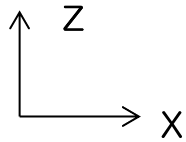
position and velocity estimation

- suppose an object is moving from left to right

navigation



  - if an inertial sensor unit is attached on the object, what is the measured acceleration

- accelerometer measurement equation

$$y_a = C(q)\tilde{g} + [a]_b + v_a$$

where $[a]_b$ is the external acceleration in the body coordnate system

- external acceleration in the navigation coordinate system

$$\hat{a}_n = C(\hat{q})'\hat{a}_b = C(\hat{q})'y_a - \tilde{g}$$

position and velocity estimation

- position (r) and velocity (v) estimation

$$\begin{aligned} \dot{\hat{v}} &= \hat{a}_n = C(\hat{q})' y_a - \tilde{g} \\ \dot{\hat{r}} &= \hat{v} \end{aligned}$$

- position and velocity estimation error

$$v_e = v_n - \hat{v}, \quad r_e = r_n - \hat{r}$$

- position error equation

$$\dot{r}_e = \dot{r} - \dot{\hat{r}} = v - \hat{v} = v_e$$

velocity error

- velocity error

$$
\begin{aligned}
\dot{v}_e &= \dot{v} - \dot{\hat{v}} \\
&= C'(q)a_b - C'(\hat{q})y_a + \tilde{g} \\
&= C'(q)(y_a - C(q)g - v_a) - C'(\hat{q})y_a + \tilde{g} \\
&= (C'(q) - C'(\hat{q}))y_a - C'(q)v_a \\
&= -(2[\bar{q}_e\times])C(\hat{q}))'y_a - C'(\hat{q})v_a - (2[\bar{q}_e\times]C(\hat{q}))'v_a \\
&\approx -2C'(\hat{q})[y_a\times]\bar{q}_e - C'(\hat{q})v_a
\end{aligned}
$$

where the following are used

$$
\begin{aligned}
C(q_e) &\approx I - 2(\bar{q}_e\times) \\
C(q) &\approx C(\hat{q}) - 2K(\bar{q}_e)C(\hat{q})
\end{aligned}
$$

Kalman filter equation

- Kalman filter state

$$x = \begin{bmatrix} q_e \\ r_e \\ v_e \end{bmatrix}$$

- state equation

$$\dot{x} = Ax + \begin{bmatrix} -\frac{1}{2}v_g \\ 0 \\ -C(\hat{q})' v_a \end{bmatrix} \qquad A = \begin{bmatrix} -[y_g \times] & 0 & 0 \\ 0 & 0 & I \\ -2C(\hat{q})'[y_a \times] & 0 & 0 \end{bmatrix}$$

- discretization

zero velocity updating

- If zero velocity interval is detected, the measurement update is done

$$0 = \hat{v} + v_e + v_v \Rightarrow 0 - \hat{v} = v_e + v_v$$

- H matrix $\quad z = 0 - \hat{v}$

$$H = \begin{bmatrix} 0 & 0 & I \end{bmatrix}$$

matlab code (ins.m)

```
% 25.7 cm lifting
load('lift.mat');

% 5 step walking
% load('walking.mat');

% 50 m walking
% load('longwalking1.mat');
% load('longwalking2.mat');

N = size(ya,2);
ra = 0.005;
rg = 0.001;
T = 0.01;
```

matlab code

```
% zero velocity detection
yanorm = zeros(1,N);
zerovel2 = zeros(1,N);
for i = 1:N
    yanorm(i) = norm(ya(:,i));
    if ( (yanorm(i) < 9.8+0.5) && (yanorm(i) > 9.8 -0.5) )
        zerovel2(i) = 1;
    end
end


zerovel = zerovel2;
M = 10;
for i = M+1:N-M
    if ( sum(zerovel2(i-M:i+M)) == 2*M+1 )
        zerovel(i) = 1;
    else
        zerovel(i) = 0;
    end
end
```

matlab code

```
qhat = zeros(4,N);
vhat = zeros(3,N);
rhat = zeros(3,N);

Q = diag([0.25 * rg,0.25 * rg,0.25 * rg,0,0,0,ra,ra,ra]);
A = zeros(9,9);
A(4:6,7:9) = eye(3);

qhat(:,1) = quaternionya(ya(:,1));
gtilde = [0 ; 0 ; 9.8];
H = zeros(3,9);
H(:,7:9) = eye(3);
R = 0.001 * eye(3);
P = diag([0.001 0.001 0.001 0 0 0 0 0 0 ]);
oldomega4 = zeros(4,4);
```

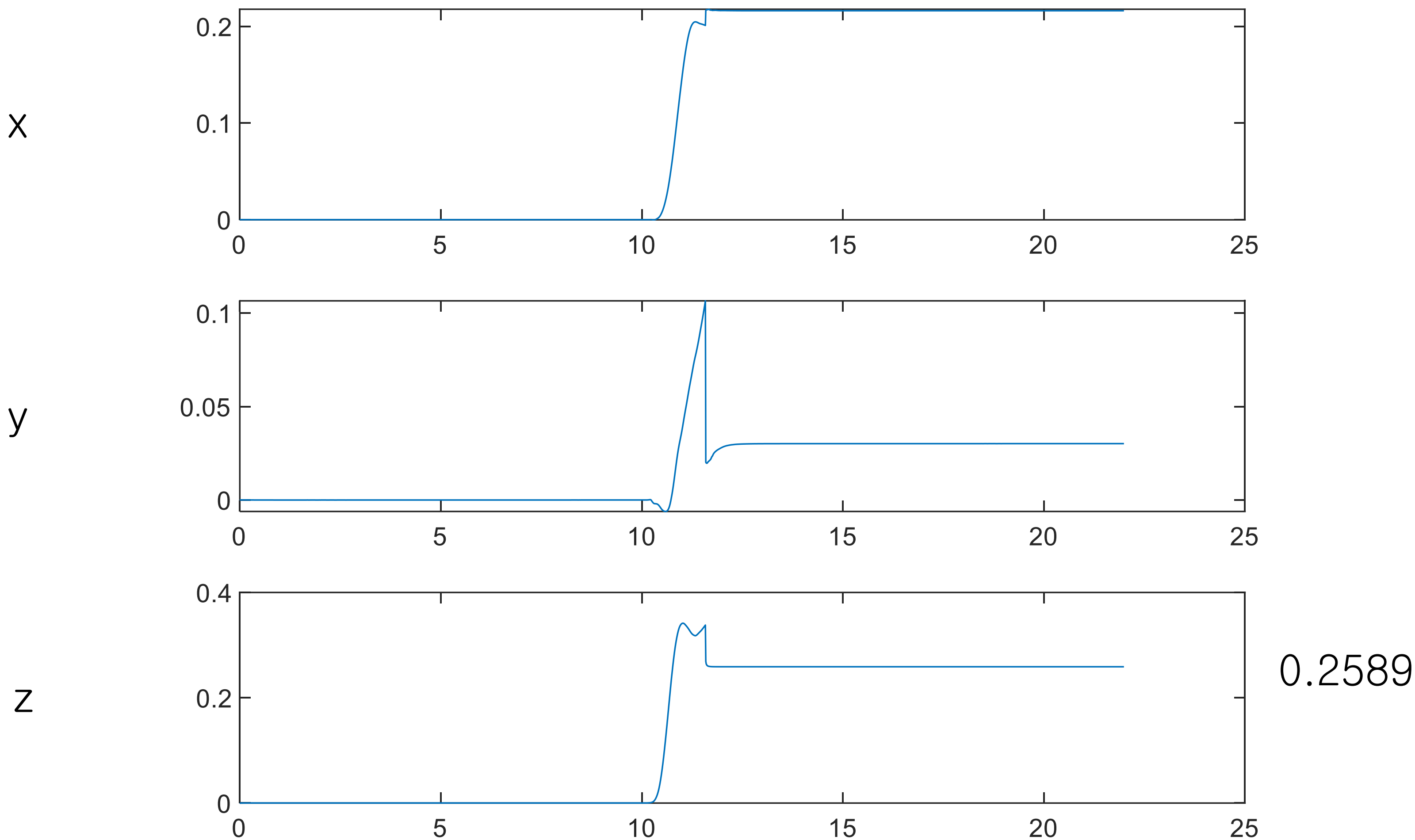
matlab code

```
for i = 2:N
    Cq = quaternion2dcm( qhat(:,i-1) );

    A(1:3,1:3) = -vec2product(yg(:,i-1));
    A(7:9,1:3) = - 2 * Cq' * vec2product(ya(:,i-1));
    Qd = Q * T + (T^2 / 2) * A * Q + (T^2/2) * Q * A';

    dA = eye(9) + A * T + A * A * T^2 / 2;
    P = dA * P * dA' + Qd;
    omega4 = compute_44(yg(:,i-1));
    qhat(:,i) = ( eye(4) + 0.75 * omega4 * T - 0.25 * oldomega4 * T - (1/6) *
norm(yg(:,i-1))^2 * T^2 * eye(4) - (1/24) * omega4 * oldomega4 * T^2  -
(1/48)*norm(yg(:,i-1))^2 *omega4 * T^3) * qhat(:,i-1);
    qhat(:,i) = qhat(:,i) / norm(qhat(:,i));
    oldomega4 = omega4;
    vhat(:,i) = vhat(:,i-1) +  0.5* T * ( quaternion2dcm(qhat(:,i-1))' * ya(:,i-1)
+ quaternion2dcm(qhat(:,i))' * ya(:,i) )  -  T * [ 0 ; 0 ; 9.8];
    rhat(:,i) = rhat(:,i-1) + 0.5* T * (vhat(:,i) + vhat(:,i-1));
```

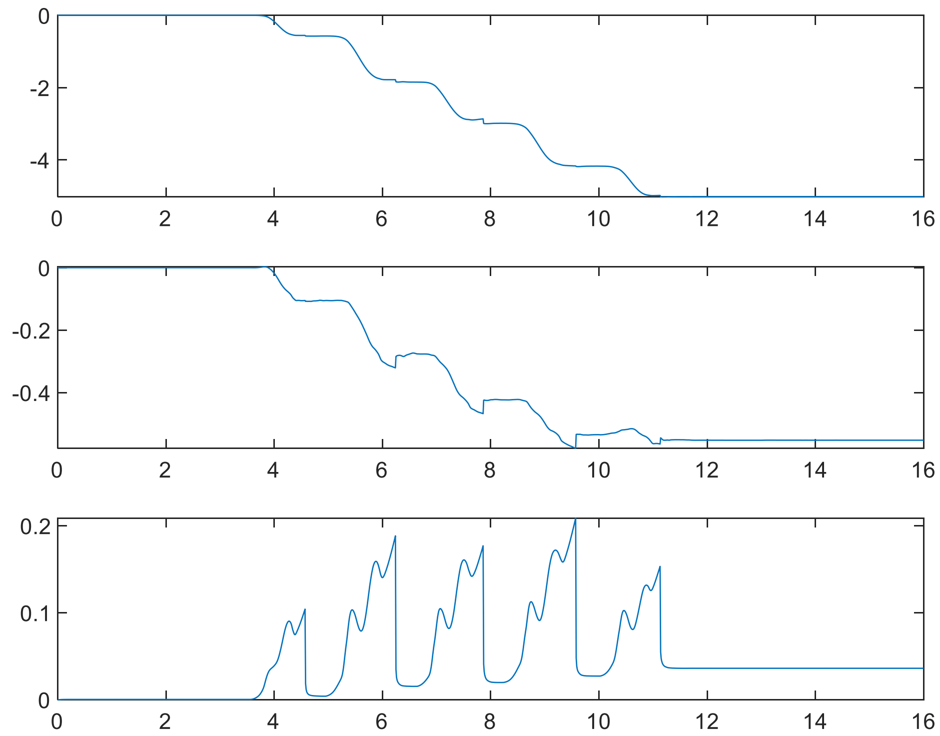
matlab code

```
if ( zerovel(i) == 1 )
        z = zeros(3,1) - vhat(:,i);
        K = P * H' * inv(H * P * H' + R);
        x = K * z;

        P = ( eye(9) - K * H) * P;
        P = 0.5 * (P + P');

        vhat(:,i) = vhat(:,i) + x(7:9);
        rhat(:,i) = rhat(:,i) + x(4:6);

        qe = [ 1 ; x(1:3) ];
        qhat(:,i) = quaternionmul(qhat(:,i),qe);
        qhat(:,i)=  qhat(:,i) / norm(qhat(:,i));
    end
end

plotsensor(rhat)
```

25.7cm lifting (lift.mat)

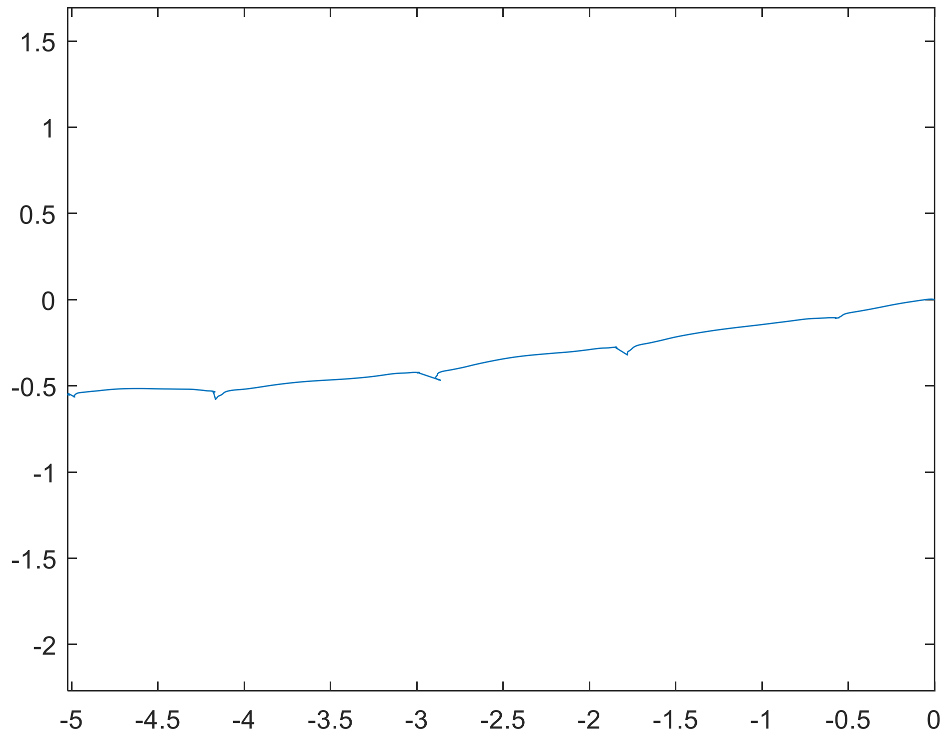


5 step walking (walking.mat)

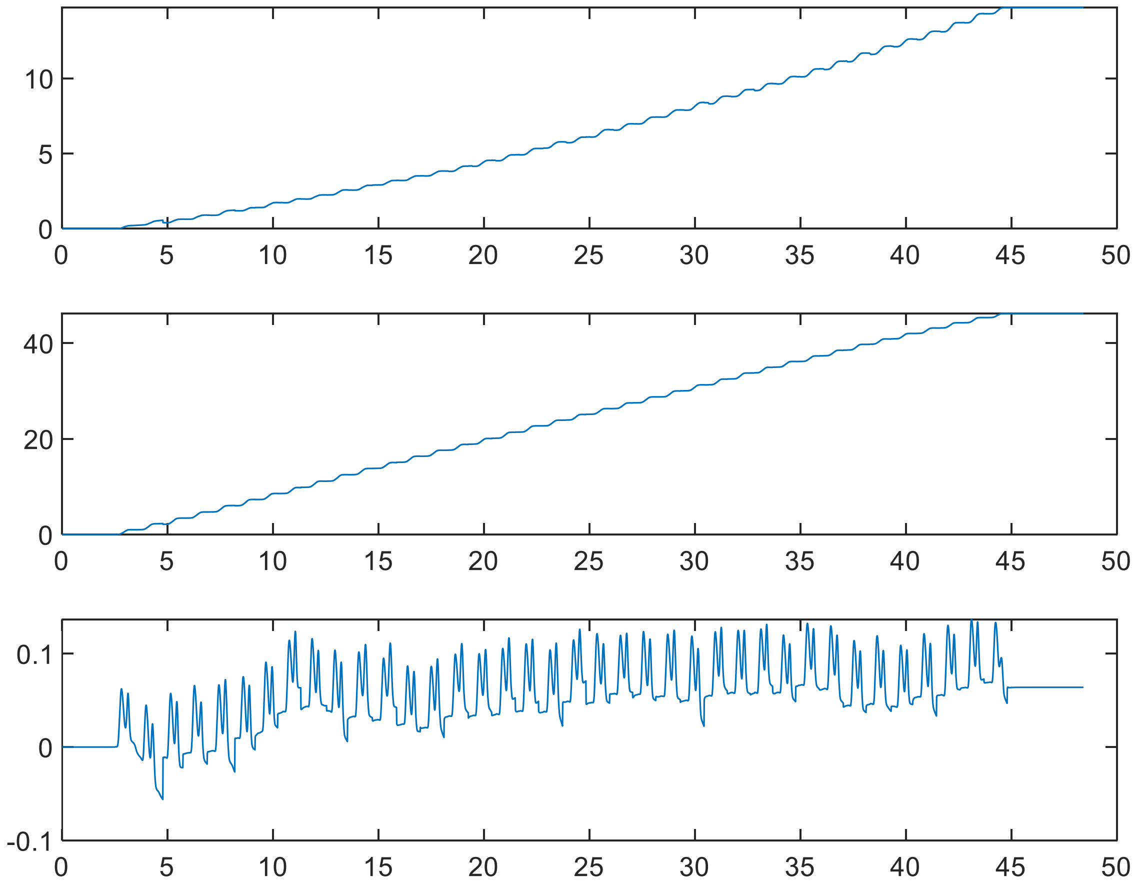


5 step walking (walking.mat)

```
plot(rhat(1,:),rhat(2,:)); axis equal
```

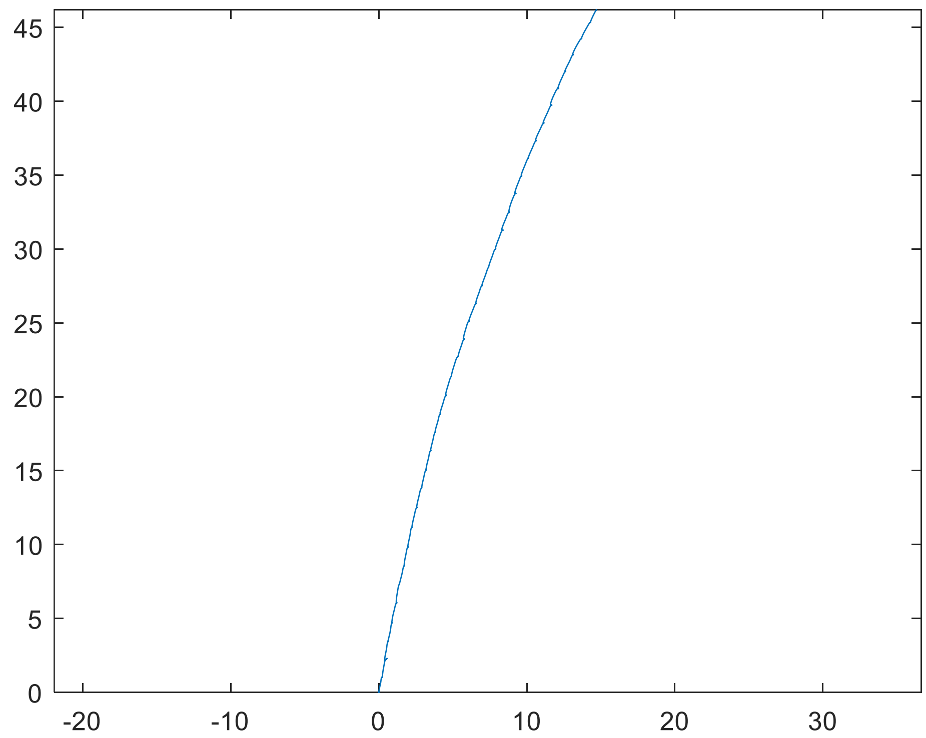


long walking (longwalking2.mat)



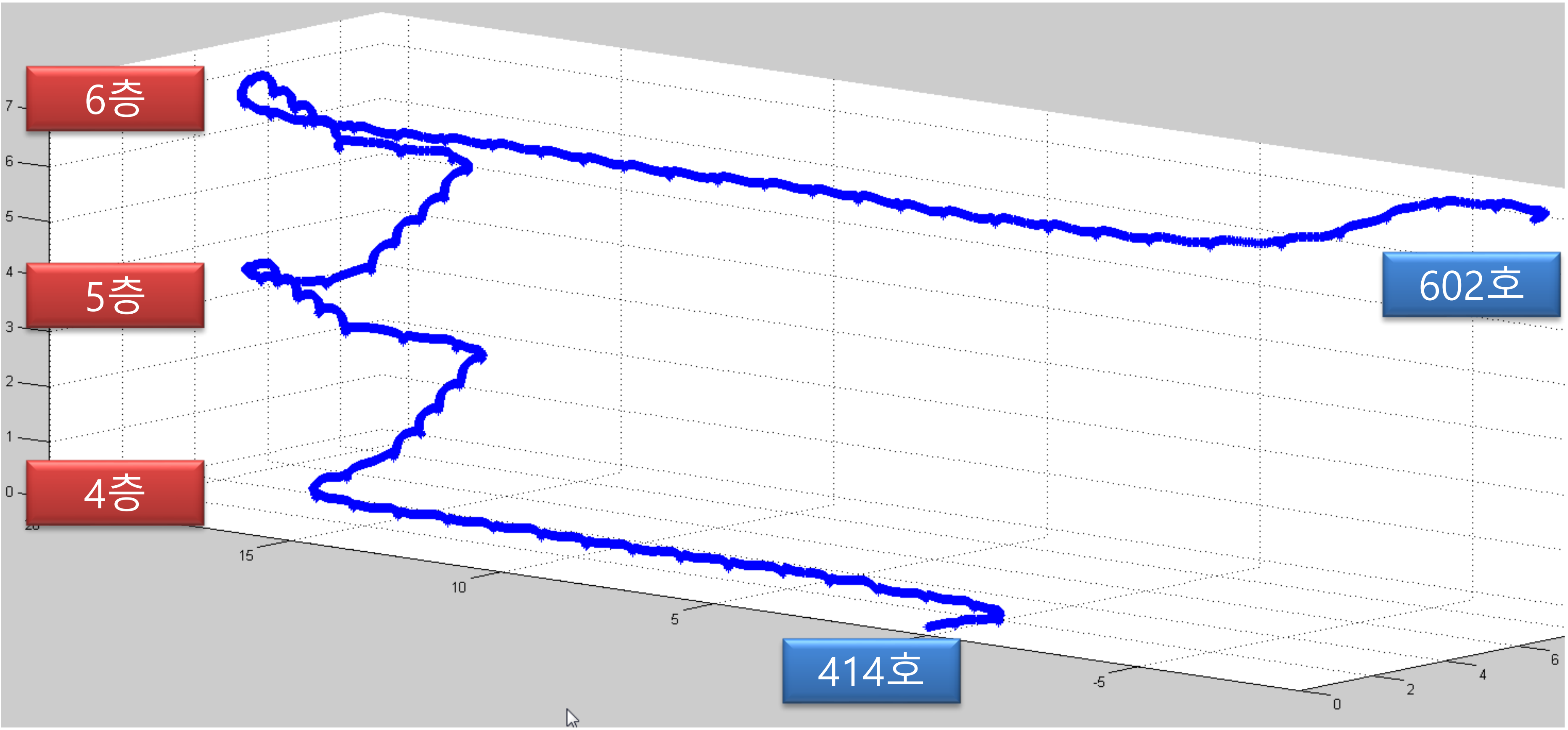
long walking (longwalking2.mat) (50m walking)

```
plot(rhat(1,:),rhat(2,:)); axis equal
norm( rhat(1:2,N) )
```



48.4761

indoor navigation



울산대학교
UNIVERSITY OF ULSAN

term project (replacing the final exam)

- project 1: modification of ins.m
  - during zero velocity updating, make the height zero

$$0 = \hat{r} + r_e + v_r \Rightarrow 0 - \hat{r} = r_e + v_r$$

  - now the number of measurement is 6 (from 3)
  - test the program with 'longwalking1.mat'

- project 2: modification of ins.m
  - modify the zero velocity detection algorithm so that it can work with 'running.mat'. Find the total running length
  - hint: try to use gyroscope data

- project 3: walking distance estimation
  - Use IMUs in your smartphone to measure walking distance (about 30m)

Presentation

- matlab code, power point file submission
  - power point : explain your idea and include the main results (graph)
  - Picture of your smartphone attached on a foot

- Presentation date: Dec 10