EKF with Unicycle Model Equations

**UNICYCLE MODEL EQUATIONS**

function Xdot = unicycle\_ode(t,f,v,w) % just to get the simulated ground truth

% Process model

Xdot = zeros(3,1);

Xdot(1) = v\*cos(f(3));

Xdot(2) = v\*sin(f(3));

Xdot(3) = w; % usually w but try function of time sin(0.5\*t)

end

function f = unicycle(f, v, w, dt)

% Process model for KF

f(1,1) = f(1,1) + v\*cos(f(1,3))\*dt;

f(1,2) = f(1,2) + v\*sin(f(1,3))\*dt;

f(1,3) = f(1,3) + w\*dt;

end

function Fk = unicycle\_linearized(f,v, dt)

% Jacobian of process model

Fk = [ 1 0 -v\*sin(f(1,3))\*dt;

0 1 v\*cos(f(1,3))\*dt;

0 0 1];

end

**MEASUREMENT MODEL EQUATIONS (not sure about)**

function IMUsim = imu(w, v, gt, dt) % params will be changed to imu data

% Initialize IMU data containers

pos = zeros(2, 1);

head = 0;

positionStore = zeros(1,2);

headStore = zeros(1);

for i=1:length(gt)

% calculate heading from gyro

orientationChange = w \* dt;

head = head + orientationChange;

headStore = [headStore; head];

% convert input velocities to acceleration

a = [0 ;0];

% transform acceleration data to global coordinates

Rz = [cos(head) -sin(head);

sin(head) cos(head)];

aT = Rz \* a; % 2x2 ' 2x1

% already have velocity from kinematic model inputs

% don't feel like going to acc then back to vel

% so transform vel to global coordinates instead

vel = [v\*cos(head); v\*sin(head)];

% vT = Rz \* vel;

positionChange = vel \* dt;

pos = pos + positionChange;

positionStore = [positionStore;

pos(1), pos(2)];

end

IMUsim = [positionStore(:,1), positionStore(:,2), headStore];

end

function Hjacobian = imu\_linearized(x)

% not correct for the actual model, but in simulation the values are

% directly accesible I guess?

Hjacobian = eye(3,3); % doesn't seem right

end

**EKF EQUATIONS**

function ekf(dt, z, x0, P0, v, w, gt)

n = size(x0, 1);

m = size(z, 1);

T = size(z, 1);

xk = zeros(T,n);

xk\_ = xk;

P = zeros(n,n,T);

P\_ = P;

zModel = zeros(T,n);

f = @(x) unicycle(x, v, w, dt);

Fk = @(x) unicycle\_linearized(x, v, dt);

Wk = @(x,dt) [cos(x(3))\*dt 0;

sin(x(3))\*dt 0;

0 dt];

Qk = [0.05 0;

0 0.05];

% Q = Wk\*Qk\*Wk';

Q = @(x,dt) [cos(x(3))\*dt 0; sin(x(3))\*dt 0; 0 dt]\*...

[0.05 0;

0 pi/8]\*[cos(x(3))\*dt 0; sin(x(3))\*dt 0; 0 dt]';

h = imu(w, v, gt, dt);

Hjacobian = imu\_linearized;

Rk = [0.5 0 0;

0 0.5 0;

0 0 pi/4];

k0=1;

kF=T;

xk\_(k0,:) = x0;

P\_(:,:,k0) = P0;

for k=k0:kF

[xk(k,:), P(:, :, k, 1)] = update(xk\_(k,:), P\_(:,:,k,1), Rk, z(k,:), h(k,:), Hjacobian);

[xk\_(k+1,:), P\_(:, :, k+1, 1)] = predict(xk(k,:), P(:,:,k,1), Q(xk(k,:),dt), f, Fk(xk(k,:)));

zModel(k,:) = h(k,:); % to get measurement model output

% but what is the input to calculate

% measurement model values?

% should they be theoretical values given

% by control inputs?

end

show\_plots(gt, z, xk, zModel);

end

**UPDATE STEP**

function [xk, P] = update(xk\_, P\_, R, z, h, Hjacob)

% get size of states

[~,n] = size(xk\_);

% handle Jacobian of measurement model

Hk = Hjacob;

% Innovation Covariance

S = Hk\*P\_\*Hk' + R;

% Measurement Model

zh = h;

% Innovation

innovation = z - zh;

% Kalman Gain

K = P\_\*Hk' / (S);

% Update states and covariance

xk = xk\_' + K\*innovation';

P = (eye(n) - K\*Hk) \* P\_;

end

**PREDICTION STEP**

function [xk\_, P\_] = predict(xk, P, Qk, f, Fk)

% States

xk\_ = f(xk);

% Covariance

P\_ = Fk\*P\*Fk' + Qk;

end

**OUTPUTS**

Diagram

Description automatically generated

Diagram

Description automatically generated