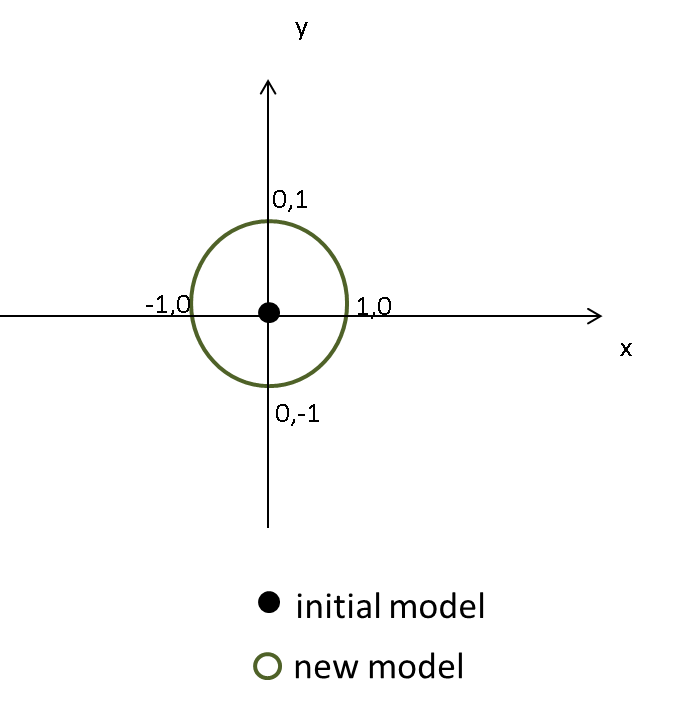
Ch3

4.

(a)



(b)

syms x y t d % parameters (x,y,theta), move d units

f = [x+d\*cos(t); y+d\*sin(t); t]; % transition function

v = [x, y, t];

% Derive Jacobian

G = jacobian(f, v)

G =

[ 1, 0, -d\*sin(t)]

[ 0, 1, d\*cos(t)]

[ 0, 0, 1]

Sig = [0.01, 0 , 0;

0 , 0.01 , 0;

0 , 0 , 10000];

x = 0; y = 0; d = 1; t = 0; R = 0;

G = [1, 0, -d\*sin(t);

0, 1, d\*cos(t);

0, 0, 1];

% Calculate 

Sigb = G\*Sig\*G' + R

Sigb =

1.0e+004 \*

0.0000 0 0

0 1.0000 1.0000

0 1.0000 1.0000

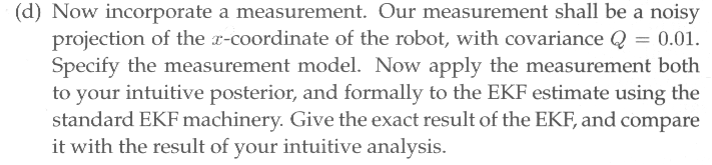
(c)



y

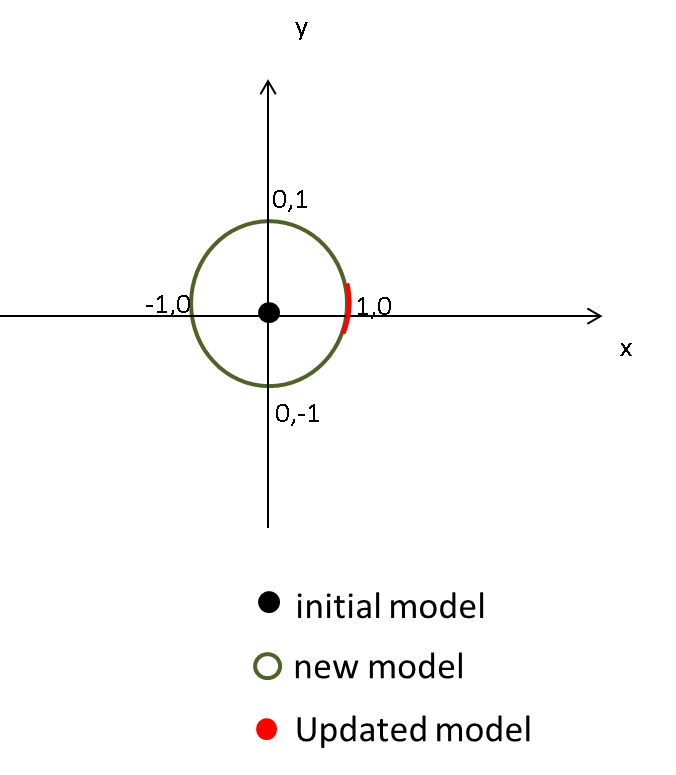
x

(d)



The measurement model 

By intuition:



Using EKF:

H=[1, 0, 0];

K = Sigb\*H' \* inv(H\*Sigb\*H'+Q);

z = 1;

% Calculate 

u = ub + K\*(z-ub(1));

u =

1

0

0

% Calculate 

Sig = (eye(3,3)-K\*H)\*Sigb;

Sig =

1.0e+004 \*

0.0000 0 0

0 1.0000 1.0000

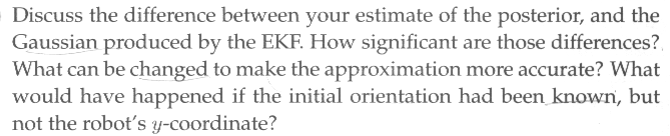
0 1.0000 1.0000



y

x

(e)



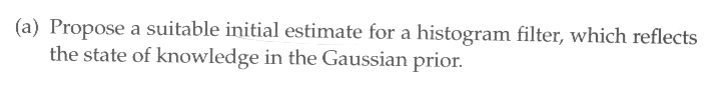
My posterior seems to converge after the measurement is used for update, while the EKF still have a great uncertainty along y axis. At the prediction step of EKF, the Gaussian estimate didn’t produce a correct presentation for the uncertainty since the possible robot pose should be a circle centered at the initial point. Due to linearization on *g* the EKF prediction have a great linearization error on the nonlinear function on .

If the orientation had been known, but not the robot’s y-coordinate, from my point of view, the EKF prediction will be more accurate, since y is a linear variable for *g.* Therefore the linearization error is smaller. However, the update won’t have much effect because the measurement is the projection of x coordinate of robot pose. This won’t help to lower the uncertainty on y axis.

Ch4

2.

(a)





x

y

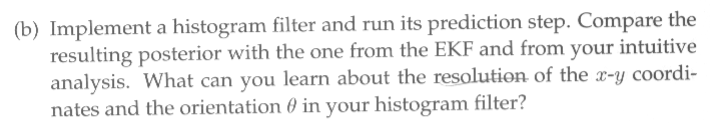
% a uniform distribution of probability at (x=0,y=0,theta)

for ti=t\_axis

Px(ind(0,0,ti)) = 1/t\_len;

end

(b)



x

y

x

y

1

(x, y resolution=0.05, =pi/180) (x, y resolution=0.2, =pi/45)

% histogram filter

eta = 0;

for xi=-1:x\_res:1

for yi=-1:y\_res:1

for ti=t\_axis

% if succ

%[xi yi ti]

try

Pxb(ind(xi+cos(ti),yi+sin(ti),ti)) = Pxb(ind(xi+cos(ti),yi+sin(ti),ti)) + Px(ind(xi,yi,ti));

eta = eta + Px(ind(xi,yi,ti));

catch

end

end

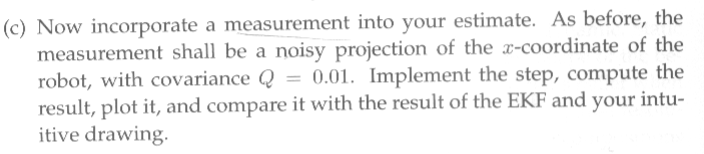
end

end

Pxb = Pxb / eta;

The distribution produced is more like the intuitive result than EKF. The result reflects more like real distribution. If we use raise the resolution, then the prediction result will reflect more precise distribution after the transition but the computation speed will get slower. If the resolutions of each axis double, the overall runtime will be 8 times.

(c)



x

y

1,0

x

y

1,0

(x, y resolution=0.05, =pi/180) (x, y resolution=0.2, =pi/45)

eta = 0;

for xi=-1:x\_res:1

for yi=-1:y\_res:1

for ti=t\_axis

% if succ

%[xi yi ti]

try

pp = Px(ind(xi,yi,ti))\*normpdf(xi+cos(ti),1,0.1);

%normpdf(xi+cos(ti),3,0.1)

Pxb(ind(xi+cos(ti),yi+sin(ti),ti)) = Pxb(ind(xi+cos(ti),yi+sin(ti),ti)) + pp;

eta = eta + Px(ind(xi,yi,ti)) + pp;

catch

end

% end

end

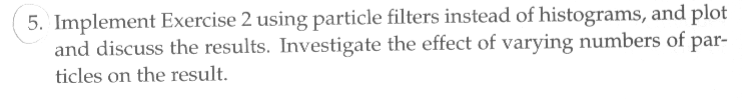
end

end

Pxb = Pxb / eta;

The updated distribution produced is more like the intuitive result than EKF.

5.





Number of particle = 1000

function ch4\_2\_c

M = 1000;

oldChi = [0.1 .\* randn(M,1) 0.1 .\* randn(M,1) 100 .\* randn(M,1)];

Chib = zeros(0,4);

Chi = zeros(0,4);

wsum = 0;

for m = 1:M

% sample x = p(x|u,x);

x = oldChi(m,1);

y = oldChi(m,2);

t = oldChi(m,3);

xt = [x+cos(t) , y+sin(t), t];

wt = normpdf(1,x+cos(t),0.1);

wsum = wsum + wt;

Chib = [Chib; xt, wsum ];

end

for m=1:M

i = bsearch(Chib, rand(1,1)\*wsum);

Chi = [Chi; Chib(i,:)];

end

hold on

plot(oldChi(:,1),oldChi(:,2),'\*','color','k');

plot(Chib(:,1),Chib(:,2),'\*','color','c');

plot(Chi(:,1),Chi(:,2),'\*','color','b');

end

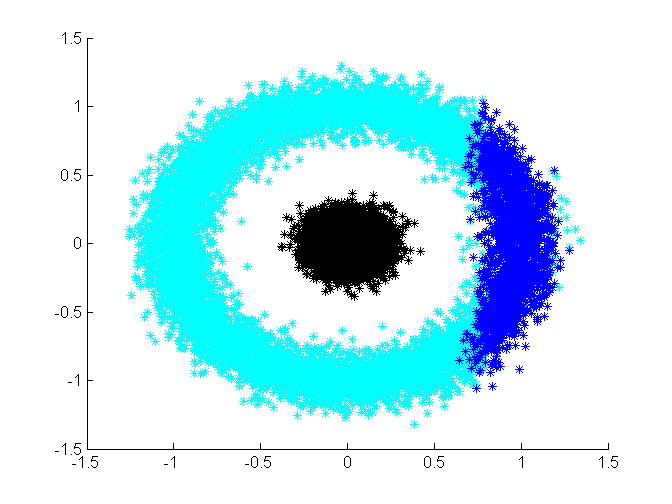
The result is much similar to the intuitive result unlike the EKF prediction and update. The runtime is much faster than using histogram because histogram is a dense calculation on grids but particle filter only predict and update on the particles.



Number of particle = 100



Number of particle = 1000



Number of particle = 10000

To compare the effect of different particle numbers, we generate the above figures. From the above figures, we can conclude that the more particles we apply the more accurate model for to describe the posterior distribution. The time complexity is , where is for binary search.