K-MOOC 로봇공학과 인공지능 강의 정리.

1. 주차.

로봇 : actuated mechanism.

2개 이상 축 - ~~관절들로 구성~~ 으로 구성.

programmable, 의도된 작업을 수행.

autonomy 의 수준.

로봇 반응 과정 : sensing → thinking → 반응

센서 cpu actuator.

로봇은 actuator가 있어야 함.

로봇 분류
- 산업용 로봇 : 제한된 환경에서 동작 재프로그램, 속도가 빠름 ~~단순한~~ ~~환경은 인식할 수 없음~~ 범용성 높음
- 서비스 로봇 - 전문 서비스
  인간의 서비스를 대체함 | 일반 서비스
- 협동 로봇
  인간과 직접 접촉을 함. 안전장치 강화
- 인텔리전트 로봇
  주변 환경을 센서를 인식 (산업용 로봇은 사람과가 주변 환경을 지정)

AI 발전 사례

빅데이터가 필요 고성능의 CPU가 필요

빠른 반응속도 인공망

AlphaGo + 클라우드 + 5G.

AI 로봇 종류 — 휴머노이드 로봇
인간형태 인간과 인터페이스가 같음

— AI 로봇 : 외부환경을 인식,
스스로 상황을 판단

— 안드로이드 로봇
피부, 동작, 표현을 인간과 비슷하게
구현

로봇 은 단순작업 → 생산성 높음 (산업용 로봇)
생산성 ↑ ↓ 유연성 ↑
유연성 낮음 자동 높음

Intelligence 인간 은 복잡한, 다양한 작업 → 생산성 낮음
(지능) 유연성 높음 (AI 로봇)
자동 높음

로봇은 자유도가 높아질수록 가격이 높아짐
→ 특정, 목표 작업을 분석하여 최소한의 actuator,
end-effector를 구성해야 가격을 낮출 수 있음.

로봇의 종류 — 산업용 로봇
(용접, 도장 등)

— 개인 서비스 로봇
(청소,)

— 전문 서비스 로봇
(의료, 국방, 극한 환경 대응)

— 엔터테인먼트 로봇
(애완 로봇)

로봇 workspace에 따른 분류

- 카르테시안 (직교 좌표 시스템)

- 실린드리컬 직교 2개, 회전각 1개를 조합
  - 그 외 1개
- 스피리컬 : ~~변화율~~ ; 회전각 2개.
- SCARA : Selective Compliance Assembly Robot Arm → 수직방향으로 강성 높음
  빠른 속도, 4자유도 3개 회전 + 1개 수직
- Articulated Robot 인간 팔의 움직임 처리
  (마누플레이터, 6축 로봇) → 다목적에 대응
  6자유도

로봇 부품

① Manuplator : 링크 + 조인트

② Controllor : actuator를 제어

③ Actuator : 모터, Manuplator 안에 설치

④ End-Effector : 마지막 축에 설치. 그리퍼 등

⑤ Sensor : 외부 환경, 내부 상태 인식

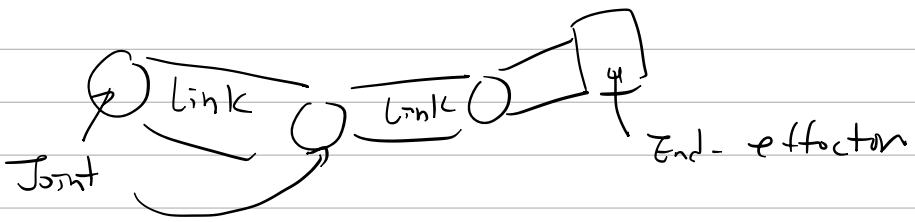
⑥ processor : Controllor를 제어하는 하드웨어
상황 인식

⑦ Software operating : 소프트웨어.

소프트웨어는 눈에 보이지 않으므로 개발속도가 느림. 궁극적으로는 AI가 목표.

산업용 로봇 기술과 서비스 로봇으로 이전.
다시 서비스 로봇 기술이 산업용 로봇으로 확대

로봇의 물리적 이해



자유도. 공간에서 (시스템) 안에서 움직이는 물체가 갖는 운동의 수.

강체 (Rigid Body)는 보통 6축 운동의 자유도

유체는 무한대의 자유도 (불과 수증)

Joint 종류

① Revolute 는 회전 ② prismatic : 직선

③ Helical 은 회전하면서 높이가 변하는 것

④ Cylindrical 은 회전과 돌아가 변할. 높이는 회전과 독립적으로 변함

⑤ Universal 은 직교한 2개축의 회전

DOF = 로봇 관절 자유도 합 - 관절로인한 구속조건 합

Grübler's Formular

$$DOF = m(N-1-J) + \sum_{i}^{J} f_i$$

m: 정의된 공간 차원의 수 2차원 은 3, 3차원은 6

N: 링크 수

J: Joint 수

f_i : i번째 관절 자유도

작업공간 (work space)

end effor 는 기준으로 Link, Joint 수에 따라 다름

강체의 위치 / 회전 표현방법

Rigid Motion — Translational Motion

— Rotational Motion

R_1^0 → 0 frame에서 1 frame로 회전 행렬

$R_1^0 = [\underbrace{X_1^0}_{\text{열벡터}} \quad Y_1 \quad Z_1]$

$= \begin{bmatrix} X_1 \cdot X_0 & X_1 \cdot X_0 & Z_1 \cdot X_0 \\ X_1 \cdot Y_0 & Y_1 \cdot Y_0 & Z_1 \cdot Y_0 \\ X_1 \cdot Z_0 & Y_1 \cdot Z_0 & Z_1 \cdot Z_0 \end{bmatrix}$

Z축 기준으로 θ 만큼 회전했다면

$R_1^0 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$Y = RX$

↳ 방향은 변하지만 크기는 그대로

Similarity Matrix

$B = T^{-1} A T$ 로 정의할 때

T대신 R_1^0 (0 frame 기준으로 1 frame을 회전)

대입

R_1^0를 알고 있는 matrix

A를 알고 있는 경우 B matrix?

B는 frame 1에서 바라볼 A matrix

$A \times T$ = A를 T만큼 frame 간 좌표계를 변환

$T^{-1} \times (A \times T)$ 변환된 좌표계를 다시 원위치.

frame 1에서 바라볼 A matrix = B

회전 행렬

회전

Worst frame → 회전행렬을 왼쪽에 곱함
(회전 기준을 생각 시킬때)

fixed frame → 회전행렬을 오른쪽에 곱함

$R_2^0 = R_1^0 R_2^1$

강체 위치 / 회전 표현 방법

O_1^0 → frame 0의 origin에서 frame 1의 origin vector

P. P^0 : frame 0 기준 P^1 : frame 1 기준

$P^0 = O_1^0 + P^1$ P^1은 frame 1 기준 좌표계를 frame 0 기준으로 변환

R_1^0 p^1 : p^1을 frame 0 기준으로 옮김

$p^0 = R_1^0 p^1 + O_1^0$ $\quad p_1 = R_2^1 p^2 + O_2^1$

$= R_2^0 p^2 + O_2^0$

$p^0 = R_1^0 R_2^1 p^2 + R_1^0 O_2^1 + O_1^0$

⇒ 두식 간편하게 정리할 matrix 사용.

$$\begin{bmatrix} p^0 \\ 1 \end{bmatrix} = \begin{bmatrix} R_1^0 R_2^1 & R_1^0 O_2^1 + O_1^0 \\ 0\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} p^2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} R_1^0 & O_1^0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2^1 & O_2^1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p^2 \\ 1 \end{bmatrix}$$

$[1\times3]\ [3\times3] = [1\times3]$

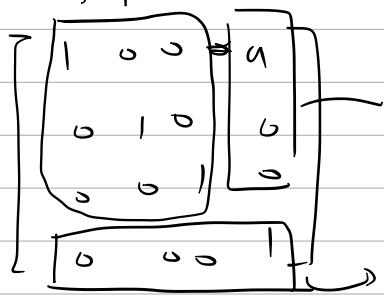
rotation matrix와 translation matrix를

하나로 → Homogeneous Transformation

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \quad H^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0 & 1 \end{bmatrix}$$

R_1^0 → 로봇의 원점

$Trans_{x,a}$ =



x축으로 a만큼 이동

계산의 편의를 위해 동차변환 행렬 사용

$H = H_1 H_2$ ~~곱~~ 곱한 순서 중요

연속적인 변환이 원래 좌표계에 대한 것인지 확인 중요

고정좌표를 기준으로 변환 진행 중요

역기구학은 그림이 없음.

로봇을 이루는 요소 effector actuator 등.
link, joint

로봇과 sensor

↳ 주변 환경 인지. 카메라, 초음파(mobile robot)

gyroscope sensor, accelerometer 가속도센서

최근에 여러 센서 정보를 취합하여 환경 상황을 인지.

여러 로봇형태의 지능을 묶어 → AI 로봇
인공지능

① Perception + ② motion planning

perception. 은 센서 해석으로 주변 환경을 이해

센서는 정확하게 상태를 측정할 수 없음

(오차. 노이즈. 물체기기 때문에 어려움)

주변 물체들이 움직일 수 있음

filter를 사용 state estimation

1) Bayes filter

로봇 위치, 노이즈, 상태를 반영하여

예측 → 보정의 확률으로 state를

estimation

2) Kalman filter

Bayes 필터의 한종류. 상태와 관측 모델이 선형이어야 함 → 속도가 빠름

하지만 사용시 전통적인 알고리즘 + deep learning 적용하여 정확도 높임

실시간 대응해야 하므로 perception이 비슷해 연산량 → computing power 필요 → deep learning 적용가능

② por motion planning

perception을 기반 → 움직임 수행 → 어떻게 수행 한 것인가? 경로 생성, 궤적은 움직임 수행 ※

로봇을 어떻게 움직일 것인가 → control

Joint 로 로봇에 전달, 바퀴는 회전

대부분 경로는 위치 기반으로 작성.

↳ 동역학 해석, 제어 알고리즘으로 빠르고 정확하게 이동

planning 과 perception 과의 차이 → error. 를 처리를

제어 ⇒ control 특징.

perception 센서에서 주변 환경을 측정을 업데이트

open loop control : perception X

closed loop control : pre perception 사용.

↳ 센서로 로봇에 대한 정보 · 바로로 반복적으로 주변 환경 업데이트 등작.

↳ human in the loop control :

조이스틱으로 사람이 로봇을 제어.

motion planning

시작에서 목표까지 장애물을 피해, 충돌없이 이동 그래프 이동

① Teaching By Demonstration

사람이 로봇을 직접 교시. 직관적.

매우 사소한 부분까지 teaching 한다면 로봇이 아님 → 어느정도 스스로 경로 설정 필요.

경로 또는 방법은 여러개일 → 어느 경로를 반드시는 로봇 알고리즘이 결정.

Path : 시간 정보가 없는 공간의 위치

Trajectory = 시간 + Path. 시간은 속도, 가속도

① point to point straight Trajectory

② Trajectories specified by multiple via points. (여러 점을 지나는 Trajectory)

p2p straight Trajectory : 직선보간 { Joint space / Task space

Configure 1 ——————→ Configure 2
어떻게?

Task Space 는 end effector 가 움직이는 (작업)공간
작업공간 이라. , task space 공간과 work space에
없을 수 있음.

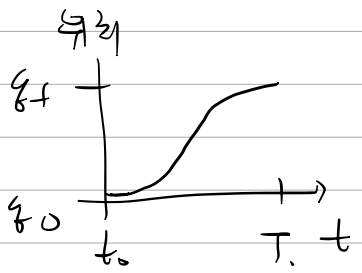
Joint Space 에서 직선궤적 (각 관절의 한계각을
위배 할 수 있음)
→ task space에는 곡선 포함 될 수 있음.

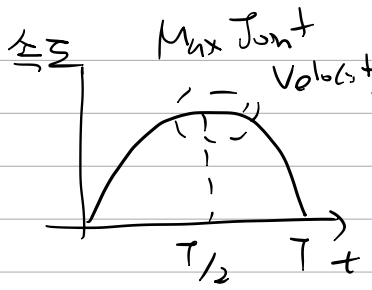
Trajectory 는 어떻게 만들 수 있을
다항식 (100 order) 으로 ~~경로를~~ 만들 수 있음
trajectory

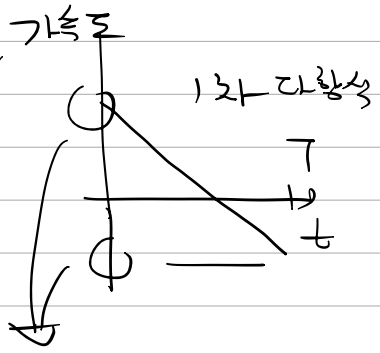
위치, 속도를 고려한다면 4개 방정식이
필요.

$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$ 간단한 3차식으로 근사
위치

* 3차 다항식은 가속도 시작, 끝위치가 0 이 된다
할







위치, 속도, 가속도 profile

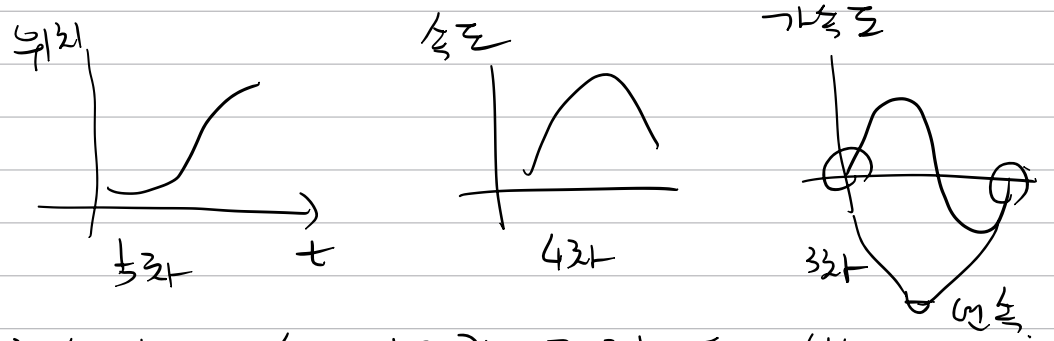
연속이 아님

의도치 않은 진동을 유발(3차 다항식 만족)

↳ 가속도 조건 추가 하여 3차를 5차 다항식

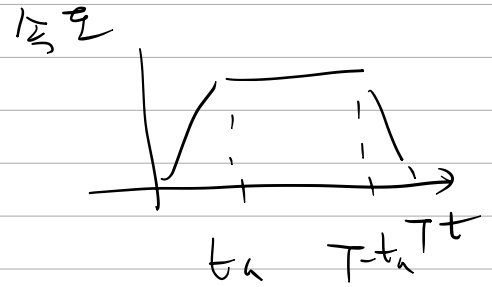
6개 구속조건 → 5차 다항식

- 5차 다항식 profile



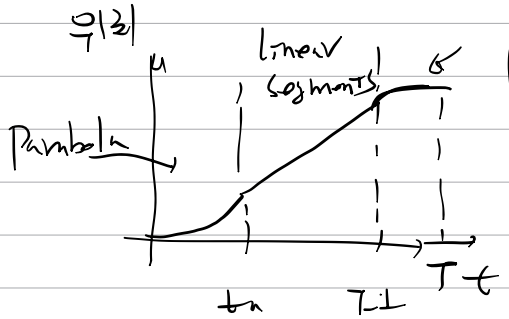
→ 차수가 높을수록 속도와 가속도 그래프는 부드럽게 (부드럽게 가감속)

- 사다리꼴 속도 profile = linear segments blends with parabolic



속도 증가 시간 = 감소시간

대칭



Linear segments with parabolic blend = 위치 profile

가속, 감속, 등속 구분없이 같은 , 계산이 쉽고, 프로그램이 간편함

시작 끝 부분에서 가속도가 큼 ⇒ 전동이 많을 것임 실제 시스템에서는 사용하지 않음.

시스템에 따른 관절 토크계산 순서.

$F = ma$ $\quad T = I_m \ddot{\theta} \quad \ddot{\theta}(t) = \frac{m}{I_m} a(t)$

가속도에 따라 관절 각도 변화 ⟶ 관절을 따라 계산해야

관절수 만큼 독립적 제어기 ⟵ 필요 수학적인 제어기

feed forward 제어기

모터의 feed forward 제어는 error가 나기 전에 미리 가속도의 토크를 주는 함수.

trajectories by multiple via points.

시작, 끝 위치, 속도, 가속도가 주어질 경우 3개 조건

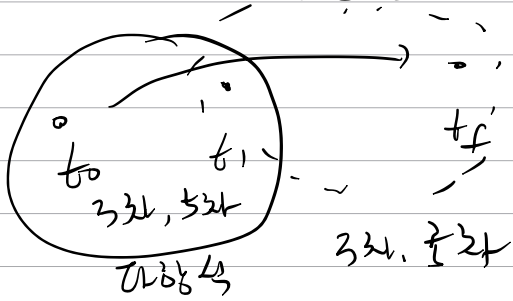
$q(t_0)$, $\dot{q}(t_0)$, $\ddot{q}(t_0)$, $q(t_f)$, $\dot{q}(t_f)$, $\ddot{q}(t_f)$

경계조건 → 6차 다항식으로 궤적 생성

정유정님 방식을 계산량 분량 → 낮은 차수

를 여러개 사용하여 연속적으로 통과

근접한 포인트를 통과.



각 구간의 속도과 주변 점과 연결하기 위한 3차 또는 5차

각 점들은 2 구간에서만 유효 (시간 위치?)

※

장애물이 있을때 어떻게 피해가면 될까?

motion planning + 장애물 고려

① Bug 알고리즘

로봇은 최나의 경로를 회전, 간단, 쉬울

점으로보고 장애물과 겹치는 영역을 C-space

로 고려해야 함

② Artificial Potential Function

목표는 중력을 받음 (장애물은 +, 목표는 -)

을 느끼듯이

③ Cell Decomposition

Graph를 경로탐색 Cell를 분해

A* 등 경로탐색 알고리즘 사용

④ Sampling-based 알고리즘

높은 DOF에 사용 RRT. ※

⑤ 버그 알고리즘

point로 인식할 수 있는 로봇 필요

장애물에 고정되어 있는 경우 센서로 (odometry 거리센서)

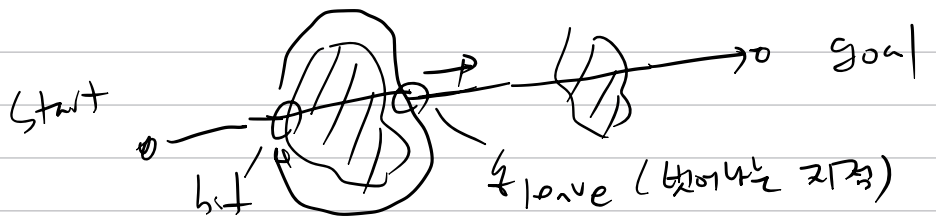
장애물 여부는 판단할 수 있어야 함

움직임은 그래프 있어야함 — 직선 (복잡X)
장애물은 한바퀴 돌아야 함

버그1, 버그2 등 여러 종류가 있음

원래는 센서가 있는 로봇을 봄 복잡: 경로 (사각→원) 또는 불가능.

obstacle 위치, 형태는 미리 알고 있지 않아도 됨.



Glove는 goal 에서 기준 가까운 점을 선택

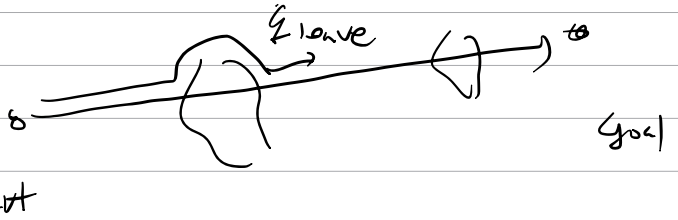
장애물을 들러싸여 있다면 bug 알고리즘으로

불가능

bug2 알고리즘

장애물을 벗어나는 지점이 다를 시작점, 끝점을

잇는 직선을 한다면 탈출



④ Configuration Space

물체이 회전하는 부피가 있을 → 포인트를 표현해줄

Configuration space를 만들어야 함

configuration = 시스템의 모든 점의 위치를 표현
(로봇)

C-space

점으로 로봇을 ~~표~~ 표현한 후 work space를

configuration space로 변환

③ Artificial Potential Function

work space를 산과 골짜기를 통해

goal 쪽은 위치 로봇은 공으로 놓으면 알아서 위치

Robot은 +전하. 장애물은 + 전하, 목표는 - 전하를 갖는

attractive potential 은 로봇과 목표 사이 거리에 비례하는 함수

repulsive potential 은 어느 범위 안에서만 대응.

충분히 먼 거리는 무시하면 된다는 의미

← 반발하는 힘은

threshold 사용.

gradient Descent Algorithm.

공간을 모두 해석 계산할 수 있음.

④ Road map

로봇은 움직이지만, 주변 환경은 변하지 않는다

map은 환경에 대한 정보 . mapping : 로봇 센서로

map을 만드는 과정

map을 생성하는 방법

- Topological representation

- Geometric Model

- Grid : 각 칸마다 나눠 로봇이 갈 수 있는지 확인

node와 edge를 통해 → graph로 인식

로드맵 : 이리저리 곡선들이 연결

start, goal 이 대표적 방법

* 접근성 (Accessibility)

로드맵에 정의된 임의 점에 start 가 접근가능한지?

- 돌발가능성 (Reparkability)

로드맵에 있는 goal 이 goal 에 접근가능? 갈 수 있는지

- Connectivity (연결성)

start 와 goal 이 연결되어 있는지?

그래 구현해 보자 해야 할

roadmap 을 위해 Visibility Graph.

장애물의 모서리를 node 로 정한

장애물과 통과하는 edge 는 제외.

→ reduced visibility Graph 를 만든

Voronoi Diagram