자율주행 자동차 공학

1. Introduction

1. self Driving specialization : · Localization and mapping
· Static and dynamic object detection
· Behaviour and maneuver planning
· Vehicle control

Ⅰ, System engineering for SDC (self driving car)

- Design a basic hardware system
- Identify main components of the autonomous driving software
- Create a safety assessment strategy for a self- driving car program

Ⅱ, State Estimation and Localization for SDC

- Understand the key methods for parameter and state estimation
- Develop and use models for typical vehicle localization sensors
- Apply Kalman filters to a vehicle state estimation problem
- Register point clouds from LIDAR to 3D Maps

Ⅲ, Visual Perception for SDC

- Project 3D points onto the camera image plane
- Calibrate the pinhole camera model
- Apply feature detection algorithms for localization and mapping
- Develop and train neural networks for object detection and semantic segmentation

Ⅳ, Motion Planning for SDC

- Devise trajectory rollout motion planning
- Calculate the time to collision
- Define high-level vehicle behaviours
- Develop kinematically feasible paths through environments
- Compute velocity profiles
- Plan behaviours and execute maneuvers to navigate through the world
- Gain valuable experience in debugging, testing in a simulator, and performing driving tests

Ⅴ, Vehicle control for SDC

- Design a feedback controller to track a reference trajectory (speed, steering, path-following, collision-avoidance, ...)
- Develop fault-tolerant control and Validate & Verify the designed controller
- Gain valuable experience in debugging, testing in a simulator, and performing driving tests

cf, ADAS

· 레벨 2 자율주행 기술 : both lateral or longitudinal control

※ 주요기술

1, 전방충돌방지 (FCA : Forward Collision-Avoidance Assist)

· 전방의 충돌위험 감지되면 경보를 울리고 운전자가 브레이크를 조작하지 않으면 브레이크를 자동으로 제어해 피할 수 있도록 도와주는 기술

· HMG는 2018년부터 출시되는 모델 신차부터 옵션이 아닌 기본 항목으로 적용

· 학습기반 인식 기술

· 인식 가능 범위 : 보행자 ⟶ 자전거 탑승자

↑ 카메라 + 전방 레이더 : 전파를 이용하여 사물로의 거리를 측정, 탐지 거리가 길고 날씨영향 X

+ 동물 or 도로에 떨어진 박스 : 현재 연구중

· FCA-JT (Junction-Turning : 교차로 대형차) : 교차로 직진신호 맞은편에서 다가오는 직진차량의 충돌 위험이 감지되면 제어

기존의 FCA는 전방충돌만 감지했었고, FCA-JT는 교차로에서 급격한 차이다며 적용

이에 따라 다양한 자동차와 사람의 움직임에 대한 특징을 제어 시스템에 미리 입력해 둔다. 그리고 전방 카메라와 전방 레이더를 통해 들어온 정보와 비교해 브레이크 제어 여부를 빠르게 판단한다. 하지만 동물이나 바위, 박스 등 불특정 사물을 모두 제어 시스템에 입력하는 데에는 물리적인 한계가 있을 수밖에 없다. 이런 이유로 우선 가장 중요한 자동차와 보행자, 자전거 탑승자부터 인식되도록 FCA를 개발했다. 그러나 ADAS 기술이 계속 발전하고 있기 때문에 앞으로 장애물 인식 범위는 매우 넓어지게 될 것이다.

2 Taxonomy(분류) of Driving Automation

1, Basic definition

Ⅰ, Driving task : 1, perceiving the environment
2, Planning how to reach from A to B
3. Controlling vehicle

Ⅱ, What makes up : 1, Lateral steering
2, Longitudinal steering
3, Object and Event Detection and Response.
4, planning (long or short term)

2 level

Ⅰ, level 0 : regular vehicles, no auto

Ⅱ, 1 : Either of lateral or longitudinal control, but not both
ex, Adaptive Cruise Control, Lane keeping Assistance (LKA)

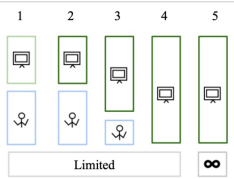
Ⅲ, 2 : Both "

Ⅳ, 3 : level 2 + automated object and event detection and response (OEDR)

Ⅴ, 4 : level 3 + Fallback (운행시작 대응), driver can entirely focus on other tasks.

Ⅵ, 5 : unlimited ODD

cf, Operational design domain (ODD) : operating system or feature thereof is specially designed to function = 운영설계영역



3. Requirements for Perception (perception)

Ⅰ, perception : we want to make sense of the environment and ourselves

identification, understanding motion → to inform our driving decisions

Ⅱ, Goals for perception

1, static objects

- Road and lane markings (on-road)
- Curbs (off-road)
- Traffic lights (off-road)
- Road signs (off-road)
- Construction signs, obstructions, and more (on-road)

2, dynamic objects : vehicles (2 or 4 wheels), Pedestrians

3, Ego localization (vehicle state estimation) : Position, velocity, acceleration, orientation, angular motion

4, Driving decisions and actions (Planning)

· reactive planning : immediative, rules that take int account

· Predictive planning : Make prediction about other vehicles and how they are moving.

5. Sensors and Computing hardware

Sensors	Main roles / functions	
Cameras (Vision)	Recognition of shape and color, such as road signs and white lines [Lane departure prevention, Parking assist, Surround view]	perceiving environment, enables depth estimation from image data
Ultrasonic waves (low cost)	Detection of objects at close distances [Parking Assist]	Short-range, all weather distance measurement
Short / Medium range radar	Detection of objects at close distances including relative speeds [Crossing vehicle warning, Rear collision warning]	Robust Object Detection and Relative speed estimation
Millimeter-wave radar	Detection of long-distance objects including relative velocity [ACC, LKAS, Simple pedestrian detection]	
LiDAR	Detection of object position and distance / Azimuth resolution: High [Pedestrian detection, Emergency braking, Collision avoidance]	detailed 3D scene geometry from LIDAR point cloud

· Global Navigation Satellite Systems (GNSS) : direct measure [position, velocity]

· IMU : Inertial mesurement : angular rotation rate, acceleration, heading

· Wheel Odometry : tracks wheel velocities and orientation

6. Computing Hardware

1, · Need a "self-driving brain"

· Image processing, Object detection, Mapping

GPUs - Graphic detection, Mapping

FPGAs - Field Programmable gate array

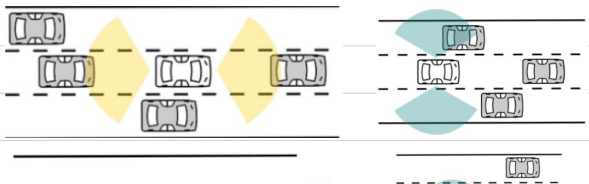
ASICs - Application Specific Integrated Chip

· Synchronization Hardware : to synchronize different modules and provide a common clock.

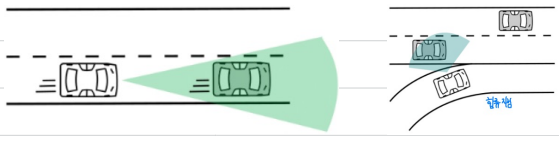
7. Hardware Configuration Design

1, Highway Analysis

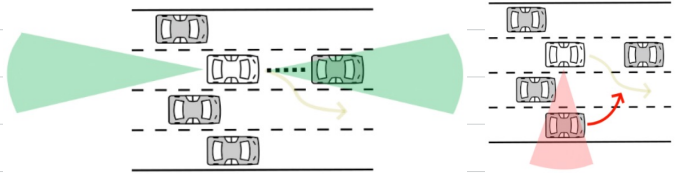
i, Emergency Stop :



ii, Maintain speed :



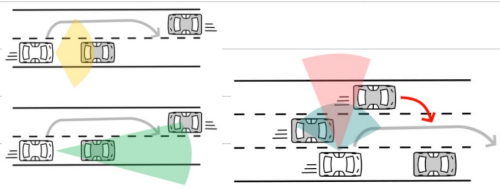
iii, Lane change :



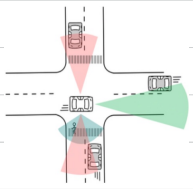
2. Urban Analysis

· Emergency Stop, Maintain speed, Lane change +

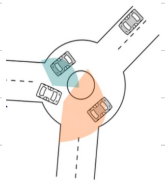
1, Overtaking



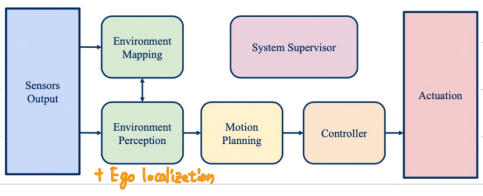
2. Intersections



3. roundabouts

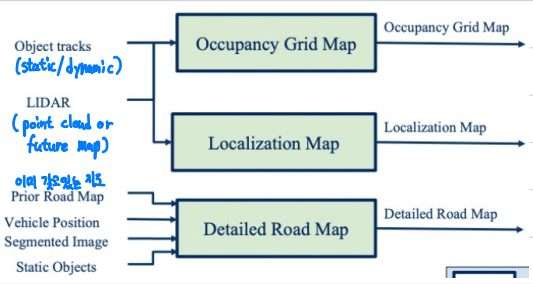


8. Software Architecture

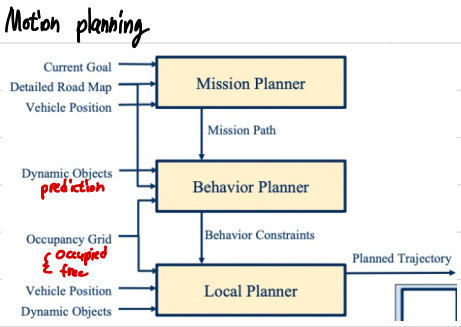


i, Environment Perception

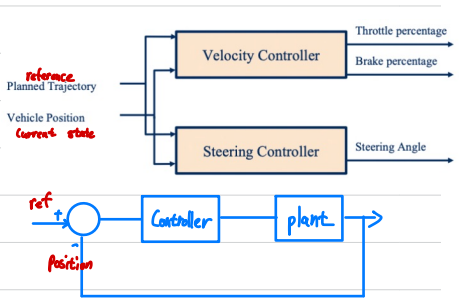
ii, Environmental Maps (주변 지도)



iii, Motion planning



iv, Vehicle Controller



/21171398 신건희

2. Vehicle Dynamic

1. Kinematic (운동학) modeling in 2D

- At low speeds, it is often suffiient to look only at kinematic model

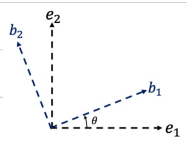
I, Coordinate frame

- Inertiel frame : fixed
- Body fixed frame : attached to vehicle, origin at vehicle center of gravity
- Sensor frame

II, Rotation matrixs

$$C_{EB} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$C_{BE} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

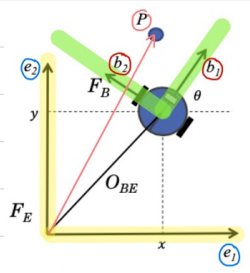


III, Coordinate transformation

E ↔ B done with a translation vector and a rotation matrix

$P_B = C_{BE}(\theta) P_E + O_{BE}$

$P_E = C_{EB}(\theta) P_B + O_{EB}$



- $P = \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \bar{P} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

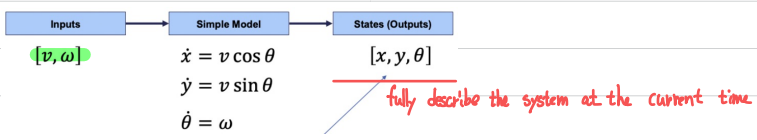
homogeneous coordinates (linear transformation)

$$\bar{P}_E = \begin{bmatrix} C_{EB}(\theta) & O_{EB} \\ 2\times2 & 2\times1 \end{bmatrix} \bar{P}_B$$

IV, 2D kinematic modeling

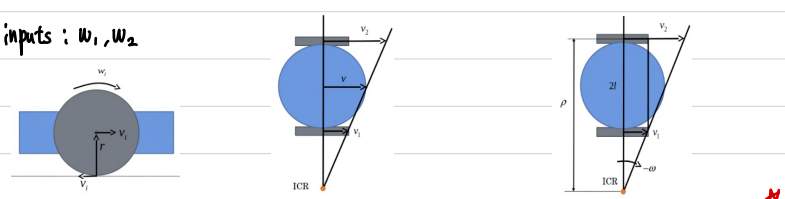
- Vehicle velocity always tangent to current path

$$\frac{dy}{dx} = \tan\theta = \frac{\sin\theta}{\cos\theta} \Rightarrow \dot{y}\cos\theta - \dot{x}\sin\theta = 0 \quad \begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \end{cases}$$



V, two-wheeled robot model

- inputs : w_1, w_2



1, $V_i = rw_i$ ($w_i \rightarrow V_i$)

2, $v = \frac{V_1+V_2}{2} = \frac{rw_1+rw_2}{2}$ ($V_i \rightarrow V$, Vehicle velocity)

3, $w = \frac{-V_2}{\rho} = \frac{-(V_2-V_1)}{2l} = \frac{rw_1-rw_2}{2l}$ ($V_i \rightarrow w$)

4, continous time model

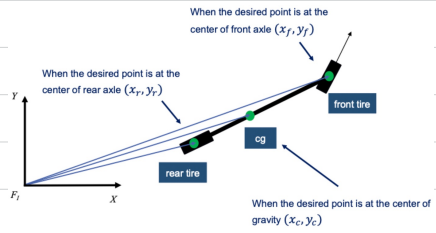
$$\dot{x} = v\cos\theta = \left(\frac{rw_1+rw_2}{2}\right)\cos\theta$$

$$\dot{y} = \left(\frac{rw_1+rw_2}{2}\right)\sin\theta$$

$$\dot{\theta} = \frac{rw_1-rw_2}{2L}$$

2. Kinematic bicycle model

- 2D bicycle model, front wheel steering

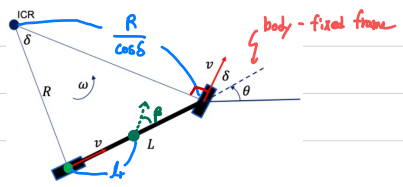


I, Rear wheel reference point

- apply instaneous center of rotation [ICR]

ICR 기준으로 회전 중심

$$\dot{\theta} = w = \frac{v}{R}, \quad \tan\delta = \frac{L}{R} \Rightarrow \dot{\theta} = w = \frac{v}{R} = \frac{v\tan\delta}{L}$$



- desired point : rear axle

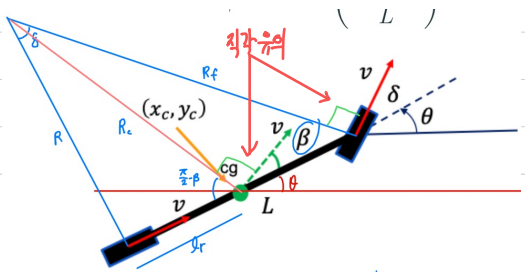
$\dot{x}_r = V\cos\theta$, $\dot{y}_r = V\sin\theta$, $\dot{\theta} = \frac{V\tan\delta}{L}$

- desired point : front axle

$\dot{x}_f = V\cos(\theta+\delta)$, $\dot{y}_f = V\sin(\theta+\delta)$, $\dot{\theta} = \frac{V}{\frac{R}{\cos\delta}} = \frac{V\sin\delta}{L}$

- desired point : center of gravity (cg)

$\dot{x}_c = V\cos(\theta+\beta)$, $\dot{y}_c = V\sin(\theta+\beta)$, $\dot{\theta} = \frac{V}{\frac{R}{\cos\beta}} = \frac{V\cos\beta\tan\delta}{L}$



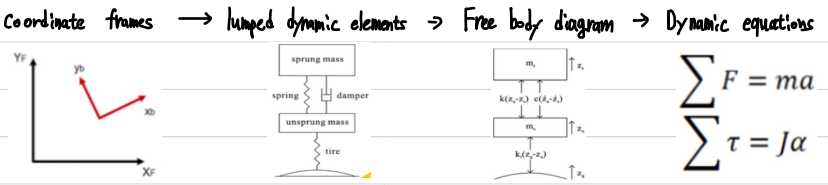
3. Basic Dynamic Modeling in 2D

- At higher speed and slippery roads, should consider slip condition

I, Dynamic modeling

- steps

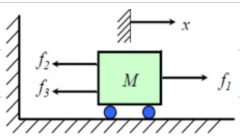
Coordinate frames ⟶ lumped dynamic elements → Free body diagram → Dynamic equations



$$\sum F = ma$$

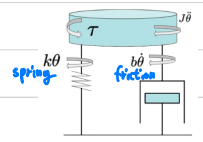
$$\sum \tau = J\alpha$$

- Translational (병진) system



$Ma = \sum F$

- rotational system



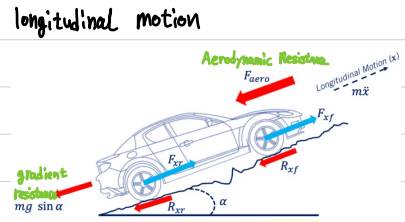
Inertia (관성) J

Torsional force : J, a load that is applied to a material through torque

$J\ddot{\theta} + b\dot{\theta} + k\theta = \Sigma\tau$

IV. Full vehicle modeling

I, longitudinal motion



$F_{load} = F_{aero} + R_x + mg\alpha$

$F_{aero} = \frac{1}{2}C_a\rho A\dot{x}^2 = c_a\dot{x}^2$ (상수값)

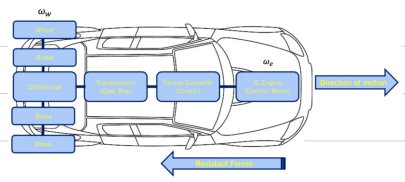
$R_x \approx C_{rr}$ 타이어

$m\ddot{x} = F_{xf} + F_{xr} - F_{aero} - R_{xf} - R_{xr} - mg\sin\alpha$

tire forces ⇒ F_x

rolling resistance = R_x

II, Power train modeling



- Rotational coupling : $W_w = GR w_t = GR w_e$
- Longitudinal velocity $\dot{x} = r_{eff} w_w$

ω_w : wheel angular speed
ω_t : turbine angular speed
ω_e : engine angular speed
GR : Combined gear ratios

- power flow

① Wheel (given)

$$I_w\dot{\omega}_w = T_{wheel} - \overline{r_{eff}F_x}$$

$$T_{wheel} = I_w\dot{\omega}_w + r_{eff}F_x$$

T_{wheel}

Transmission (w_T)

$$I_t\dot{\omega}_t = T_t - (GR)T_{wheel}$$

$$I_t\dot{\omega}_t = T_t - GR(I_w\dot{\omega}_w + r_{eff}F_x)$$

T_T

Torque Converter (w_e)

$$\omega_t = \omega_e$$

$$T_t = (I_t + I_wGR^2)\dot{\omega}_e + GRr_{eff}F_x$$

T_E

② Engine

$$I_e\dot{\omega}_e = T_{Engine} - T_t$$

$$I_e\dot{\omega}_e = T_{Engine} - (I_t + I_wGR^2)\dot{\omega}_e - GRr_{eff}F_x$$

$I = \Sigma mr^2$, $m\ddot{x} = F_x - F_{load}$

$F_x = m\ddot{x} + F_{load} = mr_{eff}GR\dot{w}_e + F_{Load}$: tire force

$\dot{x} = r_{eff}\cdot(GR)w_e$

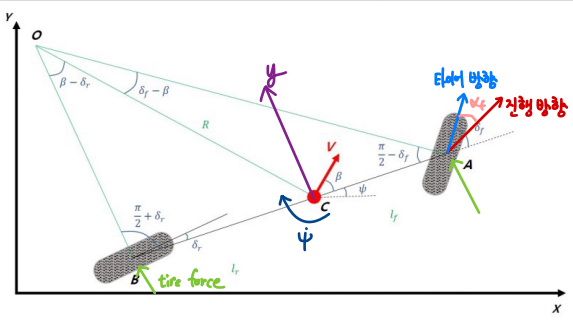
$$\{I_e + I_t + I_wGR^2 + m(GR^2)r_{eff}^2\}\dot{w}_e = T_{Engine} - (GR)r_{eff}F_{load}$$

J_e

$\therefore J_e\dot{w}_e = T_{Engine} - (GR)r_{eff}F_{load}$

total load torque

III, Lateral dynamic bicycle model



ψ : 방향각 (heading angle) ⟶ $\dot{\psi}$: yaw rate

δ_f : forward wheel steering angle

δ_r : Rear wheel steering angle

α_f : Front tire slip angle

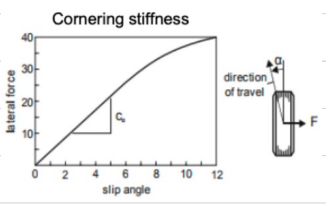
β : center of mass slip angle

1, Lateral acceleration : $a_y = \ddot{y} + w^2R = V\dot{\beta} + V\dot{\psi}$

2, $ma_y = mV(\dot{\beta}+\dot{\psi}) = F_{yf} + F_{yr}$

3, $I_z\ddot{\psi} = l_fF_{yf} - l_rF_{yr}$

$$\begin{cases} F_{yf} = C_f\alpha_f = C_f\left(\delta - \beta - \frac{l_f\dot{\psi}}{V}\right) \\ F_{yr} = C_r\alpha_r = C_r\left(-\beta + \frac{l_f\dot{\psi}}{V}\right) \end{cases}$$



4, Combination

$$\dot{\beta} = \frac{-(C_r + C_f)}{mV}\beta + \left(\frac{C_rl_r - C_fl_f}{mV^2} - 1\right)\dot{\psi} + \frac{C_f}{mV}\delta$$

$$\ddot{\psi} = \frac{C_rl_r - C_fl_f}{I_z}\beta - \frac{C_rl_r^2 + C_fl_f^2}{I_zV}\dot{\psi} + \frac{C_fl_f}{I_z}\delta$$

V. Summary

control point

lateral position

$$\dot{X}_{lat} = A_{lat}X_{lat} + B_{lat}\delta \qquad X_{lat} = [y\ \beta\ \psi\ \dot{\psi}]$$

side slip angle

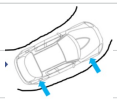
$$A_{lat} = \begin{bmatrix} 0 & V & V & 0 \\ 0 & -\frac{C_r+C_f}{mV} & 0 & \frac{C_rl_r - C_fl_f}{mV^2} - 1 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{C_rl_r - C_fl_f}{I_z} & 0 & -\frac{C_rl_r^2 + C_fl_f^2}{I_zV} \end{bmatrix} \quad B_{lat} = \begin{bmatrix} 0 \\ \frac{C_f}{mV} \\ 0 \\ \frac{C_fl_f}{I_z} \end{bmatrix}$$

5. Vehicle Actuation

I, Steering

- Steering angle → Wheel angle → Lateral forces →

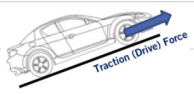
  δ_s ↑ steering system δ



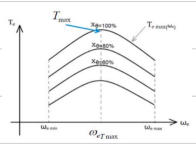
II, Throttling (Accelerating)

- Accelerator pedal position (Throttle) → Engine torque (power) → Transmission (gear ratios) → Wheel torque →

  γ_β ↑ power unit T_e, ω_e



- Typical torque curves for gasoline engines

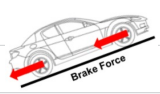


$T_{e\,max}(\omega_e) = A_0 + A_1\omega_e + A_2\omega_e^2$, $T_e(\omega_e, \gamma_\beta) = \gamma_\beta(A_0 + A_1\omega_e + A_2\omega_e^2)$

III, Braking (Decelerating)

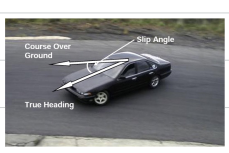
- Brake pressure → Braking disk force → Braking wheel torque →

  Δp T_{brake}



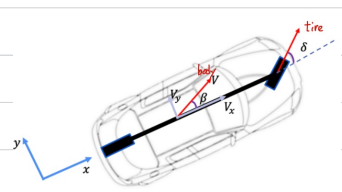
6. Tire slip & Modeling

- tire is the interface between vehicle and road

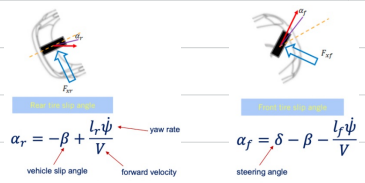


I, slip angle

slip angle → $\beta = \tan^{-1}\frac{V_y}{V_x} = \tan^{-1}\frac{\dot{y}}{\dot{x}}$, using small angle assumption $\beta \approx \frac{\dot{y}}{\dot{x}}$



tire slip angle → α : angle between the direction in which a wheel is pointing and the direction in which it is actually travelling



Slip ratios (Longitudinal slip)

$s = \frac{w_e r_e - V}{V}$

- $w_e r_e < V$: wheels are skiding (normal braking)
- $w_e r_e > V$: acceleration, in low friction driving
- $w_e r_e = V$: wheels are locked.

II, Tire modeling

Inputs to the tire model

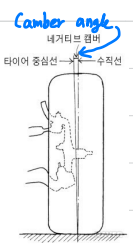
Tire Slip Angle
Slip Ratio
Normal Force
Friction Coefficient
Camber Angle
Tire properties

→ Tire Model →

Outputs of the tire model

Lateral Force
Longitudinal Force
Self-Aligning Moment
Rolling Resistance Moment
Overturning Moment

Camber angle
네거티브 캠버
타이어 중심선 — 수직선

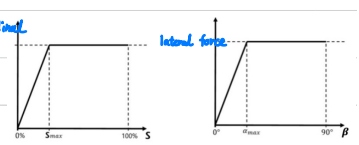


슬립각을 줄이는 방향으로 모멘트 발생

· linear tire model

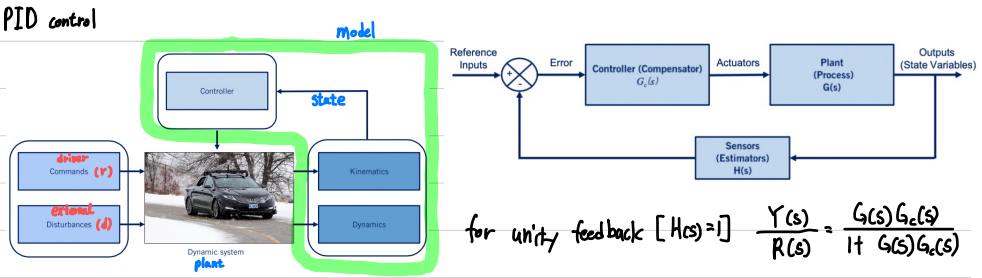
Assumption : relation between slip angle and force is linear

piecewise linear curve : $F(\alpha)$ ┌ $C\alpha$ $(\alpha \le \alpha_{max})$ / └ F_{max} $(\alpha \ge \alpha_{max})$

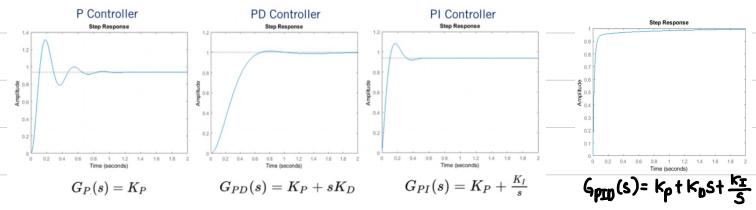


3. Vehicle Control for Autonomous Driving

1. PID control



for unity feedback [H(s)=1] $\frac{Y(s)}{R(s)} = \frac{G(s)G_c(s)}{1+G(s)G_c(s)}$

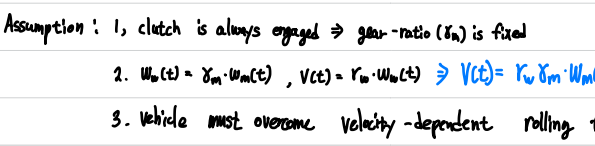


$G_P(s) = K_P$ $G_{PD}(s) = K_P + sK_D$ $G_{PI}(s) = K_P + \frac{K_I}{s}$ $G_{PID}(s) = K_P + K_D s + \frac{K_I}{s}$

2. Longitudinal Speed Control with PID

1, Cruise Control

Speed of the vehicle is controlled to kept at the reference speed.

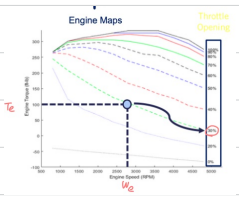


· High level controller : determine the desired acceleration based on reference and actual velocity

$$\ddot{x}_{des} = K_p(\dot{x}_{ref} - \dot{x}) + K_I\int_0^t(\dot{x}_{ref} - \dot{x})dt + K_D\frac{d(\dot{x}_{ref} - \dot{x})}{dt}$$

desired Acceleration — e — $\int e$ — $\frac{d}{dt}e$

· lower level controller : $\ddot{x}_{des}$ → Engine Torque → Throttle Angle



$$T_{Engine} = \frac{J_e}{(r_{eff})(GR)}\ddot{x}_{des} + T_{load}$$

3. Feedforward Speed Control

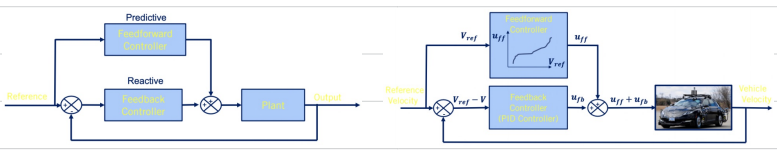
1, Combined FF and FB control

· FF provides predictive response, non-zero offset

· FB " correcting response, compensating for disturbance and errors in the model.

· Actuators (throttle angle) : FF → actuator signal u_{ff} based on the predefined table

FB → " velocity error



2, Cruise Control

i, Assumption : 1, clutch is always engaged ⇒ gear-ratio (γ_m) is fixed

2. $\omega_w(t) = \gamma_m \cdot \omega_m(t)$, $v(t) = r_w \cdot \omega_w(t)$ ⇒ $V(t) = r_w\gamma_m \cdot W_m(t)$

3. Vehicle must overcome velocity-dependent rolling friction

$$F_r(t) = (c_{r,0} + c_{r,1}\cdot v(t))\cdot m\cdot g, \quad \text{rolling friction}$$
$$F_a(t) = \tfrac{1}{2}\cdot\rho_a\cdot c_w\cdot A_f\cdot v^2(t), \text{ ar} \quad \text{aerodynamic force}$$
$$F_g(t) = m\cdot g\cdot\sin(\theta(t)). \quad \text{gravitational force}$$

4. All others forces are lumped into an unknown disturbance $F_d(t)$, assumed to zero

ii, Control systems

T_m (Motor torque) → [Vehicle Body] → v (Vehicle speed)

Can be assumed to be arbitrarily controllable within a certain range

iii, Dynamical system equations

① kinetic energy

effective mass $\left[m + \frac{4\theta_w}{r_w^2} + \frac{\theta_m}{\gamma_m^2 r_w^2}\right]$

$$E_{tot}(t) = \tfrac{1}{2}mv^2(t) + 4\cdot\tfrac{1}{2}\theta_w\omega_w^2(t) + \tfrac{1}{2}\theta_m\omega_m^2(t) = \tfrac{1}{2}M(\gamma_m, m)V^2(t) \quad (V(t) = r_w\gamma_m\cdot W_m(t))$$

② Energy conservation principle

$$\frac{d}{dt}E_{tot}(t) = P_m(t) - P_L(t) - P_b(t)$$

- $P_b(t) = F_b(t)V(t) \geq 0$: braking power
- $P_L(t) = (F_r(t) + F_a(t) + F_g(t) + F_d(t))V(t)$ — $F_d(t)$: disturbance
- $P_m(t) = \omega_m(t)T_m(t) = \frac{V(t)}{r_w\gamma_m}T_m(t)$ ≥ 0 traction mode, < 0 regeneration mode

② ODE describing system dynamics

$$M(\gamma_m, m)\frac{d}{dt}v(t) = \frac{1}{r_w\gamma_m}T_m(t) - (F_r(t) + F_a(t) + F_g(t) + F_d(t) + F_b(t))$$

assume to be 0 individually

nonlinear ⇒ equilibrium point 에서의 Jacobian linearization 사용

rolling friction force (≈ 0)

$$F_r(t) = mg(c_{r,0} + c_{r,1}v(t))$$
$$F_a(t) = \frac{1}{2}\rho_a c_w A_f v^2(t)$$
$$F_g(t) = mg\sin(\theta(t))$$

cf) Jacobian linearization around an equilibrium point

1, Jacobian linearization

Consider SISO nonlinear system, with an equilibrium point at $x=\bar{x}, u=\bar{u}$, satisfies $f(\bar{x},\bar{u}) = 0$

$\frac{d}{dt}x(t) = f(x(t), u(t))$, $y(t) = h(x(t), u(t))$ → local behavior of system around eq point →

$$\frac{d}{dt}x(t) = \frac{d}{dt}(\bar{x}+\delta x(t)) = f(\bar{x}+\delta x(t), \bar{u}+\delta u(t)) \approx f(\bar{x},\bar{u}) + \underbrace{\frac{\partial f}{\partial x}\Big|_{(\bar{x},\bar{u})}}_{A}\delta x(t) + \underbrace{\frac{\partial f}{\partial u}\Big|_{(\bar{x},\bar{u})}}_{B}\delta u(t)$$

$$\delta y := y - h(\bar{x},\bar{u}) = h(\bar{x}+\delta x(t), \bar{u}+\delta u(t)) - h(\bar{x},\bar{u}) \approx \underbrace{\frac{\partial h}{\partial x}\Big|_{(\bar{x},\bar{u})}}_{C}\delta x(t) + \underbrace{\frac{\partial h}{\partial u}\Big|_{(\bar{x},\bar{u})}}_{D}\delta u(t)$$

$$\therefore \frac{d}{dt}\delta x(t) = A\,\delta x(t) + B\,\delta u(t)$$
$$\delta y(t) = C\,\delta x(t) + D\,\delta u(t)$$

④
- $v(t)$: controlled state
- $\theta(t)$: uncontrollable state
- $T_m(t)$: control input
- $F_d(t)$: disturbance input

set-point $(\bar{v}, \bar{\theta}, \bar{T}_m, \bar{F}_d = 0)$ satisfies

$$\Rightarrow 0 = \frac{1}{r_w\gamma_m}\bar{T}_m - \left(mgc_r + \frac{1}{2}\rho_a c_w A_f\bar{V}^2 + mg\sin(\bar{\theta}) + \bar{F}_d\right)$$

define variables in terms of the set-points

$$v(t) = \bar{v} + \delta x(t),\ \theta(t) = \bar{\theta} + \delta\theta(t),$$
$$T_m(t) = \bar{T}_m + \delta u(t),\ F_d(t) = \bar{F}_d + \delta d(t)$$

Rewrite the dynamics as

$M(\gamma_m, m)\frac{d}{dt}\dot{x}(t) = f(x(t), \theta(t), u(t), d(t))$, linearize the dynamics around an equilibrium point

⇒

$$M(\gamma_m,m)\frac{d}{dt}\delta x(t) \approx \frac{\partial f}{\partial x}\Big|_{(\bar{x},\bar{\theta},\bar{u},\bar{d})}\delta x(t) + \frac{\partial f}{\partial\theta}\Big|_{(\bar{x},\bar{\theta},\bar{u},\bar{d})}\delta\theta(t) + \frac{\partial f}{\partial u}\Big|_{(\bar{x},\bar{\theta},\bar{u},\bar{d})}\delta u(t) + \frac{\partial f}{\partial d}\Big|_{(\bar{x},\bar{\theta},\bar{u},\bar{d})}\delta d(t)$$

정리

$$\tau(\gamma_m, m, \bar{v}) = \frac{M(\gamma_m, m)}{\rho_a c_w A_f\bar{v}}$$
$$\eta(m, \bar{v}, \bar{\theta}) = \frac{mg\cos(\bar{\theta})}{\rho_a c_w A_f\bar{v}}$$
$$\kappa(\gamma_m, \bar{v}) = \frac{1}{\gamma_m r_w\rho_a c_w A_f\bar{v}}$$
$$\alpha(\bar{v}) = \frac{1}{\rho_a c_w A_f\bar{v}}$$

⇒ $M(\gamma_m,m)\frac{d}{dt}\delta x(t) = -\rho_a c_w A_f\bar{v}\delta x(t) - mg\cos(\bar{\theta})\delta\theta(t) + \frac{1}{r_w\gamma_m}\delta u(t) - \delta d(t)$

⇒ $\tau(\gamma_m, m, \bar{v})\frac{d}{dt}\delta x(t) = -\delta x(t) - \eta(m,\bar{v},\bar{\theta})\delta\theta(t) + \kappa(\gamma_m,\bar{v})\delta u(t) - \alpha(\bar{v})\delta d(t)$

⑤ Define transfer functions

$$G_{x\theta}(s) = \frac{-\eta}{\tau s+1},\quad G_{xu}(s) = \frac{\kappa}{\tau s+1},\quad G_{xd}(s) = \frac{-\alpha}{\tau s+1}$$

iv, PI controller design

$C(s) = k_p + \frac{k_i}{s}$, closed-loop becomes

$$X(s) = G_{xu}(s)U(s) + G_{x\theta}(s)\theta(s) + G_{xd}(s)D(s) = -G_{xu}(s)C(s)X(s) + G_{x\theta}(s)\theta(s) + G_{xd}(s)D(s)$$

⇒ $$X(s) = -\frac{s}{\tau s^2 + (1+\kappa k_p)s + \kappa k_i}(\alpha\Theta(s) + \eta D(s))$$

arbitrary pole placement 가능

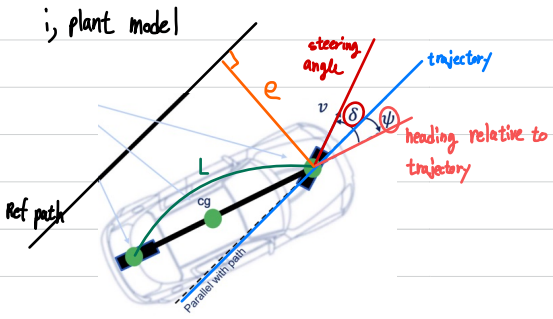
4. Lateral (측면운동) Control Design

I, Introduction

- define error relative to desired path
- goals : ┌ Heading path alignment $\psi = 0$
  └ Elimination of offset to path $e = 0$

II, Control Design : Geometric Controllers

i, plant model



- Controller error terms

① Heading error (ψ)

$$\dot{\psi}_{des}(t) - \dot{\psi}(t) = \frac{V_f(t)\sin\delta(t)}{L} \Rightarrow \dot{\psi}(t) = \frac{-V_f(t)\sin\delta(t)}{L}$$

(목표 상태)

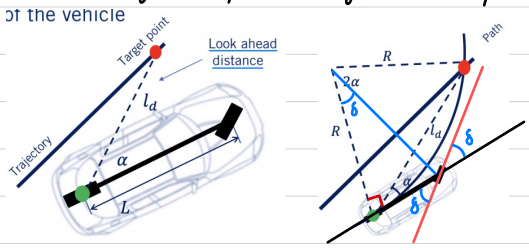
② Crosstrack error (e)

- distance from center of front axle to the closest point on path

$$\Delta e = V_f \sin(\delta - \psi) \Rightarrow \lim_{t\to 0} \frac{e(t+\Delta t) - e(t)}{\Delta t} = -\frac{\Delta e}{\Delta t} \Rightarrow \dot{e}(t) = V_f \sin(\psi - \delta)$$

ii, Control I : Pure pursuit

- consists of geometrically calculating the trajectory curvature (곡률) of the vehicle



l_d, α를 반영하여 heading error와 crosstrack error를 0으로 만드는 δ를 구하는 것

$$\frac{l_d}{\sin 2\alpha} = \frac{R}{\sin(\frac{\pi}{2} - \alpha)} \Rightarrow \frac{l_d}{2\sin\alpha\cos\alpha} = \frac{1}{\cos\alpha}$$

$$\frac{l_d}{\sin\alpha} = 2R \quad \therefore k\,(\text{curvature}) = \frac{1}{R} = \frac{2\sin\alpha}{l_d}$$

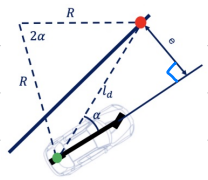
$$\tan\delta = \frac{L}{R} \Rightarrow \delta = \tan^{-1}\left(\frac{L}{R}\right)$$

$$\Rightarrow \delta = \tan^{-1}(kL) = \tan^{-1}\left(\frac{2L\sin\alpha}{l_d}\right) = f(l_d, \alpha)$$

l_d determined by target point location and angle between

α Vehicle heading direction (x_B) and lookahead direction.

- Crosstrack error (e)



$$\sin\alpha = \frac{e}{l_d} \Rightarrow k = \frac{2}{l_d^2}e$$

$$K = \frac{2\sin\alpha}{l_d}$$

- Lookahead l_d is assigned as a linear function of vehicle speed : $l_d = k_{dd} V_f$

$$\delta = \tan^{-1}(kL) = \tan^{-1}\left(\frac{2L\sin\alpha}{l_d}\right), \quad k = \frac{2}{l_d^2}e$$

$$\Rightarrow \delta\,(\text{기하학적 기하법칙}) = \tan^{-1}\left(\frac{2L\sin\alpha}{k_{dd} V_f}\right)$$

iii, Control II : Stanley Controller.

- look at both the error in heading and error in position relative to the closest point on the path
- combine three requirements $\psi \to 0$ $e \to e$

① steer to align heading with desired heading : $\delta(t) = \psi(t)$

② steer to eliminate crosstrack error

- Essentially proportional to error
- Inversely proportional to speed
- Limit effect for large errors with inverse tan
- Gain k determined experimentally

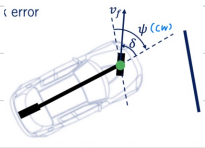
$$\delta(t) = \tan^{-1}\left(\frac{ke(t)}{V_f(t)}\right)$$

③ $\delta(t) \in [\delta_{min}, \delta_{max}]$

$$\therefore \delta(t) = \psi(t) + \tan^{-1}\left(\frac{ke(t)}{V_f(t)}\right), \quad \delta(t) \in [\delta_{min}, \delta_{max}]$$

- For large heading error, steer in opposite direction

first term ($\psi(t)$) is dominant



- For large positive crosstrack error ($e \gg 1$)

$$\tan^{-1}\left(\frac{ke(t)}{V_f(t)}\right) \approx \frac{\pi}{2} \to \delta(t) \approx \psi(t) + \frac{\pi}{2}$$

- error dynamics when not at maximum steering angle

$$\dot{e}(t) = -V_f(t)\sin(\psi(t) - \delta(t)) = -V_f(t)\sin\left(\tan^{-1}\left(\frac{ke(t)}{V_f(t)}\right)\right) = \frac{-ke(t)}{\sqrt{1 + \left(\frac{ke(t)}{V_f}\right)^2}}$$

and if, for small crosstrack error ($e(t) \approx 0$)

Linearization through taylor series, $\dot{e}(t) \approx -ke(t)$

4. Vehicle State Estimation for Autonomous

1. Filtering : Least-Squares to Kalman Filtering

1. Method of Least Squares

i, concept

- State estimation : x (unknown) ← state
  given measurement ← obtained from sensor
) ⇒ $y = h(x)$
- the least error criterion and how it is used in parameter estimation.

ii, equation (Ordinary Least Squares : assumes measurements are weighted equally)

i, x : real state
y : measurement
) $y = x + V$ (V : measurement noise)

	Measurement Model	Squared Error
when $x = [1]$ ⇒	$y_1 = x + v_1$	$e_1^2 = (y_1 - x)^2$
	$y_2 = x + v_2$	$e_2^2 = (y_2 - x)^2$
	$y_3 = x + v_3$ ⇒	$e_3^2 = (y_3 - x)^2$
	$y_4 = x + v_4$	$e_4^2 = (y_4 - x)^2$

ii, The squared error criterion : $\hat{x}_{LS} = \text{argmin}_x (e_1^2 + e_2^2 + e_3^2 + e_4^2)$

The 'best' estimate of resistance is the one that minimizes the sum of squared errors

⇒ square sum을 최소로 만드는 x

H : state의 계수

re-write our criterion using vectors : $e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = y - Hx = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} x$ (x unknown)

$$\mathcal{L}_{LS}(x) = e_1^2 + e_2^2 + e_3^2 + e_4^2 = \mathbf{e}^T\mathbf{e}$$
$$= (\mathbf{y} - \mathbf{H}x)^T(\mathbf{y} - \mathbf{H}x)$$
$$= \mathbf{y}^T\mathbf{y} - x^T\mathbf{H}^T\mathbf{y} - \mathbf{y}^T\mathbf{H}x + x^T\mathbf{H}^T\mathbf{H}x$$

"optimization problem"

iii. To minimize $\mathcal{L}(x)$, we can compute the partial derivative at $x = \hat{x}$, set to 0

$$\left.\frac{\partial \mathcal{L}}{\partial x}\right|_{x=\hat{x}} = 0 = -y^TH - y^TH + 2\hat{x}^TH^TH \Rightarrow (H^TH)\hat{x} = H^Ty \text{ : linear system}$$
$$\Rightarrow \hat{x}_{LS} = (H^TH)^{-1}H^Ty$$

2. Weighted Least Squares

i, concept

- the higher precision sensor, the more weighted in error
- the higher expected noise, the lower weight we place on the measurement

ii, equation

i, in Ordinary L.S, assumed that each noise term was equal variance, independent

$\Rightarrow E(v_i^2) = \sigma^2$, $R = E[VV^T] = \begin{bmatrix} \sigma^2 & & 0 \\ & \ddots & \\ 0 & & \sigma^2 \end{bmatrix}$

ii, If, we assume each noise term is independent, but of different variance

$\Rightarrow E(v_i^2) = \sigma_i^2$, $R = E[VV^T] = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_m^2 \end{bmatrix}$

then we can define a weighted least squares criterion as :

$\Rightarrow \mathcal{L}_{WLS}(x) = e^TR^{-1}e = \frac{e_1^2}{\sigma_1^2} + \frac{e_2^2}{\sigma_2^2} + \cdots + \frac{e_m^2}{\sigma_m^2} = (y - Hx)^TR^{-1}(y - Hx)$

We can minimize it as before

$$\hat{x} = \text{argmin}_x \mathcal{L}(x) \longrightarrow \left.\frac{\partial \mathcal{L}}{\partial x}\right|_{x=\hat{x}} = 0 = -y^TR^{-1}H + \hat{x}^TH^TR^{-1}H$$

$H^TR^{-1}H\hat{x}_{WLS} = H^TR^{-1}y$: normal eq

$\Rightarrow \hat{x} = (H^TR^{-1}H)^{-1}H^TR^{-1}y$

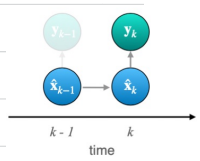
3. Recursive Least Square.

i, Concepts

- to compute a 'running estimate' of the least squares solution as measurements stream in.
- If we have a stream of data, do we need to re-solve our solution every time?

ii, Linear Recursive Estimator (추정기)

- $\hat{x}_{k-1}$: optimal estimate (best guess) of our unknown parameters ($x, y, \theta \cdots$) at time k-1
- y_k : new measurement at time k.
- $y_k = H_kx_k + V_k$ ⇒ to compute $\hat{x}_k$ (new estimate)



- linear recursive update : $x_k = x_{k-1} + K_k(y_k - H_k\hat{x}_{k-1})$

iii, Recursive Least squares

- to obtain gain Matrix K_k, we use a probabilistic formulation.

i, We wish to minimize the expected value of sum of squared errors

$$\mathcal{L}_{RLS} = E[(x_k - \hat{x}_k)^2] = \Delta_k^2$$

for n unknown parameters at time step k,

$$\mathcal{L}_{RLS} = E\left[\sum_{i=1}^{n}(x_{ik} - \hat{x}_{ik})^2\right] = E[(x_k - \hat{x}_k)^T(x_k - \hat{x}_k)]$$
$$= E[\text{Trace}((x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = \text{Trace}(P_k)$$

$\therefore P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]$: estimator covariance

$x_k = x_{k-1} + K_k(y_k - H_k\hat{x}_{k-1})$??

ii, using linear recursive formulation, we can express covariance as a function of K_k

$$P_k = (I - K_kH_k)P_{k-1}(I - K_kH_k)^T + K_kR_kK_k^T$$

P_k is minimized when,

$\mathcal{L}_{RLS} = \text{Trace}(P_k) \Rightarrow \nabla_x \mathcal{L}_{RLS} = 0$

$\Rightarrow K_k = P_{k-1}H_k^T(H_kP_{k-1}H_k^T + R_k)^{-1}$

$\Rightarrow P_k = P_{k-1} - K_kH_kP_{k-1}$
$= (I - K_kH_k)P_{k-1}$

P_{k-1} (given) → K_k → P_k ····

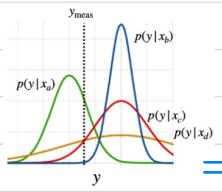
iii, Summary

1. Initialize the estimator
$$\hat{x}_0 = E[x]$$
$$P_0 = E[(x - \hat{x}_0)(x - \hat{x}_0)^T]$$
2. Set up the measurement model, defining the Jacobian and the measurement covariance matrix:
$$y_k = H_kx + v_k$$
3. Update the estimate of $\hat{x}_k$ and the covariance P_k using:
$$K_k = P_{k-1}H_k^T(H_kP_{k-1}H_k^T + R_k)^{-1}$$
$$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - H_k\hat{x}_{k-1})$$ (old estimate, new measurement)
$$P_k = (1 - K_kH_k)P_{k-1}$$

Importar
parameter c
'shrinks' w

4. Maximum likelihood

- We can ask which x makes our measurement most likely. or which x maximizes the conditional probability of y



MLE (Maximum likelihood estimator) $\hat{x} = \text{arg}_x\max\, p(y|x)$

$\Rightarrow \hat{x}_{MLE} = x_a$

i, Measurement (Sensor) Model

- $y = x + V$
- assuming noise V as some probably density function (정규분포)

$V \sim N(0, \sigma^2)$

- convert this into conditional probability (조건부확률)

$p(y|x) = N(x, \sigma^2)$: function of y, where x, σ^2 can be parameters

ii, Probability density function of a Gaussian ($Z \sim (\mu, \sigma^2)$) is

$$N(z; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(z-\mu)^2}{2\sigma^2}} \Rightarrow p(\underline{y}|x) \propto \mathcal{N}(y_1; x, \sigma^2)\mathcal{N}(y_2; x, \sigma^2) \times \cdots \times \mathcal{N}(y_m; x, \sigma^2)$$
$$z \in R \qquad z \to y = \frac{1}{\sqrt{(2\pi)^m\sigma^{2m}}}\exp\left(\frac{-\sum_{i=1}^{m}(y_i - x)^2}{2\sigma^2}\right)$$

$\hat{x}_{MLE} = \text{arg.max}_x\, p(y|x)$, given y

$= \text{arg.max}_x \log p(y|x) = -\frac{1}{2\sigma^2}((y_1 - x)^2 + \cdots + (y_m - x)^2) + C$

since, $\text{arg.max}_z f(z) = \text{arg.min}_z(-f(z))$ if we assume each y (measurement) has different variance

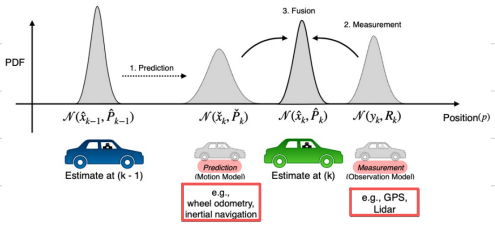
$$\therefore \hat{x}_{MLE} = \text{argmin}_x\, -(\log p(y|x))$$
$$= \text{argmin}_x \frac{1}{2\sigma^2}((y_1 - x)^2 + \ldots + (y_m - x)^2)$$
$$\hat{x}_{MLE} = \text{argmin}_x \frac{1}{2}\left(\frac{(y_1 - x)^2}{\sigma_1^2} + \ldots + \frac{(y_m - x)^2}{\sigma_m^2}\right)$$

Summary : $\hat{x}_{MLE} = \hat{x}_{LS} = \text{arg.min}_x \mathcal{L}_{LS}(x) = \text{argmin}_x \mathcal{L}_{MLE}(x)$

2. Filtering : From Least-Squares to Kalman Filtering

1. Kalman Filter

· state estimation for dynamic system



i, required model

current state, current input

· Motion model : $x_k = F_{k-1} x_{k-1} + G_{k-1} u_{k-1} + W_{k-1}$ (process noise or disturbance)

Measurement model : $y_k = H_k x_k + V_k$

· $V_k \sim N(0, R_k)$, $W_k \sim N(0, Q_k)$ — assumption by 실험 or sensor spec

ii, process

· RLS + motion model

$\hat{x}_{k-1}$ & u_{k-1}, $\hat{P}_{k-1}$ → P → C → $\hat{x}_k$, $\hat{P}_k$

i, Prediction

$$\check{x}_k = F_{k-1}\hat{x}_{k-1} + G_{k-1}u_{k-1}$$
$$\check{P}_k = F_{k-1}\hat{P}_{k-1}F_{k-1}^T + Q_{k-1}$$

Estimator covariance, 신뢰도 정의 척도 K_k

ii. optain Gain : $K_k = \check{P}_k H_k^T (H_k \check{P}_k H_k^T + R_k)^{-1}$

Correction : $\hat{x}_k = \check{x}_k + K_k(y_k - H_k\check{x}_k)$

$\hat{P}_k = (1 - K_k H_k)\check{P}_k$

RLS에서의 $\hat{x}_{k-1}$, P_{k-1}을 $\check{x}_k$, $\check{P}_k$로 대체

$\therefore x_k \sim (\hat{x}_k, \hat{P}_k)$

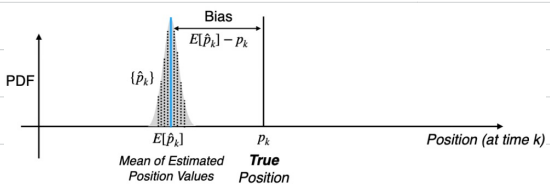
추정 불확실성

cf, $\check{x}_k$: Prediction (given motion model) at time k.

$\hat{x}_k$: Corrected prediction (given measurement) at time k.

2. Bias BLUEs (Best Linear unbiased Estimator)

i, Bias



This filter is unbiased if all k

$E[\hat{e}_k] = E[\hat{p}_k - p_k] = E[\hat{p}_k] - p_k = 0$

ii, Compute this analytically for kalman filter.

· Predicted state error : $\check{e}_k = \check{x}_k - x_k$

· Corrected estimate error : $\hat{e}_k = \hat{x}_k - x_k$

using kalman filter equations *

$$\check{e}_k = F_{k-1}\check{e}_{k-1} - w_k$$
$$\hat{e}_k = (1 - K_k H_k)\check{e}_k + K_k v_k$$

for the kalman filter, for all k

$$E[\check{e}_k] = E[F_{k-1}\check{e}_{k-1} - w_k] = F_{k-1}E[\check{e}_{k-1}] - E[w_k] = 0$$

$$E[\hat{e}_k] = E[(1 - K_k H_k)\check{e}_k + K_k v_k] = (1 - K_k H_k)E[\check{e}_k] + K_k E[v_k] = 0$$

Unbiased predictions, so long as $E[\hat{e}_0] = 0$, $E[V] = 0$, $E[W] = 0$

iii, Consistency in State Estimation

· $E[\hat{e}_k^2] = E[(\hat{p}_k - p_k)^2] = \hat{P}_k$ $\quad E[\check{e}_k \check{e}_k^T] = \check{P}_k$, $E[\hat{e}_k \hat{e}_k^T] = \hat{P}_k$ ⇒ consistent predictions

provided, $E[\hat{e}_0 \hat{e}_0^T] = \check{P}_0$ $\quad E[V] = 0$, $E[W] = 0$ + white noise

3. Extended kalman filter.

i, Concept

· linear system do not exist in reality

$x_k = f_{k-1}(x_{k-1}, u_{k-1}, W_{k-1})$ $\qquad y_k = h_k(x_k, V_k)$

current state / input ex) acceleration / process noise / measurement / state / state noise

ii, Linearization

· using a first order Taylor expansion

$$f(x) \approx f(a) + \underbrace{\left.\frac{\partial f(x)}{\partial x}\right|_{x=a}(x-a)}_{\text{First-order terms}} + \underbrace{\frac{1}{2!}\left.\frac{\partial^2 f(x)}{\partial x^2}\right|_{x=a}(x-a)^2 + \frac{1}{3!}\left.\frac{\partial^2 f(x)}{\partial x^2}\right|_{x=a}(x-a)^3 + \dots}_{\text{Higher-order terms}}$$

i, Linearized motion model

나는 원점값이므로 $u_{k-1} - \hat{u}_{k-1} = 0$

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \approx \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) + \underbrace{\left.\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}}}_{\mathbf{F}_{k-1}} (\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \underbrace{\left.\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{w}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}}}_{\mathbf{L}_{k-1}} \mathbf{w}_{k-1}$$

chosen operating point = nominal point

ii, Linearized measurement model

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k) \approx \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}) + \underbrace{\left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right|_{\check{\mathbf{x}}_k, \mathbf{0}}}_{\mathbf{H}_k} (\mathbf{x}_k - \check{\mathbf{x}}_k) + \underbrace{\left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{v}_k}\right|_{\check{\mathbf{x}}_k, \mathbf{0}}}_{\mathbf{M}_k} \mathbf{v}_k$$

F_{k-1}, L_{k-1}, H_k, M_k are called Jacobian matrics of the system.

iii, computing first-order partial derivatives of a vector-valued function.

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial}{\partial x_1} & \cdots & \frac{\partial}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

iii, putting it all together

i, Linearized motion model

$x_k = f_{k-1}(\hat{x}_{k-1}, u_{k-1}, 0) + F_{k-1}(x_{k-1} - \hat{x}_{k-1}) + L_{k-1}W_{k-1}$

ii, Linearized measurement model

$y_k = h_k(\check{x}_k, 0) + H_k(x_k - \check{x}_k) + M_k v_k$

iii, Prediction

$\check{x}_k = f_{k-1}(\hat{x}_{k-1}, u_{k-1}, 0)$

$\check{P}_k = F_{k-1}\hat{P}_{k-1}F_{k-1}^T + L_{k-1}Q_k L_{k-1}^T$

iv, Optimal gain

$K_k = \check{P}_k H_k^T (H_k \check{P}_k H_k^T + M_k R_k M_k^T)^{-1}$

loop : k 1 → 2, t 0 → 1

v, Correction

$\hat{x}_k = \check{x}_k + K_k(y_k - h_k(\check{x}_k, 0))$

$\hat{P}_k = (1 - K_k H_k)\check{P}_k$

$x_{init} \rightarrow \check{x}_1, \check{P}_1 \rightarrow \hat{x}_1, \hat{P}_1$

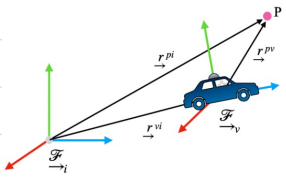
$l(1), r($

t=0 $\hat{x}_{k-1} = x_{init}$, $u_{k-1} = u(0)$ ⇒ $\check{x}_0, \check{P}_0$

3. Vehicle Pose Estimation

1. Coordinate transformation

i, Concept



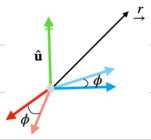
$\vec{r}^{pi} = \vec{r}^{vi} + \vec{r}^{pv} \longrightarrow r_i^{pi} = r_i^{vi} + C_{iv} r_v^{pv}$ (rotation matrix)

ii, Rotation Matrix

i, direction cosine matrix

$$C_{ba} = \begin{bmatrix} \vec{b}_1 \\ \vec{b}_2 \\ \vec{b}_3 \end{bmatrix} \begin{bmatrix} \vec{a}_1 & \vec{a}_2 & \vec{a}_3 \end{bmatrix} = \begin{bmatrix} \vec{b}_1 \cdot \vec{a}_1 & \vec{b}_1 \cdot \vec{a}_2 & \vec{b}_1 \cdot \vec{a}_3 \\ \vec{b}_2 \cdot \vec{a}_1 & \vec{b}_2 \cdot \vec{a}_2 & \vec{b}_2 \cdot \vec{a}_3 \\ \vec{b}_3 \cdot \vec{a}_1 & \vec{b}_3 \cdot \vec{a}_2 & \vec{b}_3 \cdot \vec{a}_3 \end{bmatrix}$$

ii, Unit quaternions

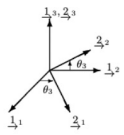


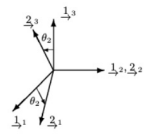
$q = \begin{bmatrix} q_w \\ q_v \end{bmatrix} = \begin{bmatrix} \cos\frac{\phi}{2} \\ \hat{u}\sin\frac{\phi}{2} \end{bmatrix}$ scalar / vector, $\|q\| = 1$

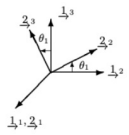
$$\mathbf{r}_b = \mathbf{C}(\mathbf{q}_{ba})\mathbf{r}_a$$
$$\mathbf{C}(\mathbf{q}) = (q_w^2 - \mathbf{q}_v^T\mathbf{q}_v)\mathbf{1} + 2\mathbf{q}_v\mathbf{q}_v^T + 2q_w[\mathbf{q}_v]_\times \quad \text{where} \quad [\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

iii, Euler angles

$C(\theta_3, \theta_2, \theta_1) = C_3(\theta_3)\, C_2(\theta_2)\, C_1(\theta_1)$







$$\mathbf{C}_3 = \begin{bmatrix} \cos\theta_3 & \sin\theta_3 & 0 \\ -\sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{C}_2 = \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \quad \mathbf{C}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ 0 & -\sin\theta_1 & \cos\theta_1 \end{bmatrix}$$

$\theta_3 \doteq \phi$ $\theta_2 \doteq \theta$ $\theta_1 \doteq \psi$

2. IMU

i, Concept

- Gyroscope : measure angular rotation rates about three separate axes.
- Accelerometers : measure accelerations along three orthogonal axes.

Gyroscope

measurement signal (or data)
$$\omega(t) = \omega_s(t) + \mathbf{b}_{\text{gyro}}(t) + \mathbf{n}_{\text{gyro}}(t)$$
($\omega_s(t)$: true)

$\omega_s(t)$	angular velocity of the sensor expressed in the sensor frame.
$\mathbf{b}_{\text{gyro}}(t)$	slowly evolving bias
$\mathbf{n}_{\text{gyro}}(t)$	noise term

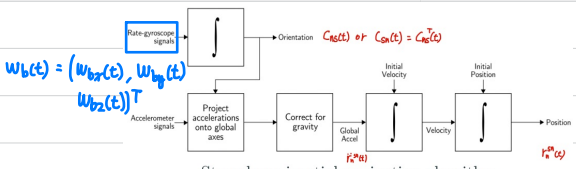
Accelerometer

$a_s(t)$: Measurement (sensor frame)
$$\mathbf{a}(t) = \mathbf{C}_{sn}(t)(\underbrace{\ddot{\mathbf{r}}_n^{sn}(t) - \mathbf{g}_n}_{a_n(t)}) + \mathbf{b}_{\text{accel}}(t) + \mathbf{n}_{\text{accel}}(t)$$

$\mathbf{C}_{sn}(t)$	orientation of the sensor (computed by integrating the rotational rates from the gyroscope) ↑ orientation estimation의 정확
$\mathbf{b}_{\text{accel}}(t)$	bias term
$\mathbf{n}_{\text{accel}}(t)$	noise term
$\mathbf{g}_n$	gravity in the navigation frame

for ignoring b_{accel}, n_{accel}
$a_n(t) = C_{ns}(t)\, a_s(t) = \ddot{r}_n^{sn}(t) - g_n$

ii, Strapdown Platform



$w_b(t) = (w_{bx}(t), w_{by}(t), w_{bz}(t))^T$, $C_{ns}(t)$ or $C_{sn}(t) = C_{ns}^T(t)$

Strapdown inertial navigation algorithm.

iii, Tracking Orientation C(t) from $w_b(t) = (w_{bx}(t), w_{by}(t), w_{bz}(t))^T$

- In order to specify orientation ($C_{ns}(t)$), we will use the direction cosines representation

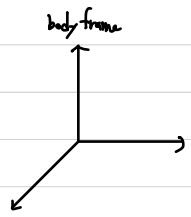
$\dot{C}(t)$ (rate of change of C) $= \lim_{\delta t \to 0} \frac{C(t+\delta t) - C(t)}{\delta t}$

$C(t+\delta t) = C(t)A(t)$
- $C(t+\delta t) = G_x(\phi+\delta\phi)\, C_y(\theta+\delta\theta)\, C_z(\psi+\delta\psi)$
- $C(t) = G_x(\phi)\, C_y(\theta)\, C_z(\psi)$
- $A(t)$: rotation matrix which relates the body frame at $t \to t+\delta t$

- $\delta\phi, \delta\theta, \delta\psi \approx 0$ ⇒ small angle approximation

$A(t) = I + \delta\Psi(t)$

$$\delta\Psi = \begin{pmatrix} 0 & -\delta\psi & \delta\theta \\ \delta\psi & 0 & -\delta\phi \\ -\delta\theta & \delta\phi & 0 \end{pmatrix} \xrightarrow{\lim_{\delta t \to 0} \frac{\delta\Psi}{\delta t}}$$



$$\dot{C}(t) = \lim_{\delta t \to 0} \frac{C(t+\delta t) - C(t)}{\delta t} = \lim_{\delta t \to 0} \frac{C(t)A(t) - C(t)}{\delta t} = \lim_{\delta t \to 0} \frac{C(t)(I + \delta\Psi) - C(t)}{\delta t} = C(t) \lim_{\delta t \to 0} \frac{\delta\Psi}{\delta t}$$

$$\lim_{\delta t \to 0} \frac{\delta\Psi}{\delta t} = \Omega(t) \qquad \Omega(t) = \begin{pmatrix} 0 & -\omega_{bz}(t) & \omega_{by}(t) \\ \omega_{bz}(t) & 0 & -\omega_{bx}(t) \\ -\omega_{by}(t) & \omega_{bx}(t) & 0 \end{pmatrix}$$

$$\dot{C}(t) = C(t)\Omega(t) \qquad C(t) = C(0) \cdot \exp\left(\int_0^t \Omega(t)\, dt\right)$$

⇒ $C(t+\delta t) = C(t) \cdot \exp\left(\int_t^{t+\delta t} \Omega(t) dt\right)$, $\int_t^{t+\delta(t)} \Omega(t) dt = B$

$$B = \begin{pmatrix} 0 & -\omega_{bz}\delta t & \omega_{by}\delta t \\ \omega_{bz}\delta t & 0 & -\omega_{bx}\delta t \\ -\omega_{by}\delta t & \omega_{bx}\delta t & 0 \end{pmatrix}$$

- Matrix exponential $\exp(B) = \sum_{k=0}^{\infty} \frac{B^k}{k!}$, where $B^0 = I$

$$\begin{aligned} C(t+\delta t) &= C(t)\left(I + B + \frac{B^2}{2!} + \frac{B^3}{3!} + \frac{B^4}{4!} + \dots\right) \\ &= C(t)\left(I + B + \frac{B^2}{2!} - \frac{\sigma^2 B}{3!} - \frac{\sigma^2 B^2}{4!} + \dots\right) \\ &= C(t)\left(I + \left(1 - \frac{\sigma^2}{3!} + \frac{\sigma^4}{5!} \dots\right) B + \left(\frac{1}{2!} - \frac{\sigma^2}{4!} + \frac{\sigma^4}{6!} \dots\right) B^2\right) \\ &= C(t)\left(I + \frac{\sin\sigma}{\sigma} B + \frac{1 - \cos\sigma}{\sigma^2} B^2\right) \quad \text{where } \sigma = \|w_b \delta t\| \end{aligned}$$

iv, Tracking Position

$a_b(t) = (a_{bx}(t), a_{by}(t), a_{bz}(t))^T$ ⇒ $a_g(t) = C(t) a_b(t)$

$$v_g(t) = v_g(0) + \int_0^t \underbrace{a_g(t) + g_g}_{\ddot{r}_g^{sg}}\, dt \quad \Rightarrow \quad V_g(t+\delta t) = V_g(t) + \delta t \cdot (a_g(t+\delta t) + g_g)$$

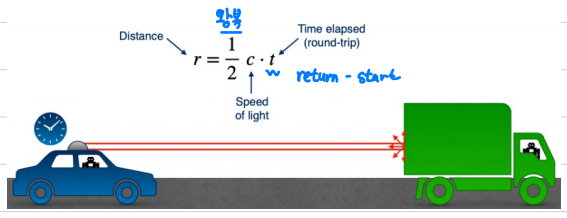
$$s_g(t) = s_g(0) + \int_0^t v_g(t)\, dt \quad \Rightarrow \quad S_g(t+\delta t) = S_g(t) + \delta t\, V_g(t+\delta t)$$

3. Lidar

i, concept

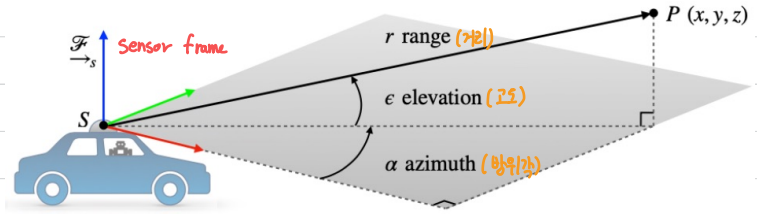
Data (a set point cloud) ⟶ Method (point cloud alignment) ⟶ Goal (Vehicle pose est by least squre)

ii, Measuring Distance with Time-of-Flight (ToF)



iii, Measurement Model

- report range, azimuth, elevation cf, model 입력에 noise 포함 . $V \sim N(0, R)$



iv, Inverse model (polar → cartesian)

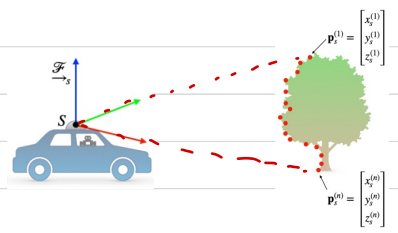
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{h}^{-1}(r, \alpha, \epsilon) = \begin{bmatrix} r\cos\alpha\cos\epsilon \\ r\sin\alpha\cos\epsilon \\ r\sin\epsilon \end{bmatrix}$$

v, Forwards Sensor Model (cartesian → polar)

$$\begin{bmatrix} r \\ \alpha \\ \epsilon \end{bmatrix} = \mathbf{h}(x, y, z) = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \tan^{-1}\left(\frac{y}{x}\right) \\ \sin^{-1}\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right) \end{bmatrix}$$

vi, Data Structures

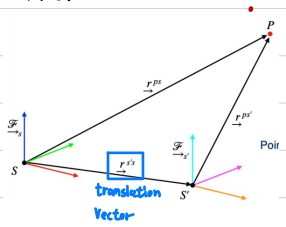
$$\mathbf{P}_s = \begin{bmatrix} \mathbf{p}_s^{(1)} & \mathbf{p}_s^{(2)} & \cdots & \mathbf{p}_s^{(n)} \end{bmatrix} = \begin{bmatrix} x_s^{(1)} & x_s^{(2)} & \cdots & x_s^{(n)} \\ y_s^{(1)} & y_s^{(2)} & \cdots & y_s^{(n)} \\ z_s^{(1)} & z_s^{(2)} & \cdots & z_s^{(n)} \end{bmatrix}$$



Ⅳ, Operations on Point Clouds

translation → Rotation → Scailing

i, traslation



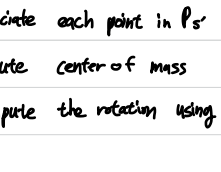
$\vec{r}^{Ps'} = \vec{r}^{Ps} - \vec{r}^{s's}$

$P_{s'}^{(j)} = P_s^{(j)} - r_s^{s's} = \vec{r}^{P(j)s'}$

$P_{s'} = P_s - R_s^{s's}$

$R_s^{s's} = [r_s^{s's}\ r_s^{s's} \cdots r_6^{s's}]$

ii, Rotation



$r_{s'} = C_{s's} r_s$

$P_{s'}^{(j)} = C_{s's} P_s^{(j)}$

$P_{s'} = C_{s's} P_s$

iii, Scaling

$r_{s'} = \underbrace{\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix}}_{S_{ss}} r_s$

$P_{s'}^{(j)} = S_{s's} P_s^{(j)}$

$P_{s'} = S_{s's} P_s$

iv, Putting all together

$P_{s'} = S_{s's} C_{s's} (P_s - R_s^{s's})$

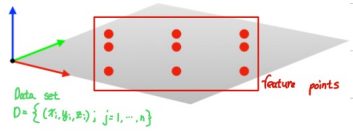
Ⅴ, Finding the road with 3D plane fitting

Equation of a plane in 3D:

$z = a + bx + cy$

model

Measurement $(x, y, z) \longrightarrow (a, b, c)$



error : $e_j = \hat{z}_j - z_j = (\hat{a} + \hat{b}x_j + \hat{c}y_j) - z_j \quad j = 1 \cdots n$

$$\Rightarrow \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}}_{e} = \underbrace{\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{x} - \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}}_{b}$$

- Minimize squared-error to get least-square solution for the parameters.

$\hat{x} = \arg\min_x \mathcal{L}_{LS}(x)$

$\mathcal{L}_{LS}(x) = e^T e = (Ax - b)^T (Ax - b) = x^T A^T A x - x^T A^T b - b^T A x + b^T b$

$\left.\frac{\partial \mathcal{L}_{LS}(x)}{\partial x}\right|_{x=\hat{x}} = 2A^T A\hat{x} - 2A^T b = 0 \longrightarrow A^T A\hat{x} = A^T b \Rightarrow \hat{x} = (A^T A)^{-1} A^T b$

1. Associate each point in $P_{s'}$ with the nearest point in P_s
2. Compute center of mass
3. Compute the rotation using SVD

SVD-Based alignment

ICP (iterative (반복적인) closest Point) : 반복적으로 근접한 점들을 퍼즐맞추듯 맞춘다.

SLAM

Frontend	backend
데이터 연결, 센서위치 파악	필터링 방법, Pose graph 최적화 방법

point set $x_n, y_n \Rightarrow \sum \|y_n - \bar{x}_n\|^2 p_n \rightarrow \min$

Means of point sets : $y_o = \frac{\sum y_n p_n}{\sum p_n}$, $x_o = \frac{\sum x_n p_n}{\sum p_n}$

Compute mean-reduced coordinates $a_n = y_n - y_o$, $b_n = x_n - x_o$

Covarance matrix : $H = \sum a_n b_n^T p_n$

SVD : $SVD(H) = UDV^T$

Rotation matrix : $R = UV^T$

Translation vector : $t = y_o - R x_o$

$\Rightarrow \bar{x}_n = Rx_n + t$ (알고 싶은 x)

prove

let, $y_o = \frac{\sum y_n p_n}{\sum p_n}$: Weighted mean of y_n

$\bar{x}_n - y_o = Rx_n + t - y_o$

$\bar{x}_n - y_o = R(x_n + R^T t - R^T y_o)$ x_o : new variable

$\bar{x}_n - y_o = R(x_n - x_o)$

'initial formulated problem' $\sum \|y_n - \bar{x}_n\|^2 p_n \rightarrow \min \Rightarrow \sum \|y_n - y_o - R(x_n - x_o)\|^2 p_n \rightarrow \min$

What we want? $R^*, x_o^* = \underset{R, x_o}{\arg\min} \sum$ ''

$\Phi(x_o, R) = \sum [(y_n - y_o) - R(x_n - x_o)]^T [(y_n - y_o) - R(x_n - x_o)] p_n$ 1. first derivative and goes to zero

$= \sum (y_n - y_o)^T (y_n - y_o) p_n + \sum (x_n - x_o)^T (x_n - x_o) p_n - 2\sum (y_n - y_o)^T R \cdot (x_n - x_o) p_n$

we need to find R that maximizes

$\frac{\partial \Phi(x_o, R)}{\partial x_o} = -2\sum (x_n - x_o) p_n + 2\sum R^T (y_n - y_o) p_n = 0$

$R^* = \underset{R}{\arg\max} \sum (y_n - y_o)^T R \cdot (x_n - x_o) p_n$

$\Rightarrow \sum (x_n - x_o) p_n = R^T \sum (y_n - y_o) p_n$

$= 0$ ∵ y_o is the weighted mean of y_n

$\Rightarrow \sum (x_n - x_o) p_n = 0 \Rightarrow \sum x_n p_n - \sum x_o p_n = 0$

$\therefore x_o = \frac{\sum x_n p_n}{\sum p_n}$: Weighted mean of x_n

Ⓥ 표시법

Π, $a_n = y_n - y_o$, $b_n = x_n - x_o \Rightarrow R^* = \underset{R}{\arg\max} \sum b_n^T R a_n p_n$

$= \underset{R}{\arg\max}\ tr(RH)$, $H = \sum (a_n b_n^T) p_n$

대각합

to find R that maximize $tr(RH) \Rightarrow$ SVD (singular value (특이값) decomposition)

$[m \times n] = [m \times m] \times [m \times n] \times [n \times n]$

$M \quad U \quad \Sigma \quad V^*$ (대각행렬)

Unitary matrix ($U^T U = I$)

$SVD(H) = UDV^T$, let, $R = VU^T$

$tr(RH) = tr(VU^T UDV^T) = tr(VDV^T)$

I diagonal

$tr(VD^{\frac{1}{2}} D^{\frac{1}{2}} V^T) = tr(VD^{\frac{1}{2}} (D^{\frac{1}{2}} V)^T)$, $VD^{\frac{1}{2}} = A$

∴ $tr(RH) = tr(AA^T)$

for every pos. $tr(AA^T) \geq tr(R'AA^T)$ for any rotation matrix R'

$\Rightarrow tr(RH) = tr(AA^T) \geq tr(R'AA^T) = tr(R'RH)$ (any other rotation matrix)

$\Rightarrow R = VU^T$ was optimal as it maximize the trace

4. Camera Basic

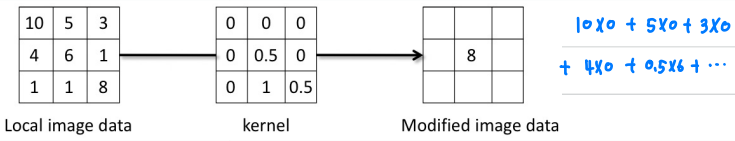
1. Image formation & Basic Filtering

I, filtering : Form a new image whose pixels are a combination of the originals pixels.

- to get useful information from images ; edge or contours HPF : 고주파(이미지 경계) 확인
- to enhance the image ; to remove noise etc.. LPF : 고주파 차단
- A key operator in CNN

II, Convolution

- linear filtering : replace each pixel by a linear combination (a weighted sum) of its neighbors.
- kernel : prescription for the linear combination



10x0 + 5x0 + 3x0 + 4x0 + 0.5x6 + ⋯

- Convolution (합성곱, $f * g$) : 두 함수 f, g 가운데 하나의 함수를 반전, 이동시킨 후 곱의 적분

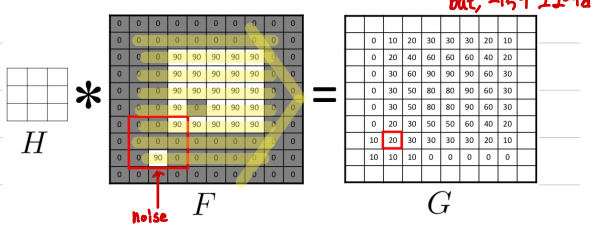
" kernel is flipped "

F = image
H = kernel [2k+1 x 2k+1]
G = Output

$$G[i,j] = \sum_{u=0}^{2k}\sum_{v=0}^{2k} H(u,v)F[i-u,j-v] = H * F$$

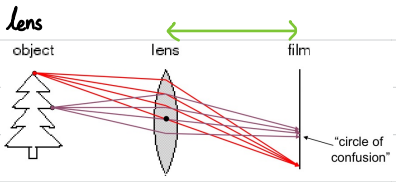
H → 좌우, 상하로 반전

but, 외곽이 모호해짐

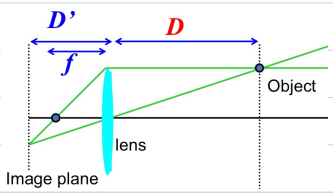


2. Camera Model

I, lens



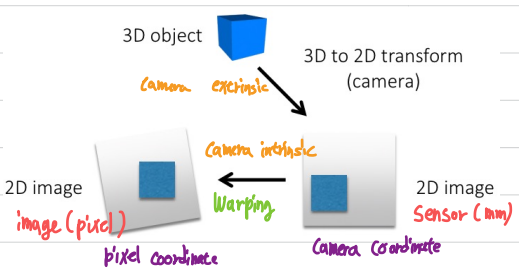
- lens focuses light onto film
- There is a specific distance at which objects are "in focus"
- Shape of lens → change this distance



- Any point satisfying thin lens equation is in focus

$$\frac{1}{D'} + \frac{1}{D} = f$$

II, coordinate transformation



camera matrix

$$x = PX$$

2D image (u_{img}, v_{img}) : 3D point (x, y, z)

3x1 = [3x4] x [4x1]

$$\tilde{x}_c = R \cdot (\tilde{X}_w - \tilde{C}) \Rightarrow \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad \text{or} \quad X_c = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} X_w$$

(3x3) (3x1)

finally, $x = PX_w$,

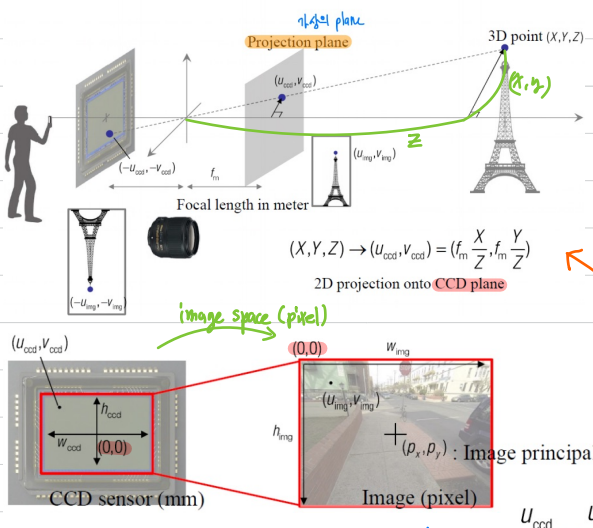
$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{I} \mid \mathbf{0}] \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ \mathbf{0} & 1 \end{bmatrix}$$

sensor → image intrinsic parameters (3x3) : camera internals information

3D → 2D (to sensor) (3x4)

$X_w \rightarrow X_c$: extrinsic parameters (4x4) : camera external information

· 3D Point Projection [Meter level → pixel level]



f_m : focal length in meter

f_x : focal length in pixel

$$u_{img} = f_m \frac{w_{img}}{w_{ccd}} \frac{X}{Z} + p_x \qquad \frac{f_x}{}$$

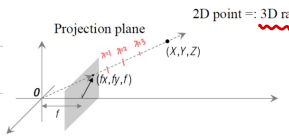
$$v_{img} = f_m \frac{h_{img}}{h_{ccd}} \frac{Y}{Z} + p_y \qquad \frac{}{f_y}$$

비율

$$\frac{u_{ccd}}{w_{ccd}} = \frac{u_{img} - p_x}{w_{img}} \qquad \frac{v_{ccd}}{h_{ccd}} = \frac{v_{img} - p_y}{h_{img}}$$

$$u_{img} = u_{ccd}\frac{w_{img}}{w_{ccd}} + p_x \qquad v_{img} = v_{ccd}\frac{h_{img}}{h_{ccd}} + p_y$$

· Homogeneous coordinates



$$(x, y) \rightarrow (x, y, 1) = f(x, y, 1) = \lambda(x, y, 1) \Rightarrow \lambda \begin{bmatrix} u_{img} \\ v_{img} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$\lambda u_{img} = f_x X + p_x Z$

$u_{img} = \frac{f_x}{\lambda} X + p_x \frac{Z}{\lambda}$ $\therefore \lambda = Z$

· Projection matrix

$$\pi = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3x3} & -C \\ 0\ 0\ 0 & 1 \end{bmatrix} = K[R \mid -Rc]$$

intrinsic : projection : rotation : translation

K : converts 3D (camera's coordinate system) ⟶ 2D (image points in image(pixel) coordinates)

$$F = \alpha(F - F * H) = (1+\alpha)F - \alpha(F * H) = F * ([1+\alpha]e - \alpha H)$$

F : image

F * H : blurred image

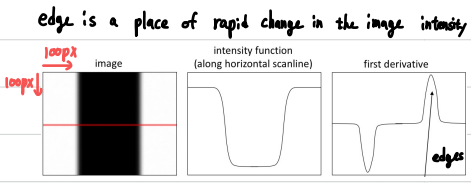
unit impulse (identity kernel with single 1 in center, zeros elsewhere)

5. Geometric Vision

1. Edge detection

I. Characterizing edges

edge is a place of rapid change in the image intensity function



edges correspond to extrema of derivative

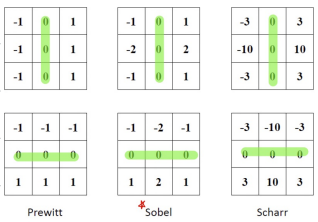
II. Image derivatives

· take discrete derivative (finite difference) using linear filter

$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$

$\frac{\partial f}{\partial x}$: H_x = [1 -1] $\frac{\partial f}{\partial y}$: H_y = [-1; 1]

Various Derivative Mask.



· Sobel operator : common approximation of derivative of Gaussian

$\frac{1}{8}$ s_x = [-1 0 1; -2 0 2; -1 0 1] $\frac{1}{8}$ s_y = [1 2 1; 0 0 0; -1 -2 -1]

1/8 doesn't make difference for edge detection but is needed to get the right gradient magnitude

· image gradient

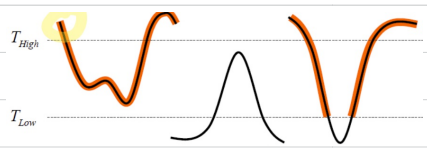
$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$, $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$, $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

III. Canny edge detector

i. Filtering image with Gaussian Filter (노이즈 제거)
ii. " derivative of Gaussian (ex. Sobel)
iii. Find magnitude and orientation of gradient
iv. Apply Non-maximum suppression
- 한개의 object에 많은 픽셀이 생기면 가장 높은 확률값(response)를 가지는 픽셀만 남기고 나머지 제거

V. Linking and thresholding

- use two threshold values : T_high, T_low

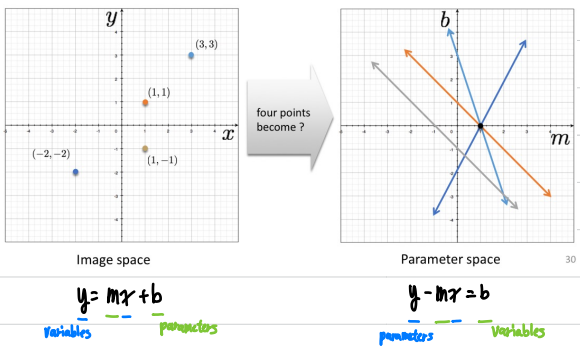


magnitude → 변화의 크기, ∝ 의미↑
i. Thigh 보다 크면 무조건 Edge 선택 Link 되어 있는지 또한 Edge
ii. But Tlow 보다 작으면 Edge X

2. Find actual lanes. (Hough)

· 점들이 주어지면 대표하는 직선 찾기

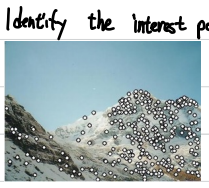
· Image and parameter space



$y = mx + b$ (variables / parameters)
$y - mx = b$ (parameters / variables)

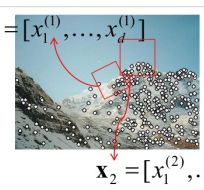
3. Feature detection

I. Detection : Identify the interest points

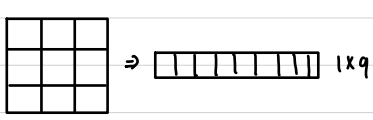


II. Description : Extract vector feature descriptor surrounding each interest point

$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$



$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

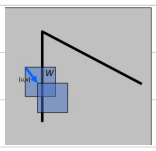


III. Matching : determine correspondence between descriptors in two views

a → b, c, ⋮ 비교

4. Feature Detection

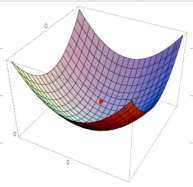
I. corner detection , consider shifting the window W by (u,v)



$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2 = Au^2 + 2Buv + Cv^2$
미분값 derivative

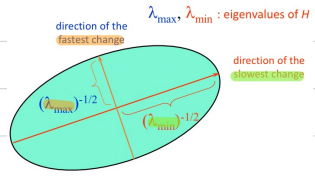
$= [u\ v] \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$

· $H = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

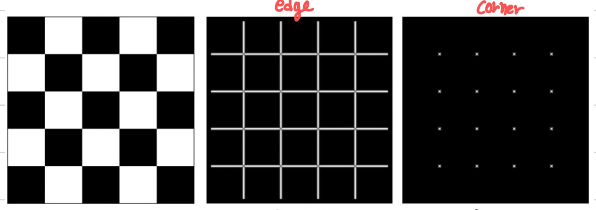


변화의 크기를 나타내는 형상

λ_{max}, λ_{min} : eigenvalues of H

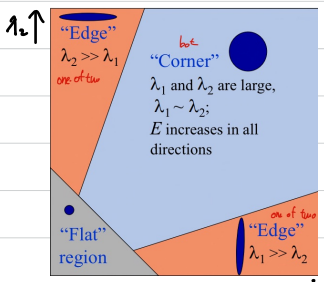


H can visualize as an ellipse



λ_{max} > Threshold : edge
λ_{min} > Threshold : corner

Corner detection : Want E(u,v) to be larger for small shift in all directions



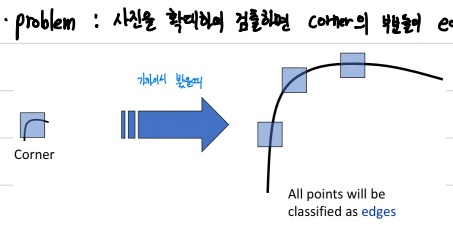
Harris λ_{min}을 계산는 과정 생략

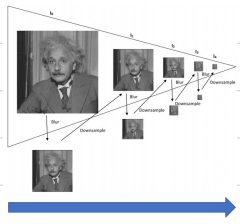
⇒ Harris operator : $f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\text{determinant}(H)}{\text{trace}(H)} = (h_n \text{th} \alpha_2)$

II. Scaling

· problem : 사진을 확대하여 검출하면 corner의 부분이 edge로 검출

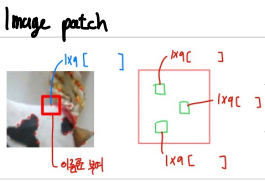
· Solution : Gaussian Pyramid,

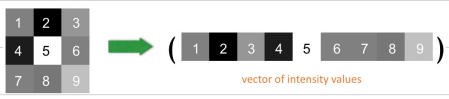




III. Description (의미 묘사)

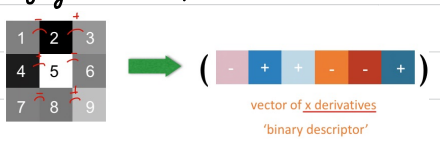
i. Image patch



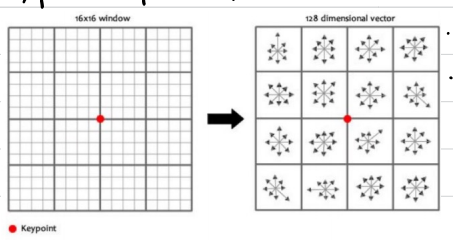


같은 feature point ⇒ 같은 Image patch를 가져야 한다. but, 제한사항: 조명, 방향, 크기

ii. Image gradients (use pixel differences)



iii. Keypoint descriptor (SIFT)



- Take 16X16 square window around detected feature.
- Divide into 4x4 grid cells
- Compute edge orientation for each pixel
- Create histogram

V. Matching

i. feature distance

Simple approach : distance $\|f_1 - f_2\|$ ⇒ can give small distances for ambiguous matches.

ratio distance = $\|f_1 - f_2\| / \|f_1 - f_2'\|$

∴ distance < Th, ratio < Th } && ⇒ match

5. Warping (뒤틀기)

Ⅰ, Homogeneous coordinates

$x = \begin{bmatrix} x \\ y \end{bmatrix}$: image point in pixel coordinates (heterogeneous)

$\bar{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ a \end{bmatrix}$: " homogeneous coordinates

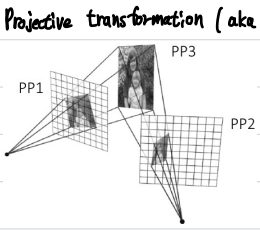
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \beta_x & 0 \\ \beta_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
shearing

Ⅱ, Projective transformation (aka Homographies, planar perspective maps)



$H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ homography matrix

Applying a homography

Ⅰ, $p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Ⅱ, $p' = H \cdot p$

Ⅲ, p'(target image) $= \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \Rightarrow p' \begin{bmatrix} x'/w' \\ y'/w' \end{bmatrix}$

original image target image

6. Semantic Vision

1. 종류

Ⅰ, Recognition, classification

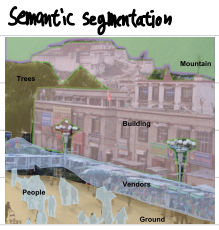
class 1 : street light

Ⅱ, Detection (: people)

Ⅲ, Identification

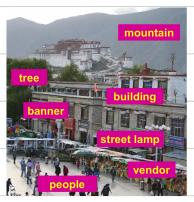
Q: Is that Potala palace

Ⅳ, Semantic segmentation

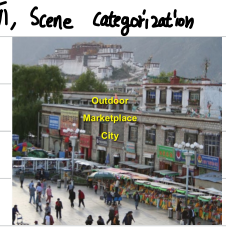


Q. What's in the scene?

Ⅴ, Object categorization



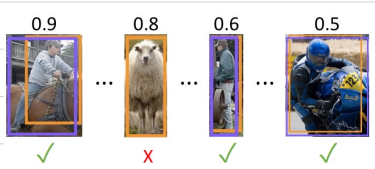
Ⅵ, Scene categorization



Q. What type of scene is it? (이미지 전체/일부)

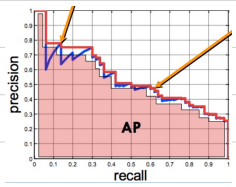
2. Object detection (classification + localization)

Ⅰ, Evaluation metric



$$\text{precision} = \frac{\#\text{ true positives}}{\#\text{true positives} + \#\text{ false positives}} = \frac{3}{4} = \frac{V}{V+X}$$

$$\text{recall} = \frac{\#\text{ true positives}}{\#\text{ ground truth objects}}$$ (뽑아 찾았어야 하는 것수 ex, 10)

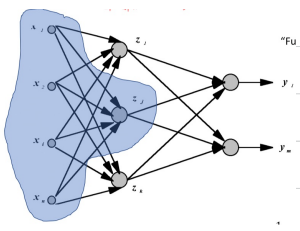


Average Precision (AP) 0% is worst

100% is best

3. CNN

Ⅰ, Standard Neural Networks



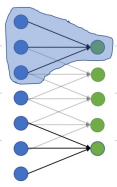
Fully connected

$Z_1 = W_1 x_1 + W_2 x_2 + \cdots + W_n x_n$

$x = (x_1, \cdots, x_{nsq})^T$, $Z_j = g(W_j^T x)$

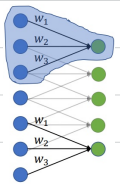
Ⅱ, Convolutional NNs (NNs → CNNs : differences)

i, local connectivity



· Each green unit is only connected to neighboring blue units

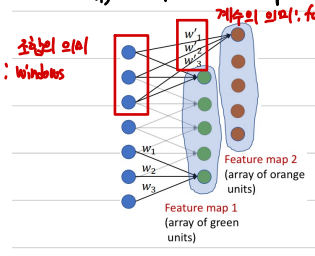
ii, Shared ("tied") weights



· Share same parameters W

· Each green unit computes the same function, but with different input window

iii, Multiple feature maps

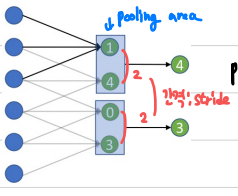


입력의 입력: windows

· All orange units compute the same function but with a different input windows

· Orange and green units compute different functions

iv, pooling (max, average)

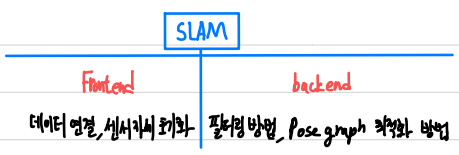


pooling area : 2 units

stride : 2 units

ICP 심화

1, ICP (iterative (반복적인) closest Point) : 반복적으로 근접된 점들을 퍼즐맞추듯 맞춘다.



2, Registeration of 3D Data Points

· Goal : point set $x_n, y_n \Rightarrow \sum \| y_n - \bar{x}_n \|^2 \rho_n \to min$, $\bar{x}_n = R x_j + t$ (translation, Rotation)

· Summary : Means of point sets : $y_0 = \frac{\sum y_n \rho_n}{\sum \rho_n}$, $x_0 = \frac{\sum x_n \rho_n}{\sum \rho_n}$

Compute mean-reduced coordinates $a_n = y_n - y_0$, $b_n = x_n - x_0$

Covarance matrix : $H = \sum a_n b_n^T \rho_n$

SVD : $SVD(H) = UDV^T$

Rotation matrix : $R = UV^T$

Translation vector : $t = y_0 - R x_0$

$\Rightarrow \bar{x}_n = R x_n + t$ (일반 좌표계 X)

· Point alignment 전개

let, $y_0 = \frac{\sum y_n \rho_n}{\sum \rho_n}$: Weighted mean of y_n

$\bar{x}_n - y_0 = R x_n + t - y_0$

$\bar{x}_n - y_0 = R(x_n + R^T t - R^T y_0)$, x_0 : new variable

$\bar{x}_n - y_0 = R(x_n - x_0)$

initial formulated problem $\sum \| y_n - \bar{x}_n \|^2 \rho_n \to min \Rightarrow \sum \| y_n - y_0 - R(x_n - x_0) \|^2 \rho_n \to min$

What we want? $R^*, x_0^* = \underset{R, x_0}{\arg\min} \sum$ "

$\Phi(x_0, R) = \sum [(y_n - y_0) - R(x_n - x_0)]^T [(y_n - y_0) - R(x_n - x_0)] \rho_n$ 1. first derivative and goes to zero

$= \sum (y_n - y_0)^T (y_n - y_0) \rho_n + \sum (x_n - x_0)^T (x_n - x_0) \rho_n - 2 \sum (y_n - y_0)^T \cdot R \cdot (x_n - x_0) \rho_n$

We need to find R that maximizes $R^* = \underset{R}{\arg\max} \sum (y_n - y_0)^T \cdot R \cdot (x_n - x_0) \rho_n$

$\because \frac{\partial \Phi(x_0, R)}{\partial x_0} = -2 \sum (x_n - x_0) \rho_n + 2 \sum R^T (y_n - y_0) \rho_n = 0$

$\Rightarrow \sum (x_n - x_0) \rho_n = R^T \sum (y_n - y_0) \rho_n$

$= 0$: y_0 is the weighted mean of y_n

$\Rightarrow \sum (x_n - x_0) \rho_n = 0 \Rightarrow \sum x_n \rho_n - \sum x_0 \rho_n = 0$

$\therefore x_0 = \frac{\sum x_n \rho_n}{\sum \rho_n}$: Weighted mean of x_n

ⓥ 공식정리

$\because a_n = y_n - y_0$, $b_n = x_n - x_0 \Rightarrow R^* = \underset{R}{\arg\max} \sum b_n^T R a_n \rho_n$

$= \underset{R}{\arg\max}\ tr(RH)$, $H = \sum (a_n b_n^T) \rho_n$

대각합

to find R that maximize $tr(RH) \Rightarrow$ SVD (singular value (특이값) decomposition)

$[m \times n] = [m \times m] \times [m \times n] \times [n \times n]$

$M = U \Sigma V^*$ (Σ: singular, unitary matrix ($U^T U = I$))

$SVD(H) = UDV^T$, Let, $R = VU^T$

$tr(RH) = tr(VU^T U D V^T) = tr(VDV^T)$ ($U^T U = I$, D: diagonal)

$tr(V D^{1/2} D^{1/2} V^T) = tr\left(V D^{\frac{1}{2}} (D^{1/2} V)^T\right)$, $V D^{\frac{1}{2}} = A$

$\therefore tr(RH) = tr(AA^T)$

for every pos. $tr(AA^T) \geq tr(R' AA^T)$ for any rotation matrix R'

$\Rightarrow tr(RH) = tr(AA^T) \geq tr(R' AA^T) = tr(R'RH)$ (R': any other rotation matrix)

$\Rightarrow R = VU^T$ was optimal as it maximize the trace

3. ICP : Point Cloud Registration Estimating the data association

* the correct correspondences are not known.

· ICP variants : Ⅰ, Consider point subsets

Ⅱ, Different data association strategies

Ⅲ, Weight the correspondences

Ⅳ, Reject potential outlier point pairs

· ICP 알고리즘 전개

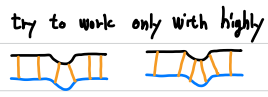
Ⅰ, Selecting Source point : i, Use all points

ii, Uniform sub-sampling

iii, Random sampling

iv, Feature-based sampling : try to work only with highly distinct points

v, Normal space sampling :



Ⅱ, Closest-Point Matching ; Find closest point in other the point set

Method : 1, kd-trees : physically closest point

2, Feature Compatibility : Match only points that have compatibly features

3, Normal shooting : Project along normal, intersect other point set to find a correspondence

4, Point-to-Plane Metric

Ⅲ, Summary

1. Select points on point cloud
2. find closest on other point cloud
3. minimize distances
4. iterate

t=k

1, use previous guess q_k, transform coordinates of current scan into the frame of previous scan

translated by roto-translation operator

$p_i^W \triangleq p_i \oplus q_k = R(\theta_k)p_i + t_k$

2, For each point p_i^W, find the two closest point in the first scan; call their indexes j_1^i, j_2^i

(point-to-line correspondence search) C_k: · all the point to line segment correspondences at step k

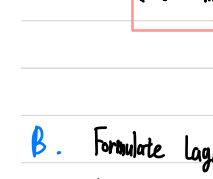
· set of tuples $\langle i, j_1^i, j_2^i \rangle$: point i is matched to segment $j_1^i - j_2^i$

3, update transform

a, Formulate point to line error objective

$J(q_{k+1}, C_k) = \sum_i n_i^T [R(\theta_{k+1})p_i + t_{k+1} - p_{j_1^i}]$

Finding distance from point to line



$\vec{n}^T\vec{a} = \vec{a}^T\vec{n} = |a||n|\cos(\theta) = |a|\cos(\theta)$

$J(q_{k+1}, C_k) = \sum_i n_i^T [R(\theta_{k+1})p_i + t_{k+1} - p_{j_1^i}]$, p_i^W, $\vec{a}$

b, Find transform q_{k+1} that minimize objective $\Rightarrow \pi_i = p_{j_1^i}$

$\min_q J(q_{k+1}, C_k) = \min_{\theta,t} \sum_i n_i^T [R(\theta_{k+1})p_i + t_{k+1} - p_{j_1^i}] = \min_{t,\theta} \sum_i \| (R(\theta)p_i + t) - \pi_i \|^2_{C_i}$

A. Solution (t_x, t_y, θ) will be found in $[\pi_1, \pi_2, \pi_3, \pi_4] \triangleq [t_x, t_y, \cos\theta, \sin\theta]$, $\pi_3^2 + \pi_4^2 = 1$

⇒ Let, $p_i = (p_{i0}, p_{i1})$:

$M_i = \begin{bmatrix} 1 & 0 & p_{i0} & -p_{i1} \\ 0 & 1 & p_{i1} & p_{i0} \end{bmatrix}$

to quadratic form $x^TAx = Q(x)$

$\min_\pi \sum_i (M_i\pi - \pi_i)^T C_i (M_i\pi - \pi_i) = \sum_i (\pi^T M_i^T C_i M_i \pi + \pi_i^T C_i \pi_i - 2\pi_i^T C_i M_i \pi)$

ignoring the constant terms, $\pi^T \underbrace{\left(\sum_i M_i^T C_i M_i\right)}_{M} \pi + \underbrace{\left(\sum_i -2\pi_i^T C_i M_i\right)}_{g} \pi$

$\pi = [t_x, t_y, \cos\theta, \sin\theta]$

by defining the matrix $W \triangleq \begin{bmatrix} 0_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & I_{2\times2} \end{bmatrix}$, constraint $\pi_3^2 + \pi_4^2 = 1 \Rightarrow \pi^T W \pi = 1$

$\therefore \min_\pi \pi^T M \pi + g^T \pi$, $\pi^T W \pi = 1$

B. Formulate Lagrangian

Quadratically Constrained Quadratic Program

$L(\pi, \lambda) = \pi^T M \pi + g^T \pi + \lambda(\pi^T W \pi - 1)$

$\frac{\partial L}{\partial \pi} = 2\pi^T M + g^T + 2\lambda \pi^T W = 0$

$\pi = -(2M + 2\lambda W)^{-T} g$, need to find lambda

using $\pi^T W \pi = 1$

$\Rightarrow g^T (2M + 2\lambda W)^{-1} W (2M + 2\lambda W)^{-T} g = 1$

let $= \begin{bmatrix} A & B \\ B^T & D + 2\lambda I \end{bmatrix}$

In the middle of the quadratic form (25) there is the matrix **W**: that matrix has a sparse form, therefore one needs to compute only the last column of $(2\mathbf{M} + 2\lambda\mathbf{W})^{-1}$. Using the

$(2M + 2\lambda W)^{-1} = \begin{bmatrix} A & B \\ B^T & (D + 2\lambda I) \end{bmatrix}^{-1} = \begin{bmatrix} * & -A^{-1}BQ^{-1} \\ * & Q^{-1} \end{bmatrix}$, where $Q = (D - B^T A^{-1} B + 2\lambda I) \triangleq (S + 2\lambda I)$

$g^T (2M + 2\lambda W)^{-1} W (2M + 2\lambda W)^{-T} g = g^T \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B}\mathbf{Q}^{-1}\mathbf{Q}^{-\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{Q}^{-1}\mathbf{Q}^{-\mathrm{T}} \\ (\mathrm{symm}) & \mathbf{Q}^{-1}\mathbf{Q}^{-\mathrm{T}} \end{bmatrix} g = 1$ (28

define Q inverse as

$Q^{-1} = (S + 2\lambda I)^{-1} = \frac{S^A + 2\lambda I}{p(\lambda)}$ where, $S^A = \det(S) \cdot S^{-1}$, $p(\lambda) = \det(S + 2\lambda I)$

$Q^{-1}Q^{-T} = \frac{\left(\mathbf{S}^A + 2\lambda\mathbf{I}\right)\left(\mathbf{S}^A + 2\lambda\mathbf{I}\right)^{\mathrm{T}}}{p(\lambda)^2} = \frac{\mathbf{S}^A\mathbf{S}^{A^{\mathrm{T}}} + 4\lambda^2\mathbf{I} + 4\lambda\mathbf{S}^A}{p(\lambda)^2}$ (3

$\therefore$

$\lambda^2 \cdot 4g^{\mathrm{T}} \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B}\mathbf{B}^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}} & -\mathbf{A}^{-1}\mathbf{B} \\ (\mathrm{symm}) & \mathbf{I} \end{bmatrix} g +$ (31)

$\lambda \cdot 4g^{\mathrm{T}} \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B}\mathbf{S}^A\mathbf{B}^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{S}^A \\ (\mathrm{symm}) & \mathbf{S}^A \end{bmatrix} g +$

$g^{\mathrm{T}} \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{A^{\mathrm{T}}}\mathbf{S}^A\mathbf{B}^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}} & -\mathbf{A}^{-1}\mathbf{B}\mathbf{S}^{A^{\mathrm{T}}}\mathbf{S}^A \\ (\mathrm{symm}) & \mathbf{S}^{A^{\mathrm{T}}}\mathbf{S}^A \end{bmatrix} g = [p(\lambda)]^2$

$\min_{t_{k+1}} \sum_i \| p_i \oplus q_{k+1} - \Pi\{ S^{ref}, p_i \oplus q_{k+1} \} \|^2$

Euclidean projector on S^{ref}

point to point : Vanilla ICP : linear convergence

point to lane : 수렴 증명은 어렵 수렴 : quadratic convergence

more closely approximates the real surface distance

repeat until convergence or loop detected

Input data : ref scan y_{t-1} (j), second scan y_t (i), q_0 : first guess for the roto-translation to be found (guess 반복수 : K)

S^{ref} : created from first scan y_{t-1}

가까운 점을 연결한 polyline

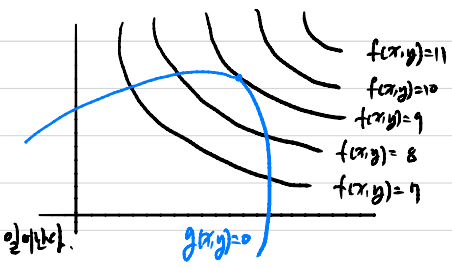
Lagrange 승수법

제약 $g(x,y) = 0$, f의 최대, 최소 ?

최대 (or 최소)는 $g(x,y) = 0$과 $f(x,y) = c$가 접할 때 일어난다.

⇒ 공통접선을 가지며 같은 법선벡터, 평행한 접선벡터

기울기벡터 ⇒ $\nabla f(a,b) = \lambda \nabla g(a,b)$



$\therefore g(x,y) = 0$, $f_x(x,y) = \lambda g_x(x,y)$, $f_y(x,y) = \lambda g_y(x,y)$: (x, y, λ) 세변수 미지수, 세개의 방정식

define S, S_A using A, B, D

? $Q = (D - B^T A^{-1} B + 2\lambda I) \triangleq (S + 2\lambda I)$

$S^A = \det(S) S^{-1}$

define Q inverse as

$Q^{-1} = (S + 2\lambda I)^{-1} = \frac{S^A + 2\lambda I}{p(\lambda)}$,