지 여부를 확인하기 위해 상태값이 사용된다. 픽업 목표값은 다음과 같은 _create_pickup_goal 메소드에서 생성된다.

```
def _create_pickup_goal(self, group, target, grasps):
    # Create goal:
    goal = PickupGoal()
    goal.group_name = group
    goal.target_name = target
    goal.possible_grasps.extend(grasps)
    # Configure goal planning options:
    goal.allowed_planning_time = 5.0
    goal.planning_options.planning_scene_diff.is_diff = True
    goal.planning_options.planning_scene_diff.robot_state.is_diff = True
    goal.planning_options.plan_only = False
    goal.planning_options.replan = True
    goal.planning_options.replan_attempts = 10
    return goal
```

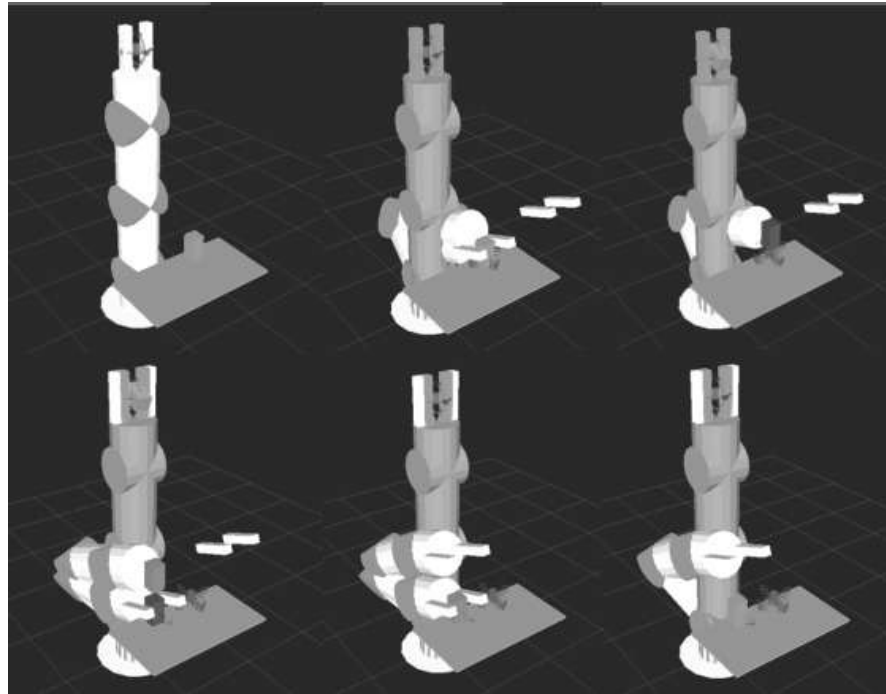
- (5) 물체를 집어 올린 다음에 매니플레이터는 적절한 위치에서 물체를 놓는 동작을 취해야 한다. MoveIt!은 /place 액션 서버를 제공하므로, 첫 번째 단계는 집어 올린 물체를 놓기 원하는 장소에 놓기 목표(place goal)를 보내는 액션 클라이언트를 생성한다. 그러면 최종적으로 목표점에 물체를 놓기 위해 여러 자세를 시도한다. _place 메소드는 아래의 코드를 사용한다.

```
def _place(self, group, target, place):
    # Obtain possible places:
    places = self._generate_places(place)
    # Create and send Place goal:
    goal = self._create_place_goal(group, target, places)
    state = self._place_ac.send_goal_and_wait(goal)
    if state != GoalStatus.SUCCEEDED:
        rospy.logerr('Place goal failed!: ' %
            self._place_ac.get_goal_status_text())
        return None
    result = self._place_ac.get_result()
    # Check for error:
    err = result.error_code.val
    if err != MoveItErrorCodes.SUCCESS:
        rospy.logwarn('Group %s cannot place target %s!: %s' % (group, target,
            str(moveit_error_dict[err])))
        return False
    return True
```

그 다음에 그 물체가 놓여졌는지 아닌지를 검사하기 위해 결과를 확인한다. 놓을 장소들을 얻었다면 _create_place_goal 메소드가 놓기 목표를 생성한다.

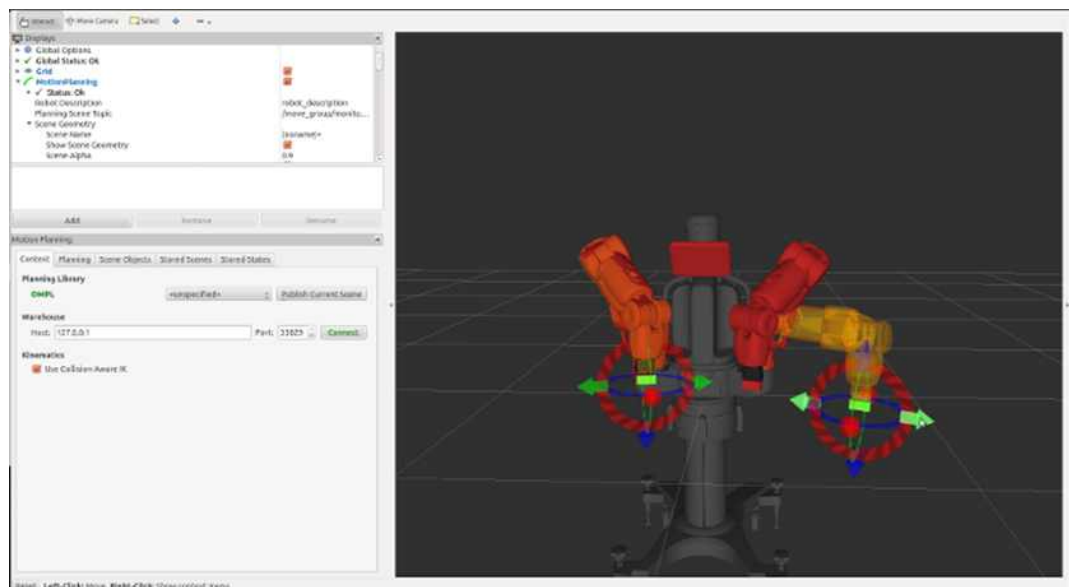
- (6) 데모 모드에서 (1)~(5)까지의 픽-플레이스 작업을 수행하기 위해 다음 명령을 실행한다. 그러면 그림 [2-33]과 같이 픽-플레이스 작업을 수행한다.

```
$ roslaunch rosbook_arm_moveit_config demo.launch
$ roslaunch rosbook_arm_pick_and_place grasp_generator_server.launch
$ rosrn rosbook_arm_pick_and_place pick_and_place.py
```



[그림 2-33] 데모 모드에서 픽-플레이스를 수행하는 매니플레이터

5. 백스터가 있으면 MoveIt!을 이용하여 양팔 작업을 시뮬레이션한다. 참고할 웹사이트 주소는 http://sdk.rethinkrobotics.com/wiki/MoveIt_Tutorial이다.



[그림 2-34] MoveIt!을 이용한 백스터 양팔 작업 시뮬레이션