AHRS 구축

An **Unmanned aerial vehicle** (UAV) is a Unmanned Aerial Vehicle. UAVs include both autonomous (means they can do it alone) drones and remotely piloted vehicles (RPVs). A UAV is capable of controlled, sustained level flight and is powered by a jet, reciprocating, or electric engine.

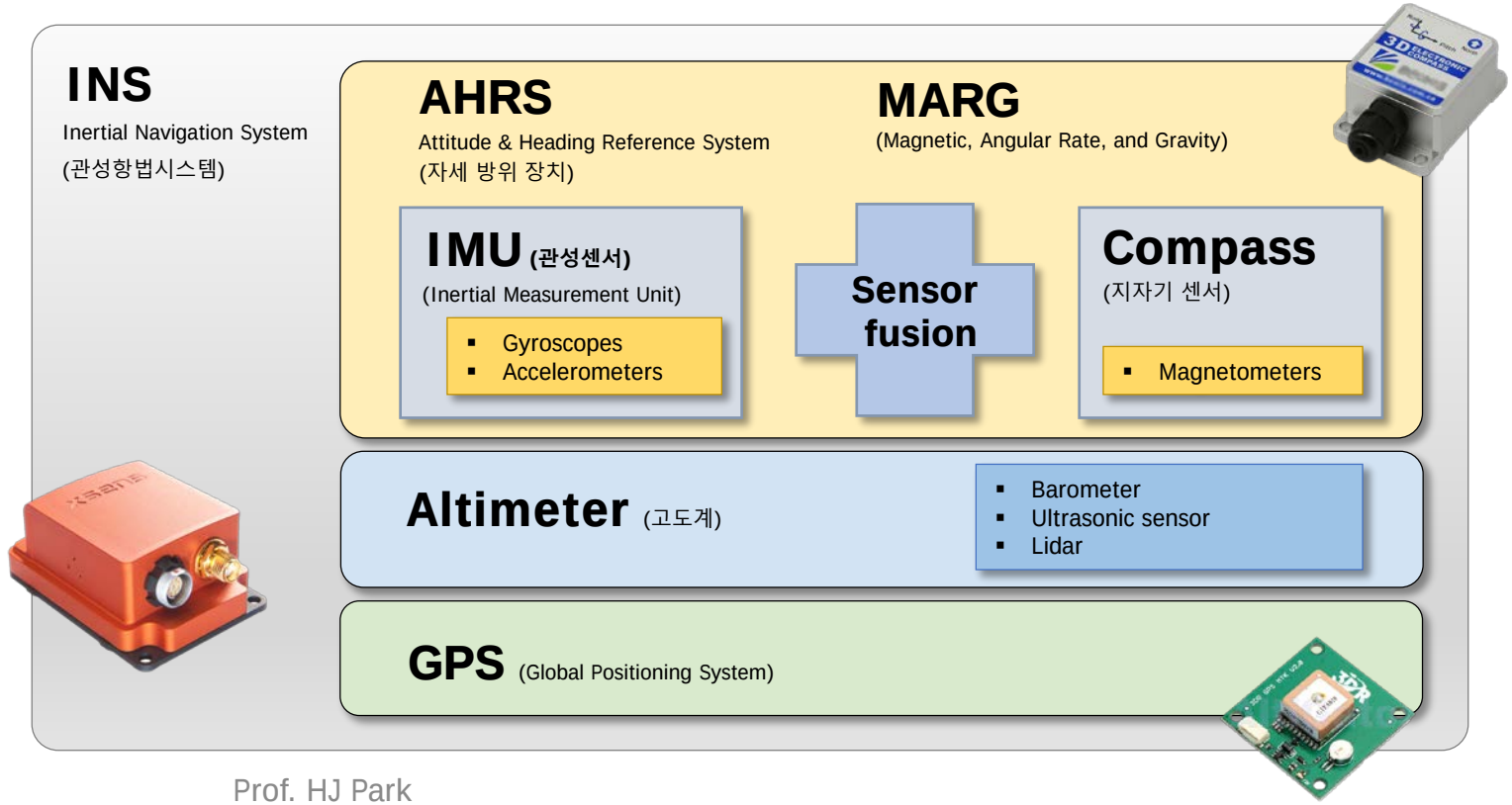
CONTENTS

AHRS 기본 개념

AHRS 기본 개념

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- AHRS(Attitude, Heading Reference System 자세방위시스템)
  - IMU센서와 Magnetometer를 융합하여 자세와 절대 방위각을 찾을 수 있는 시스템
  - MARG(Magnetic, Angular Rate, and Gravity) 라고도 불리기도 함.
  - 고도계와 GPS를 결합하여 x, y, z 좌표까지 얻을 수 있으면 관성항법시스템(INS)

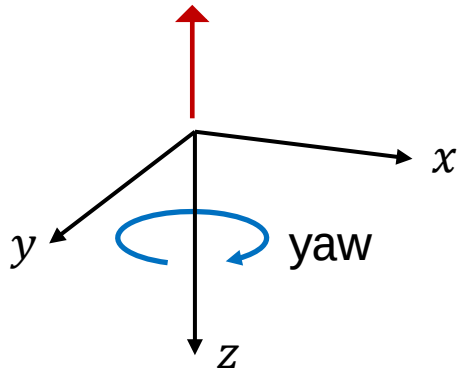


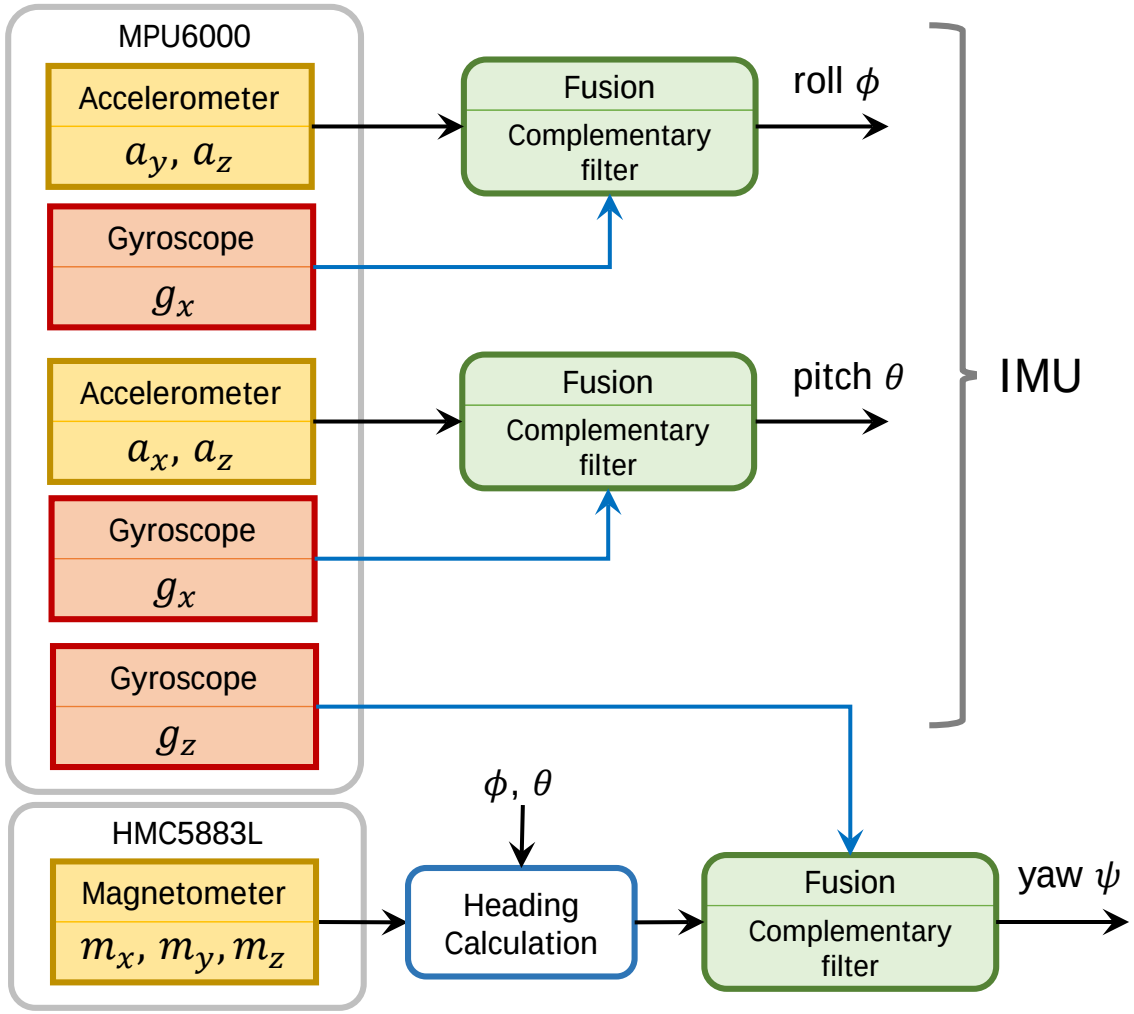
AHRS 기본 개념

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

드론의 AHRS 구축

- IMU는 중력 벡터 이용하기 때문에 동축 선상의 회전인 yaw 각도를 얻는데 정보 제공 불가능.
- Magnetometer를 이용하여 지자기력선의 벡터를 알면 yaw 정보를 얻을 수 있음.





AHRS Programming

Structure of program

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 구조
  - 클래스 MPU6000, HMC5883L 의 객체 mpu, hmc 를 선언
  - setup에서 각각을 초기화
  - loop에서 각각을 주기적으로 업데이트
  - 상보필터를 이용한 AHRS 실행

MPU_HMC_fusion.ino

```
MPU6000 mpu;
HMC5883L hmc;
setup(){
  ...
  mpu.configureMPU6000();
  hmc.init();
  ...
}
void loop(){

 AHRS(... );

}
```

class MPU6000

MPU6k.h

MPU6k.cpp

class HMC5883L

HMC5883L.h

HMC5883L.cpp

AHRS Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

code <1/3>

```
#include <SPI.h>
#include "MPU6k.h"
#include <Wire.h>
#include "HMC5883L.h"
#define BARO_CS_PIN 40
#define RadToDeg (180/PI)
#define ALPHA 0.96
extern MPU6000 mpu;
HMC5883L hmc;
float roll=0, pitch=0, yaw=0,dt=0.01;
float gyAngleX=0, gyAngleY=0, gyAngleZ=0;
uint32_t prevTime=micros();
void setup() {
  Serial.begin(115200);
  pinMode(BARO_CS_PIN, OUTPUT);  pinMode(12, OUTPUT);
  pinMode(MPU6K_CS_PIN, OUTPUT); digitalWrite(MPU6K_CS_PIN, HIGH);
  digitalWrite(BARO_CS_PIN, HIGH);   //Diselect Barometer
  SPI.begin();
  SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI_MODE0));delay(100);
  mpu.configureMPU6000();  // configure chip
  while (!hmc.begin()) delay(500);
  hmc.init();
}
```

AHRS Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

code <2/3>

```
void loop() {
  makeDt(10000);
  AHRS(roll, pitch, yaw, dt);
  Serial.print(roll*RadToDeg); Serial.print(",");
  Serial.print(pitch*RadToDeg); Serial.print(",");
  Serial.print(yaw*RadToDeg);
  Serial.print("\n");
  digitalWrite(12,!digitalRead(12));
}
void calcDT() {
  uint32_t newTime = micros();
  dt = (newTime - prevTime)/1000000.0;
  prevTime = newTime;
}
void makeDt(uint32_t p){
  uint32_t newTime;
  do {newTime = micros();}while (newTime - prevTime<p);
  prevTime = newTime;
}
```

AHRS Programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

code <3/3>

```
void AHRS(float &roll, float &pitch, float &yaw,float dt){
  mpu.updateData( );
  Vector norm = hmc.readNormalize();
  // acceleration --> angle
  digitalWrite(12,HIGH);
  float accAngleX=-atan2(mpu._accel.y,-mpu._accel.z);
  float srtAccXY=sqrt(mpu._accel.y*mpu._accel.y+mpu._accel.z*mpu._accel.z);
  float accAngleY=atan2(mpu._accel.x,srtAccXY);
  // complementary filter roll, pitch
  roll=ALPHA* (roll+mpu._gyro.x*dt)+(1-ALPHA)*accAngleX;
  pitch=ALPHA* (pitch+mpu._gyro.y*dt)+(1-ALPHA)*accAngleY;
  // magnetometer --> heading angle
  float xd=-norm.YAxis, yd=norm.XAxis, zd=-norm.ZAxis;// internal HMC
  //compensate heading angle to remove roll, pitch effects
  float s_r=sin(roll), c_r=cos(roll),s_p=sin(pitch),c_p=cos(pitch);
  float xh=xd*c_p+yd*s_r*s_p+zd*c_r*s_p;
  float yh=yd*c_r+zd*s_r;
  float heading = atan2(yh, xh);
  // complementary filter yaw
  yaw=ALPHA*(yaw+mpu._gyro.z*dt)+(1-ALPHA)*heading;
  digitalWrite(12,LOW);
}
```

AHRS 함수

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

AHRS 함수 설명

```
void AHRS(float &roll, float &pitch, float &yaw, float dt)
```

- 함수에 roll, pitch, yaw 를 참조 매개변수로 전달 &를 붙여서 결과값을 반환받을 수 있음.
- dt는 값으로 전달

```
mpu.updateData();
```

- mpu에서 측정된 값을 읽어들임.

```
Vector norm = hmc.readNormalize();
```

- hmc에서 측정된 m_x, m_y, m_z 를 읽어들임

```
float xd=-norm.YAxis, yd=norm.XAxis, zd=-norm.ZAxis;
```

- hmc의 좌표축을 변경사항 반영

AHRS 함수

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

AHRS 함수 설명

```
float s_r=sin(roll), c_r=cos(roll),s_p=sin(pitch),c_p=cos(pitch);
```

- hmc의 roll, pitch 영향을 보상하기위하여 미리 필요한 $\sin\phi, \cos\phi, \sin\theta, \cos\theta$를 구함

```
float xh=xd*c_p+yd*s_r*s_p+zd*c_r*s_p;
float yh=yd*c_r+zd*s_r;
```

- 다음 식을 반영

$$m_x^E = m_x^B c\theta + m_y^B s\phi s\theta + m_z^B c\phi s\theta$$
$$m_y^E = m_y^B c\phi + m_z^B s\phi$$

```
float heading = atan2(yh, xh);
```

- heading 각을 구함

```
yaw=ALPHA*(yaw+mpu._gyro.z*dt)+(1-ALPHA)*heading;
```

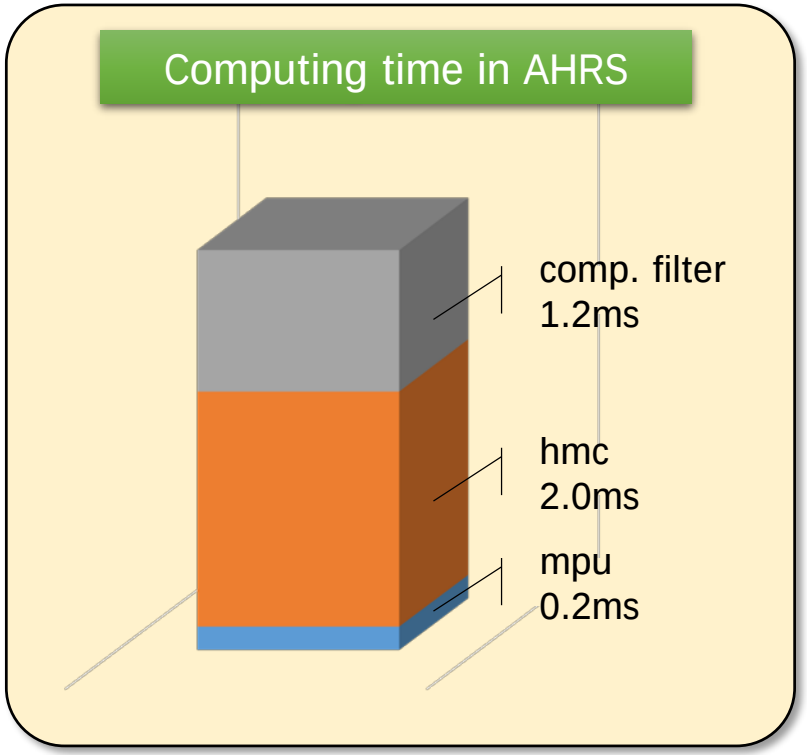
- mpu의 gyro z 값과 heading을 융합하여 yaw 값을 산출

AHRS 처리시간

AHRS 처리시간

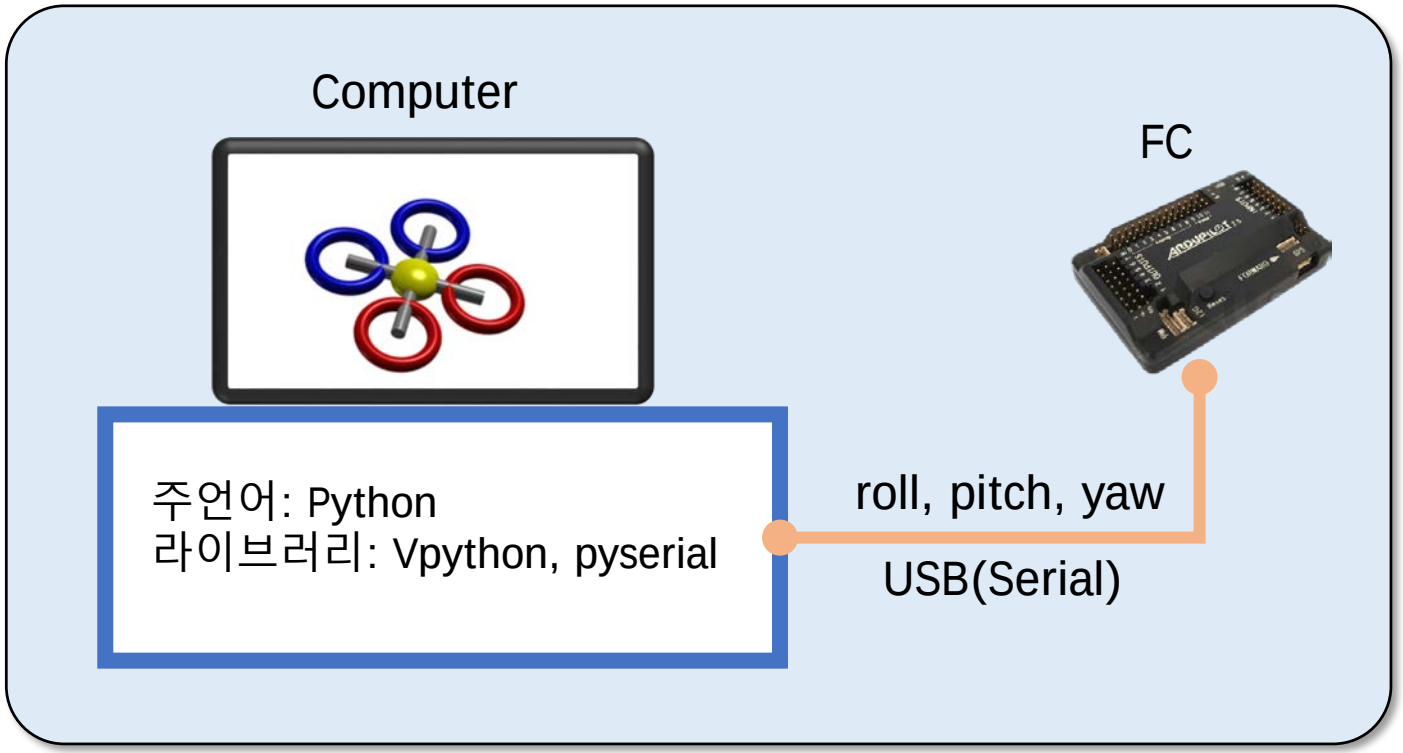
Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- loop 함수의 sample time
  - `makeDt(10000);` ➔ **10ms**
- AHRS함수 내의 소요 계산 시간
  - mpu.updateData( ) : 200us
  - hmc.readNormalize(): 2ms
  - filter 계산: 1.2 ms
- 나머지 연산
  - GPS, barometer, PID제어, motor 구동
  - 통신
  - 대략 6ms 정도에 모두 처리 필요



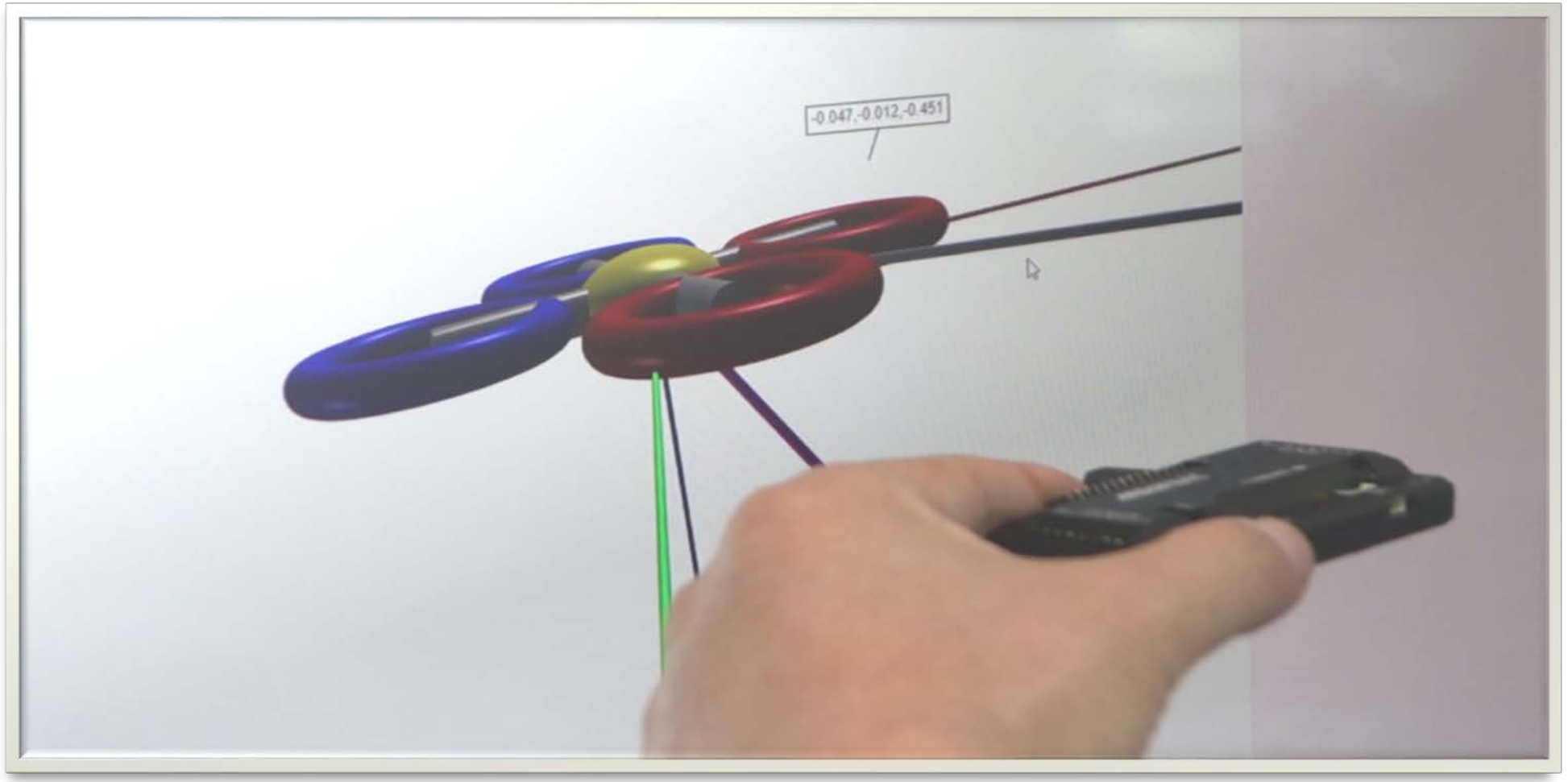
Simulation

- 검증을 위하여 python으로 시뮬레이터 구현
  - python + vpython (3D graphics)
  - threading: 통신과 그래픽 동시에 실행
  - pyserial: 직렬통신
  - 1패킷 35bytes



실험결과

- pitch, roll, yaw를 simulation으로 검증



Kalman filter의 개요

Prof. HJ Park

Kalman filter 개요

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

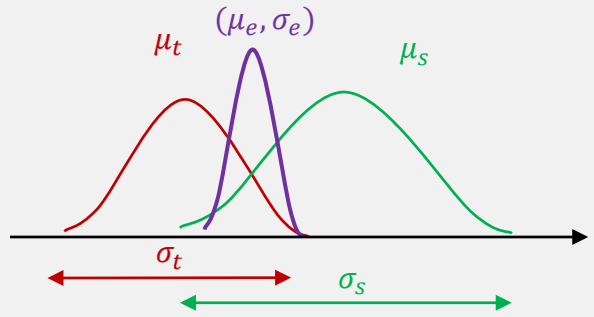
의미

- 잡음이 존재하는 선형 동역학 (linear Dynamic) 시스템에서 최적의 통계적 예측을 하는 필터
- 동역학시스템의 시간에 따른 예측과 센서 측정의 결과를 보정하는 것이 근간
  - 시간 예측 정규분포 $\mathcal{N}(\mu_t, \sigma_t)$ 과 센서 측정 $\mathcal{N}(\mu_s, \sigma_s)$ 를 곱하면 $\mathcal{N}(\mu_e, \sigma_e)$ 를 구할 수 있음.

$$\mu_e = \mu_t + k(\mu_s - \mu_t)$$
$$\sigma_e^2 = \sigma_t^2 - k\sigma_t^2$$
여기서 $k = \frac{\sigma_t^2}{\sigma_t^2 + \sigma_s^2}$

정규분포

$$\mathcal{N}(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



역사

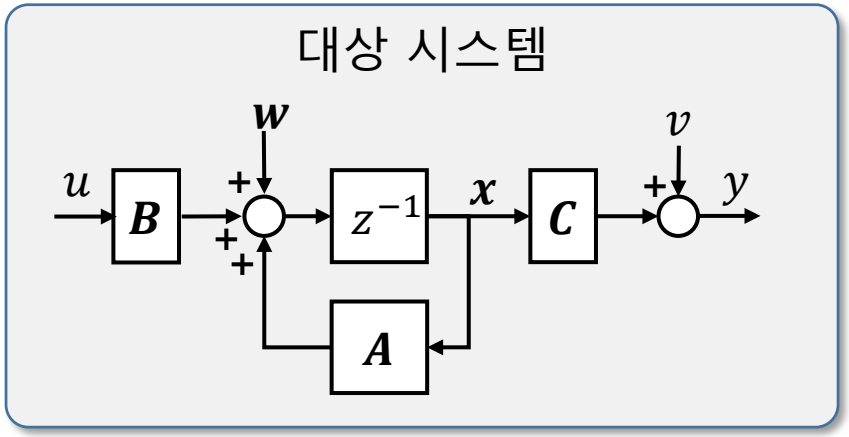
- 1960년 Rudolf E. Kalman이 개발
- 아폴로 우주선의 궤도 추정에 적용

대상 시스템 정의

대상 시스템

- 선형 이산(discrete) 시스템에 대하여 상태 방정식 정의

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}u_k + \boldsymbol{w}_k$$
$$y_k = \boldsymbol{C}\boldsymbol{x}_k + v_k$$



여기서 $\boldsymbol{A}$ 는 상태천이행렬(State transition matrix)이고, $\boldsymbol{B}$ 는 입력 행렬

$\boldsymbol{x}_k$는 상태 벡터, u_k는 입력, y_k는 출력

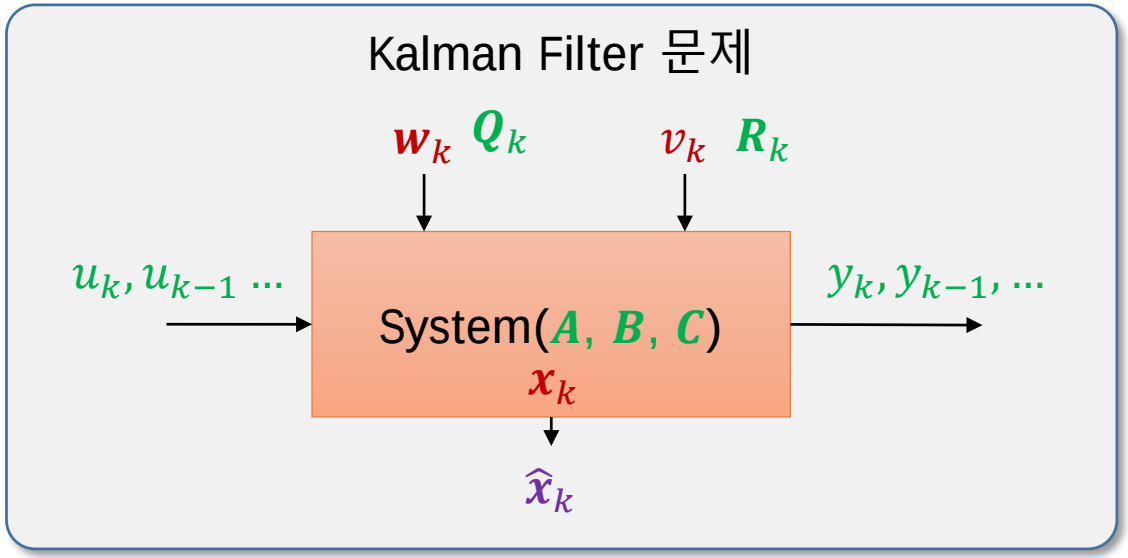
w_k는 시스템 잡음으로 $\mathcal{N}(0, \boldsymbol{Q}_k)$ 이고, v_k는 측정 잡음으로서 $\mathcal{N}(0, \boldsymbol{R}_k)$ 로 가정

$\boldsymbol{Q}_k$, $\boldsymbol{R}_k$ 는 각각 시스템 잡음과 측정 잡음의 공분산(covariance) 행렬

Kalman filter 문제

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

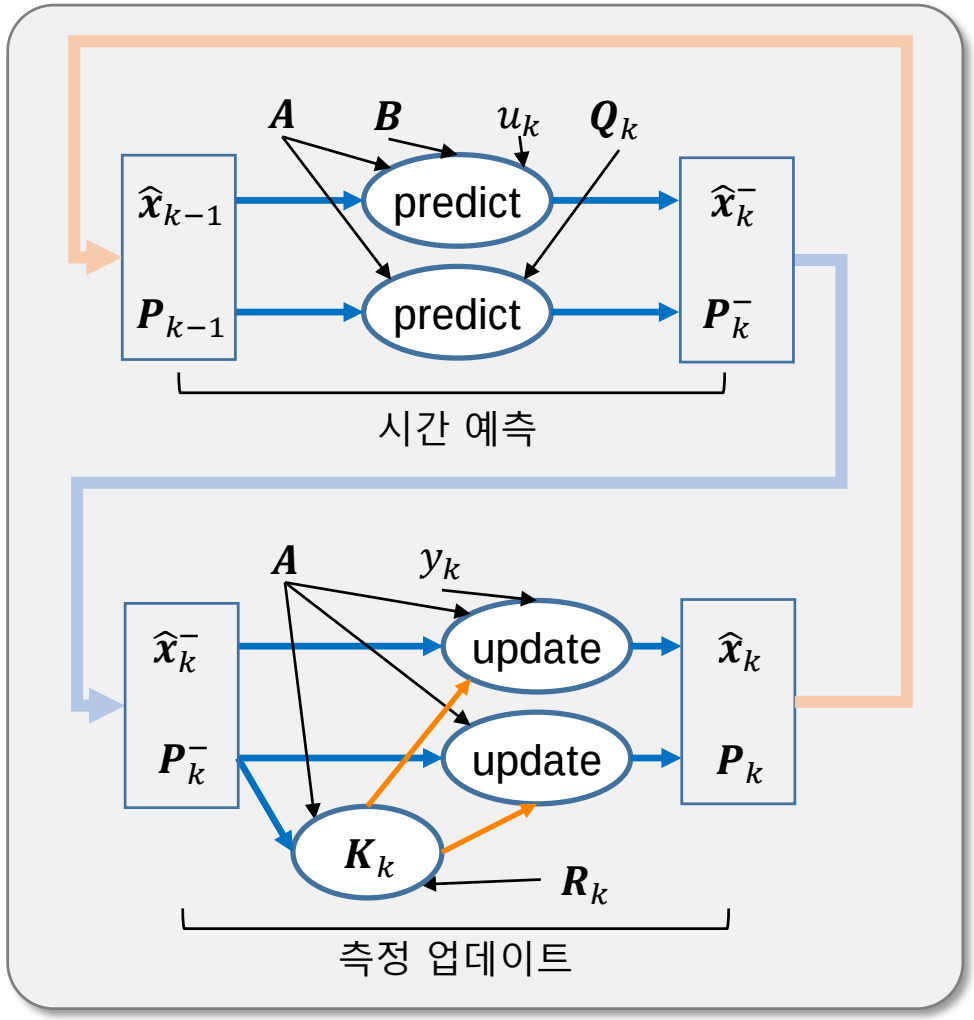
- 알고 있는 정보
  - 시스템 파라미터: $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$
  - 입력 : $u_k, u_{k-1} \cdots$
  - 출력 : $y_k, y_{k-1}, \cdots$
- 모르는 정보
  - 시스템 잡음: $\boldsymbol{w}_k$
  - 측정 잡음: v_k
  - 상태벡터: $\boldsymbol{x}_k$
- 추정하고자하는 정보
  - 상태벡터 추정: $\hat{\boldsymbol{x}}_k$



Kalman filter 알고리즘

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 알고리즘
  - 시간 예측 단계
    - 시스템의 상태 방정식을 기반
    - 시간 변화에 따른
      - 상태 벡터 예측
      - 상태 오차 공분산 예측
  - 측정 업데이트 단계
    - 센서측정으로 예측 오차 보정
    - 상태 오차 공분산 예측값으로 이득 구함
    - 측정값으로 평균(추정값) 보정
    - 상태 오차 공분산 보정
  - 이 두 과정 반복



Kalman filter: 시간 예측 단계

시간 예측 단계 (Time prediction)

- 시간에 대한 상태 벡터 예측

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}u_k$$

여기서, $\hat{\boldsymbol{x}}_{k-1}$은 이전 스텝에서의 최적 추정 상태 벡터

$\hat{\boldsymbol{x}}_k^-$ 는 상태 벡터의 예측이고, ' - '는 과거의 정보($\hat{\boldsymbol{x}}_{k-1}$)를 근거한다는 의미

- 시간에 대한 오차 공분산 예측

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q}_k$$

여기서 $\boldsymbol{P}_{k-1}$ 은 이전 스텝의 상태 오차의 공분산

$\boldsymbol{P}_k^-$는 과거의 정보($\boldsymbol{P}_{k-1}$)로부터 시간 변화를 반영하여 구한 상태 오차의 공분산

Kalman filter: 측정 업데이트 단계

측정 업데이트 단계 (Measurement update)

- Kalman Gain 계산

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{C}^T (\boldsymbol{C}\boldsymbol{P}_k^- \boldsymbol{C}^T + \boldsymbol{R}_k)^{-1}$$

여기서, $\boldsymbol{K}_k$는 Kalman gain,

공분산의 추정값으로 보정을 위한 이득을 구함

- 측정에 의한 상태 보정

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k (y_k - C\hat{\boldsymbol{x}}_k^-)$$

이식은 출력 오차 $y_k - C\hat{\boldsymbol{x}}_k^-$ 를 근거로 $\hat{\boldsymbol{x}}_k^-$을 보정

Kalman filter: 측정 업데이트 단계

측정 업데이트 단계 (Measurement update)

- 오차 공분산 보정

$$\boldsymbol{P}_k = (I - \boldsymbol{K}_k \boldsymbol{C})\boldsymbol{P}_k^-$$

이식은 $\boldsymbol{K}_k$를 사용하여 $\boldsymbol{P}_k^-$ 를 $\boldsymbol{P}_k$ 로 보정

Kalman filter 정리

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

총정리

시간 예측 단계		
상태 벡터	$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}u_k$	예측값 산출
오차의 공분산	$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q}_k$	공분산 예측값 산출
측정 업데이트 단계		
칼만 게인	$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_k^-\boldsymbol{C}^T + \boldsymbol{R}_k)^{-1}$	공분산 예측값으로 게인 산출
상태벡터 보정	$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(y_k - C\hat{\boldsymbol{x}}_k^-)$	예측값과 측정값 비교 및 보정
오차 공분산 보정	$\boldsymbol{P}_k = (I - \boldsymbol{K}_k\boldsymbol{C})\boldsymbol{P}_k^-$	

시간 예측 단계

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}u_k$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q}_{k-1}$$

측정 보정 단계

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_k^-\boldsymbol{C}^T + \boldsymbol{R}_k)^{-1}$$

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(y_k - C\hat{\boldsymbol{x}}_k^-)$$

$$\boldsymbol{P}_k = (I - \boldsymbol{K}_k\boldsymbol{C})\boldsymbol{P}_k^-$$

IMU Roll angle에 적용

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

시스템 방정식 유도

- 주요 전제
  - 샘플 시간: Δt,
  - 가속도로부터 구한 roll 각: ϕ_k^a,
  - gyro로 부터 구한 각속도: ω_k
  - gyro 의 drift : b_k
- 구하고자 하는 roll각: ϕ_k

$$\phi_k = \phi_{k-1} + (\omega_k - b_{k-1})\Delta t$$
$$b_k = b_{k-1}$$

- 시스템 상태방정식에 적용

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}u_k + \boldsymbol{w}_k$$
$$y_k = \boldsymbol{C}\boldsymbol{x}_k + v_k$$

여기서, $\boldsymbol{x}_k = \begin{bmatrix}\phi_k \\ b_k\end{bmatrix}$, $u_k = \omega_k$, $\boldsymbol{A} = \begin{bmatrix}1 & -\Delta t \\ 0 & 1\end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix}\Delta t \\ 0\end{bmatrix}$, $\boldsymbol{C} = [1 \quad 0]$

$$\begin{bmatrix}\phi_k \\ b_k\end{bmatrix} = \begin{bmatrix}1 & -\Delta t \\ 0 & 1\end{bmatrix}\begin{bmatrix}\phi_{k-1} \\ b_{k-1}\end{bmatrix} + \begin{bmatrix}\Delta t \\ 0\end{bmatrix}\omega_k$$
$$y_k = [1 \quad 0]\begin{bmatrix}\phi_k \\ b_k\end{bmatrix}$$

IMU Roll angle에 적용

- 공분산

  $\boldsymbol{Q}_k = \begin{bmatrix} q_\omega & 0 \\ 0 & q_b \end{bmatrix}$, $\boldsymbol{R}_k = r_a$는 각각 시스템과 측정 잡음의 공분산으로 가정

  q_ω 는 gyro 로 부터의 잡음,

  q_b는 gyro의 drift 관련 ,

  r_a는 가속도센서의 측정 잡음

- 오차 공분산,

  $\boldsymbol{P}_k = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$,

- Kalman gain

  $\boldsymbol{K}_k = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$

식의 간략화

계산의 간략화

- 시간 예측 단계
  - $\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}u_k$

    $x_1 = x_1 + (u - x_2)\Delta t,$

    $x_2 = x_2$

  - $\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q}_k$

    $$P = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix} + \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}$$

    $$= \begin{bmatrix} p_{11} - (p_{21} + p_{12} - p_{22}\Delta t)\Delta t + q_1 & p_{12} - p_{22}\Delta t \\ p_{21} - p_{22}\Delta t & p_{22} + q_2 \end{bmatrix}$$

시간 예측 단계

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{B}u_k$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_{k-1}\boldsymbol{A}^T + \boldsymbol{Q}_{k-1}$$

측정 보정 단계

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_k^-\boldsymbol{C}^T + \boldsymbol{R}_k)^{-1}$$

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(y_k - C\hat{\boldsymbol{x}}_k^-)$$

$$\boldsymbol{P}_k = (I - \boldsymbol{K}_k\boldsymbol{C})\boldsymbol{P}_k^-$$

계산의 간략화

- 측정 보정 단계
  - $\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{C}^T (\boldsymbol{C}\boldsymbol{P}_k^- \boldsymbol{C}^T + \boldsymbol{R}_k)^{-1}$ ← Kalman gain

    $\boldsymbol{C}\boldsymbol{P}_k^- \boldsymbol{C}^T + \boldsymbol{R}_k = p_{11} + r_a$

    $K = \begin{bmatrix} p_{11} \\ p_{21} \end{bmatrix} \frac{1}{p_{11}+r_a}$
  - $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k (y_k - C\hat{\boldsymbol{x}}_k^-)$ ← 상태벡터 보정

    $y_k = \phi_k^a$, $C\hat{\boldsymbol{x}}_k^- = x_1$

    $\hat{\boldsymbol{x}}_k = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} (\phi_k^a - x_1)$ ← 추정 오차 (측정값 - 추정값)
  - $\boldsymbol{P}_k = (I - \boldsymbol{K}_k \boldsymbol{C}) \boldsymbol{P}_k^-$ ← 오차 공분산 보정

    $\boldsymbol{P} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} k_1 & 0 \\ k_2 & 0 \end{bmatrix} \right) \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$

    $= \begin{bmatrix} 1-k_1 & 0 \\ -k_2 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} = \begin{bmatrix} (1-k_1)p_{11} & (1-k_1)p_{12} \\ p_{21} - k_2 p_{11} & p_{22} - k_2 p_{12} \end{bmatrix}$

Kalman filter programming

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

Kaman filter 함수

```
float q1=0.001,q2=0.003, r1=0.03;
float kalmanFilter(float u, float y){
    static float x1=0, x2=0,p11=0,p12=0,p21=0, p22=0;
    //= Time Update State, error covariance
    x1+=(u-x2)*dt;
    p12+=-p22*dt;   p11-=(p21+p12)*dt+q1*dt;
    p21+=-p22*dt;   p22+=q2*dt;
    //= Measurement Update Gain, State, error covariance
    float s=p11+r1,k1=p11/s, k2=p21/s, e=y-x1;
    x1+=k1*e;       x2+=k2*e;
    p11*=(1-k1);    p12*=(1-k1);
    p21+=-k2*p11;   p22+=-k2*p12;
    return x1;
}
```

Kalman filter programming

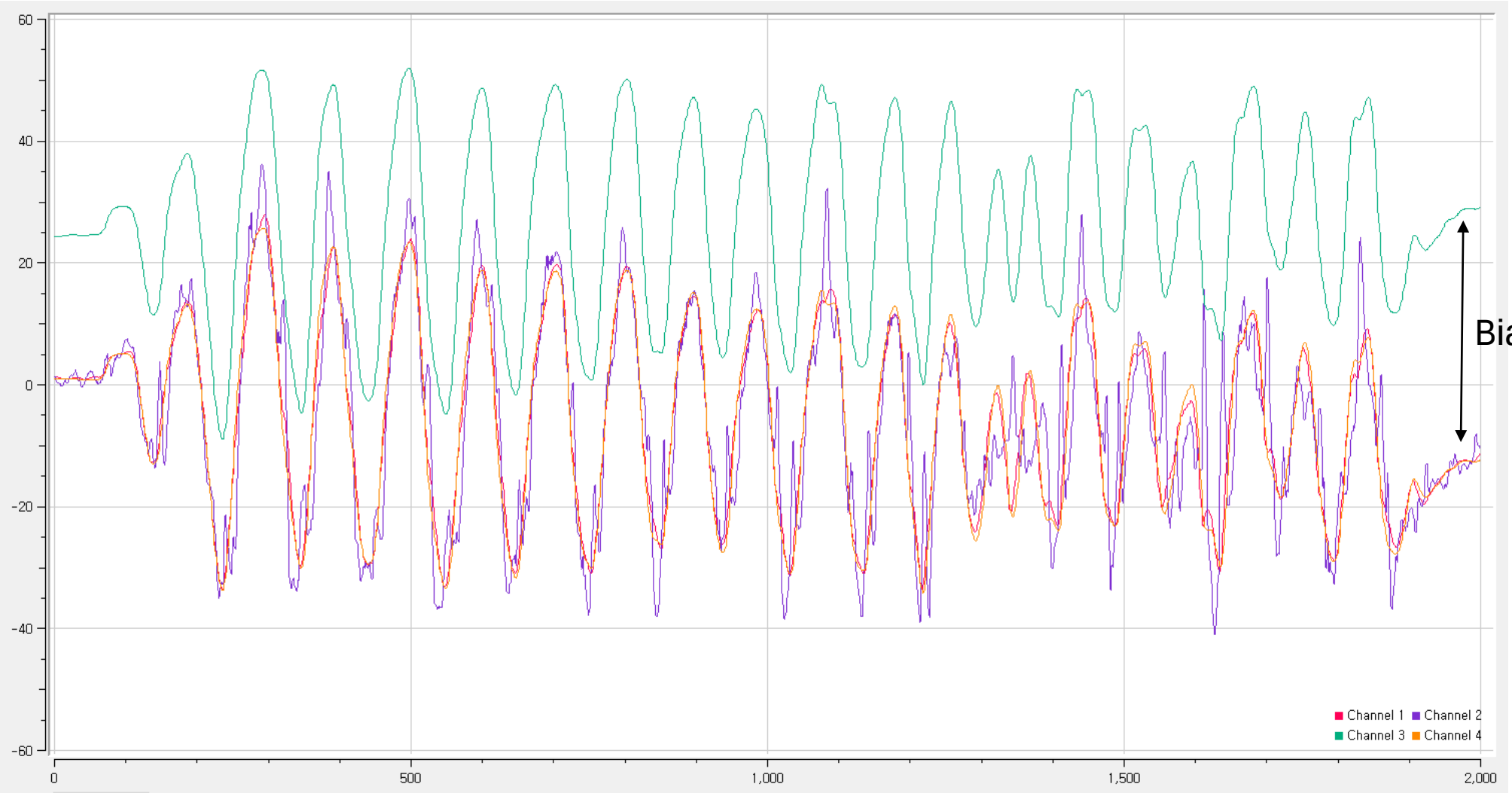
Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- loop에 Kaman filter 반영

```
void loop() {
  calcDT();
  mpu.updateData();
  float accAngleX=-atan2(mpu._accel.y,-mpu._accel.z);
  float srtAccXY=sqrt(mpu._accel.y*mpu._accel.y+mpu._accel.z*mpu._accel.z);
  float accAngleY=atan2(mpu._accel.x,srtAccXY);
  roll=ALPHA* (roll+mpu._gyro.x*dt)+(1-ALPHA)*accAngleX; //Complementary
  float roll_kalman=kalmanFilter(mpu._gyro.x, accAngleX); //Kalman filter
  pitch=ALPHA* (pitch+mpu._gyro.y*dt)+(1-ALPHA)*accAngleY;
  gyAngleX=gyAngleX+mpu._gyro.x*dt;
  Serial.print(roll*RAD_TO_DEG); Serial.print(",");
  Serial.print(accAngleX*RAD_TO_DEG);  Serial.print(",");
  Serial.print(gyAngleX*RAD_TO_DEG); Serial.print(",");
  Serial.print(roll_kalman*RAD_TO_DEG);
  Serial.print("\n");
}
```

상보필터와 비교

Dept. of Mechanical System Design, Seoul National University of Science and Technology.



ϕ 상보필터

ϕ^a 가속도

ϕ^ω 각속도

ϕ^K 칼만

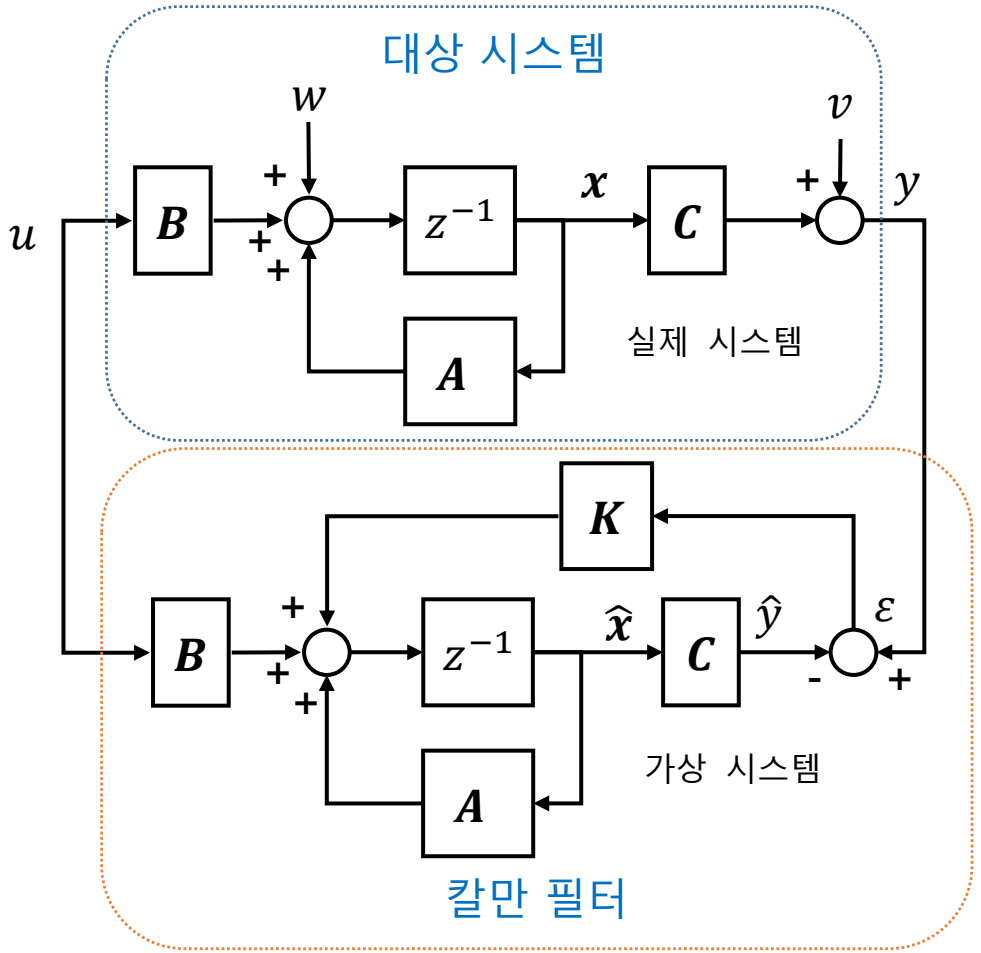
SEOULTECH

Kalman filter의 또다른 관점

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

칼만 필터의 또 다른 관점

- 대상시스템의 x를 추정하는 것이 목적
- 똑같은 가상 모델을 구축하고 똑같은 입력u를 인가하면 대상 시스템의 출력 y 는 가상 시스템의 출력 $\hat{y}$ 와 같아야 함.
- 그런데 잡음 $w,\ v$ 가 있고 $t = 0$에서의 상태 벡터의 초기값 $x_0 \neq \hat{x}_0$이기 때문에 $y \neq \hat{y}$ 임.
- 이들이 같아지도록 확률관점에서 최적의 K를 찾아 보정하여 $\hat{x} \rightarrow x$가 되도록 하는 방법론이 Kalman filter !



Quaternion 적용

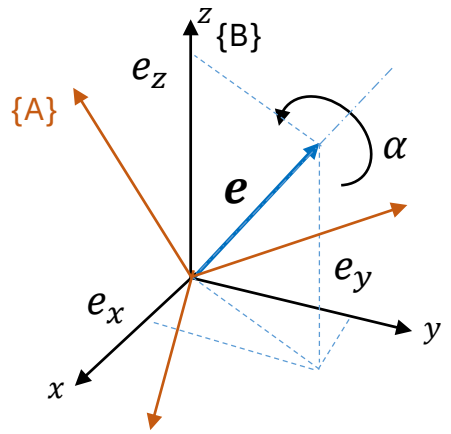
Gimbal Lock

- Gimbal lock은 Euler angle로 자세를 표시할 때 발생하는 문제
  - pitch θ가 $\frac{\pi}{2}$일 때, 회전행렬을 구해보면
    - $R(\psi,z)R\left(\frac{\pi}{2},y\right)R(\phi,x)=\begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi-c\phi s\psi & c\phi s\theta c\psi+s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi+c\phi c\psi & c\phi s\theta s\psi-s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}$
    - $=\begin{bmatrix} 0 & s\phi c\psi-c\phi s\psi & c\phi c\psi+s\phi s\psi \\ 0 & s\phi s\psi+c\phi c\psi & c\phi s\psi-s\phi c\psi \\ -1 & 0 & 0 \end{bmatrix}=\begin{bmatrix} 0 & s(\phi-\psi) & c(\phi-\psi) \\ 0 & c(\phi-\psi) & -s(\phi-\psi) \\ -1 & 0 & 0 \end{bmatrix}$
    - R 이 주어졌을 경우 roll, yaw 를 독립적으로 정할 수 없음.
- Gimbal lock의 문제를 해결하기 위하여 Quaternion 도입

Quaternion (사원수): 정의

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

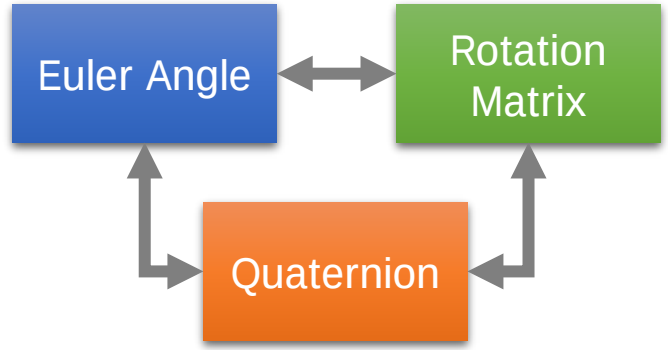
- 쿼터니언(Quaternion; 사원수)
  - Gimbal Lock 없이 물체의 회전이나 자세를 정의할 때 사용
  - 회전행렬은 9개의 원소를 갖지만 쿼터니언은 4개로 간단.
- 정의
  - 좌표계 {B}에서 {A} 좌표로의 회전을 쿼터니언 $\boldsymbol{q}_{\boldsymbol{A}}^{\boldsymbol{B}}$ 정의



$$\boldsymbol{q}_{\boldsymbol{A}}^{\boldsymbol{B}} \triangleq [q_0 \quad q_1 \quad q_2 \quad q_3]^T = \left[\cos\frac{\alpha}{2} \quad e_x \sin\frac{\alpha}{2} \quad e_y \sin\frac{\alpha}{2} \quad e_z \sin\frac{\alpha}{2}\right]^T \quad (1)$$

여기서, 이 회전의 $\boldsymbol{e} = [e_x \quad e_x \quad e_x]^T$는 중심축의 단위 행렬
그리고 α는 그 축으로의 회전각.

- 쿼터니언, 오일러각, 회전행렬 표현
  - 이들 간의 상호변환이 가능함.



Quaternion (사원수): 연산

쿼터니언 연산

- 크기

$$\|\boldsymbol{q}\| = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2} \tag{2}$$

- Conjugate:

  $\boldsymbol{q}_{\boldsymbol{A}}^{\boldsymbol{B}*} = \boldsymbol{q}_{\boldsymbol{B}}^{\boldsymbol{A}} = [q_0 \quad -q_1 \quad -q_2 \quad -q_3]^T$

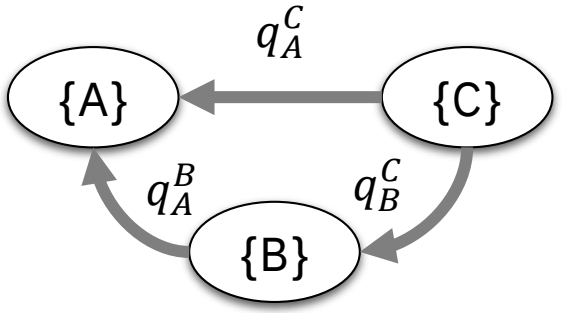
  축을 반대방향으로 하고 같은 각을 돌리는 개념

- Quaternion product
  - 좌표계{A}, {B}, {C} 에서 {C}좌표에서 본 {C} 좌표를 $\boldsymbol{q}_{\boldsymbol{A}}^{\boldsymbol{C}}$ 라고하면

    $\boldsymbol{q}_{\boldsymbol{A}}^{\boldsymbol{C}} = \boldsymbol{q}_{\boldsymbol{B}}^{\boldsymbol{C}} \otimes \boldsymbol{q}_{\boldsymbol{A}}^{\boldsymbol{B}}$

    여기서, $\otimes$ 은 쿼터니언 곱 연산자이고 다음과 같이 정의

$$\boldsymbol{p} \otimes \boldsymbol{q} \triangleq \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2 \\ p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1 \\ p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0 \end{bmatrix} \tag{3}$$



Quaternion (사원수): 응용

쿼터니언 응용

- 일반 벡터의 쿼터니언 표현
  - 벡터 $\boldsymbol{v}$ 를 쿼터니언 표현을 $\boldsymbol{v_q}$

$$\boldsymbol{v_q} = [0 \quad v_x \quad v_y \quad v_z]^T$$

- 좌표계 {A}, {B}에서 본 $\boldsymbol{v_q}$를 각각 $\boldsymbol{v_q^A}$, $\boldsymbol{v_q^B}$라 하면

$$\boldsymbol{v_q^B} = \boldsymbol{q_A^B} \otimes \boldsymbol{v_q^A} \otimes \boldsymbol{q_A^{B*}} \tag{4}$$

이식은 다음 회전 행렬 표현과 동등

$$\boldsymbol{v^B} = \boldsymbol{R_A^B}(\boldsymbol{q})\boldsymbol{v^A} \tag{5}$$

- (3)을 (4)식에 적용하여 (5)의 회전행렬을 구함

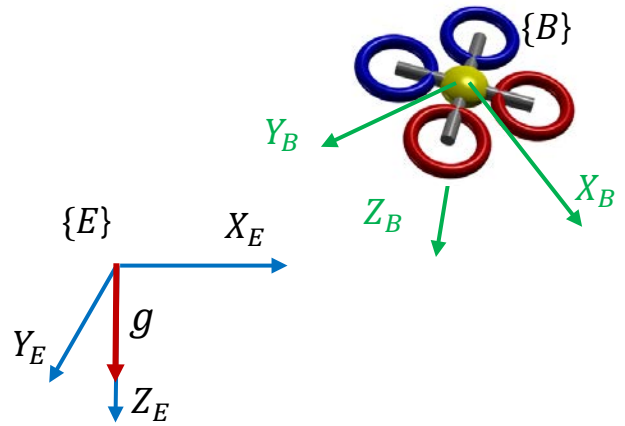
$$\boldsymbol{R_A^B}(\boldsymbol{q}) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{6}$$

Quaternion (사원수): 센서 융합에 적용

- 지구 좌표계 {E}에 대하여
  - {E}의 정규화 중력
    $\boldsymbol{g^E} = [0 \quad 0 \quad 1]^T$
  - {E}의 정규화 지자기 벡터
    $\boldsymbol{h^E} = [h_x \quad h_y \quad h_z]^T$
- Body 좌표계 {B} 에 대하여
  - {B}의 가속도센서에서의 정규화 가속도 벡터
    $\boldsymbol{a^B} = [a_x \quad a_y \quad a_z]^T$, $\|\boldsymbol{a^B}\|=1$
  - {B}의 지자기센서에서의 정규화 지자기 벡터
    $\boldsymbol{m^B} = [m_x \quad m_y \quad m_z]^T$, $\|\boldsymbol{m^B}\|=1$
  - {B}의 자이로센서에서의 정규화 각속도 벡터
    $\boldsymbol{\omega^B} = [\omega_x \quad \omega_y \quad \omega_z]^T$, $\|\boldsymbol{\omega^B}\|=1$

정규화(normalize):
벡터 $\boldsymbol{a}$ 를 정규화한 벡터를 $\boldsymbol{n}$ 이라 하면

$$\boldsymbol{n} = \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}$$



Quaternion (사원수): 센서 융합에 적용

목표

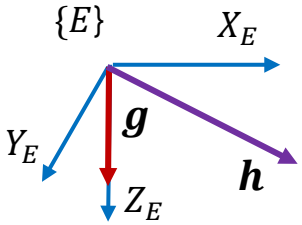
- 알고있는 불변의 정보:
  - $\boldsymbol{g^E}$: 방향- Down, 크기- 거의 9.81
  - $\boldsymbol{h^E}$: 방향- North, 크기- 지역에 따라 크게 달라짐
- 센서로부터 측정된 정보: $\boldsymbol{a^B}$, $\boldsymbol{m^B}$, $\boldsymbol{\omega^B}$
- 이 정보로부터 $\boldsymbol{q_B^E}$, 또는 $\boldsymbol{R_B^E(q)}$ 를 구하면 euler angle 을 구할 수 있음.

Quaternion (사원수): 센서 융합에 적용

- 중력, 지자기 관계식

$$\boldsymbol{R_B^E(q)a^B = g^E} \quad (7)$$

$$\boldsymbol{R_B^E(q)m^B = h^E} \quad (8)$$



  - 문제점
    - 식 6개 미지수 3개 이므로 해가 많은 과일 결정(Overdetermined)
    - 잡음, 지자기 변동 등에 해가 영향을 받음.

- 쿼터니언의 분리
  - 가속도와 지자기에 대한 항으로 분리

$$\boldsymbol{q_B^E = q_{acc} \otimes q_{mag}} \quad (9)$$

$$\boldsymbol{R_B^E(q) = R(q_{acc})R(q_{mag})} \quad (10)$$

  - 지자기의 경우 z축에 대한 회전만 사용할 것이기 때문에

$$\boldsymbol{q_{mag}} = [q_{0mag} \quad 0 \quad 0 \quad q_{3mag}]^T \quad (11)$$

Quaternion (사원수): 가속도 센서 적용

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 중력, 지자기 관계식

$$\boldsymbol{R}_{\boldsymbol{E}}^{\boldsymbol{B}}(\boldsymbol{q})\boldsymbol{g}^{\boldsymbol{E}} = \boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{acc}})\boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{mag}})\boldsymbol{g}^{\boldsymbol{E}} = \boldsymbol{a}^{\boldsymbol{B}} \quad (12)$$

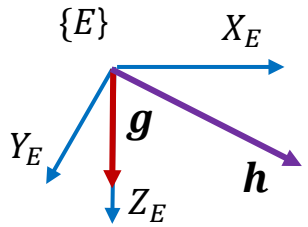
$$\boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{mag}}) = \begin{bmatrix} c\Delta & -s\Delta & 0 \\ s\Delta & c\Delta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - 즉

$$\boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{acc}})\boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{mag}})[0 \quad 0 \quad 1]^T = [a_x \quad a_y \quad a_z]^T$$

  - 중력 벡터는 z 성분만 있고 $\boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{mag}})$는 z 방향으로의 회전이기 때문에

$$\boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{acc}})[0 \quad 0 \quad 1]^T = [a_x \quad a_y \quad a_z]^T \quad (13)$$



  - 식(6)을 (13)에 적용하면

$$\begin{bmatrix} 2(q_{1acc}q_{3acc} + q_{0acc}q_{2acc}) \\ 2(q_{2acc}q_{3acc} - q_{0acc}q_{1acc}) \\ q_{0acc}^2 - q_{1acc}^2 - q_{2acc}^2 + q_{3acc}^2 \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \quad (14)$$

  - (14) 에서 $\boldsymbol{a}^{\boldsymbol{B}}$를 측정하여 알고 있기 때문에 식 3개에 미지수는 4개가 됨.

Quaternion (사원수): 가속도 센서 적용

- 식(14) 는 z축에 대한 회전 성분을 제공하지 못하기 때문에 $q_{3acc} = 0$ 라고 가정.
- 그러면 다음 식이 되고

$$\begin{bmatrix} 2q_{0acc}q_{2acc} \\ -2q_{0acc}q_{1acc} \\ q_{0acc}^2 - q_{1acc}^2 - q_{2acc}^2 \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{15}$$

- 식 (15)를 풀면,

$$q_{acc} = \begin{cases} \begin{bmatrix} \sqrt{\frac{a_z+1}{2}} & -\frac{a_y}{\sqrt{2(a_z+1)}} & \frac{a_x}{\sqrt{2(a_z+1)}} & 0 \end{bmatrix}^T, & a_z \geq 0 \\ \begin{bmatrix} -\frac{a_y}{\sqrt{2(1-a_z)}} & \sqrt{\frac{1-a_z}{2}} & 0 & \frac{a_x}{\sqrt{2(1-a_z)}} \end{bmatrix}^T, & a_z < 0 \end{cases} \tag{16}$$

Quaternion (사원수): 지자기 센서 적용

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

지자기 센서에 적용

- 식 (8)에서

$$\boldsymbol{R}_{\boldsymbol{B}}^{\boldsymbol{E}}(\boldsymbol{q})\boldsymbol{m}^{\boldsymbol{B}} = \boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{mag}})\boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{acc}})\boldsymbol{m}^{\boldsymbol{B}} = \boldsymbol{h}^{\boldsymbol{E}} \quad (17)$$

- (16)에서 $\boldsymbol{l} = \boldsymbol{R}_{\boldsymbol{B}}^{\boldsymbol{E}}(\boldsymbol{q}_{\boldsymbol{acc}})\boldsymbol{m}^{\boldsymbol{B}}$을 구하여 (17)에 대입

$$\boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{mag}})\boldsymbol{l} = \boldsymbol{h}^{\boldsymbol{E}} \quad (18)$$

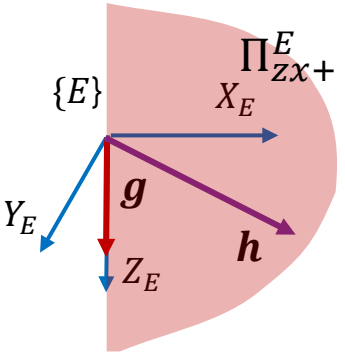
- 그림과 같이 z와 +x 축으로 구성된 평면 Π_{zx+}^{E}를 정의하면,

$$\boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{mag}})\boldsymbol{l} \in \Pi_{zx+}^{E} \quad (19)$$

- 여기에 식(14) 를 만족시키는 평면상의 벡터를 지정하면

$$\boldsymbol{R}(\boldsymbol{q}_{\boldsymbol{mag}})\boldsymbol{l} = \begin{bmatrix} \sqrt{\Gamma} & 0 & l_z \end{bmatrix}^T \quad (20)$$

여기서 $\Gamma = \sqrt{l_x^2 + l_y^2}$



Quaternion (사원수): 지자기 센서 적용

Dept. of Mechanical System Design, Seoul National University of Science and Technology.

- 지자기 센서
  - 연산을 거치면

$$q_{mag} = \begin{cases} \left[\frac{\sqrt{\Gamma + l_x\sqrt{\Gamma}}}{\sqrt{2\Gamma}} \quad 0 \quad 0 \quad \frac{l_y}{\sqrt{2}\sqrt{\Gamma + l_x\sqrt{\Gamma}}}\right]^T, & l_x \geq 0 \\ \left[\frac{l_y}{\sqrt{2}\sqrt{\Gamma - l_x\sqrt{\Gamma}}} \quad 0 \quad 0 \quad \frac{\sqrt{\Gamma - l_x\sqrt{\Gamma}}}{\sqrt{2\Gamma}}\right]^T, & l_x < 0 \end{cases} \tag{21}$$

- 센서의 융합
  - (16), (21) 식을 이용한 상보필터 연구 [1]
  - (16), (21) 식에 최대경사법을 이용한 연구 [2]
  - (16), (21) 식에 Extended Kalman Filter를 적용한 연구 [3]

참고

- Quaternion to Euler angle (ZYX)

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}atan2(q_0q_1 + q_2q_3, 0.5 - q_1^2 - q_2^2)\\ asin(2(q_0q_2 - q_1q_3))\\ atan2(q_1q_2 + q_0q_3, 0.5 - q_2^2 - q_3^2)\end{bmatrix} \quad (22)$$

- Euler angle (ZYX) to Quaternion

$$q = \begin{bmatrix}\cos(0.5\phi)\cos(0.5\theta)\cos(0.5\psi) + \sin(0.5\phi)\sin(0.5\theta)\sin(0.5\psi)\\ \sin(0.5\phi)\cos(0.5\theta)\cos(0.5\psi) - \cos(0.5\phi)\sin(0.5\theta)\sin(0.5\psi)\\ \cos(0.5\phi)\sin(0.5\theta)\cos(0.5\psi) + \sin(0.5\phi)\cos(0.5\theta)\sin(0.5\psi)\\ \cos(0.5\phi)\cos(0.5\theta)\sin(0.5\psi) - \sin(0.5\phi)\sin(0.5\theta)\cos(0.5\psi)\end{bmatrix} \quad (23)$$

참고문헌

[1] Roberto G. Valenti, Ivan Dryanovski and Jizhong Xiao. "Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs", Sensors 2015, 15, 19302-19330

[2] Sebastian O.H. Madgwick, Andrew J.L. Harrison, Ravi Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm", 2011 IEEE International Conference on Rehabilitation Robotics. 2011

[3] Angelo M. Sabatini, "Quaternion-Based Extended Kalman Filter for Determining Orientation by Inertial and Magnetic", IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, VOL. 53, NO. 7, JULY 2006

THANK YOU

Powerpoint is a complete presentation graphic package it gives you everything you need to produce a professional-looking presentation