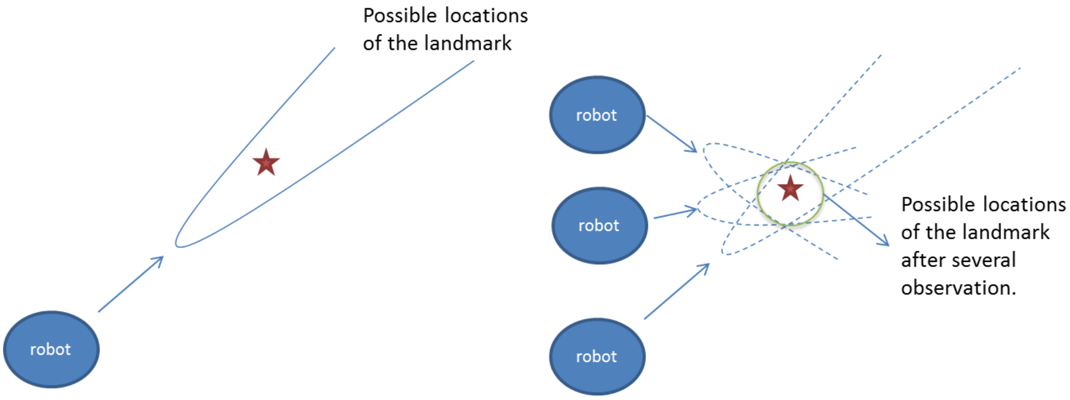
AMR HW8
R99922070 俞冠廷
2011/6/11

(a) Chapter 10, Exercise 2

Why initialization is a problem in bearing only SLAM?

The reason why initialization of landmark is an important issue is because there is no direct depth information from the bearing-only sensor. Therefore, the possible location of a newly observed landmark lies in a cone shape region. After several observation and updates, the region will converge into a closed region. However, traditional Gaussian distribution in XYZ space cannot provide a unified representation of these two kinds of probability distribution.

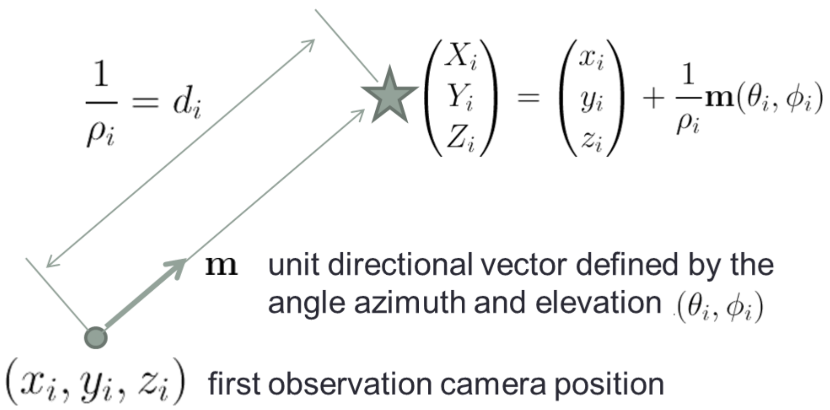


Landmark Initialization

Here I will describe the Inverse depth method proposed by Civera *et al.* It is a unified representation can initialize the landmark without delay.

On first observation, 6 parameters are used to represent the landmark state:

$(x_i \quad y_i \quad z_i \quad \theta_i \quad \phi_i \quad \rho_i)$, where each terms are described in the figure below.



The camera pose can be easily obtained. We can also compute $(\theta_i \quad \phi_i)$ using the camera parameters. For the inverse depth, ρ_i is assigned a general Gaussian prior in inverse depth that encodes probabilistically the fact that the point has to be in front of the camera.

$$\hat{\rho}_0 = 0.1, \sigma_\rho = 0.5$$

which means

Initial inverse depth confidence region: $[1.1, -0.9]$

Initial depth confidence region: $[\frac{1}{1.1}, \infty] \cup [-\infty, \frac{1}{-0.9}]$

With the probabilistic framework (e.g. EKF), we can keep on updating the state. After the uncertainty reduced, the possible region of landmark location will form a closed space.

(b) Chapter 12, Exercise 4

Why?

Because we assume that the variable x_t does not depend on the passive features m^- if we know the active features m^0 and m^+. Therefore, we can set m^- to arbitrary value without affecting the conditional posterior.

What would be the update equation if these features would not be conditioned away?

If these features are not conditioned away, we should do a full EIF update (without multiplying the F-matrix).

Would the result be more accurate or less accurate?

The result would be more accurate if we conduct a full EIF update. On the other hand, if we perform the SEIF update without conditioning m^- away, the result should be less accurate because some information of some link is arbitrarily ignored.

Would the computation be more or less efficient?

Less efficient. If we conduct the full EIF update, the efficiency is similar to EKF.

(c) Chapter 13, Exercise 1.

EKF

1. Using Gaussian distribution to represent the states makes the computation very fast.
2. No need to sample among the state distribution.
3. Capable to consider uncertainty over high dimensional state. In contrast, the number of particles required by particle filter increase exponentially.

GraphSLAM

1. Solve the full SLAM problem. It calculates posteriors over the full robot path along with map.
2. Consider data association with probability.
3. Incorporate sparsification idea by using information matrix.

FastSLAM

1. Multiple hypothesis tracking through per-particle data association.
2. Use sampling on highly non-linear portions of state space can avoid linearization error

using EKF.

3. Particle filter is generally easier to implement.

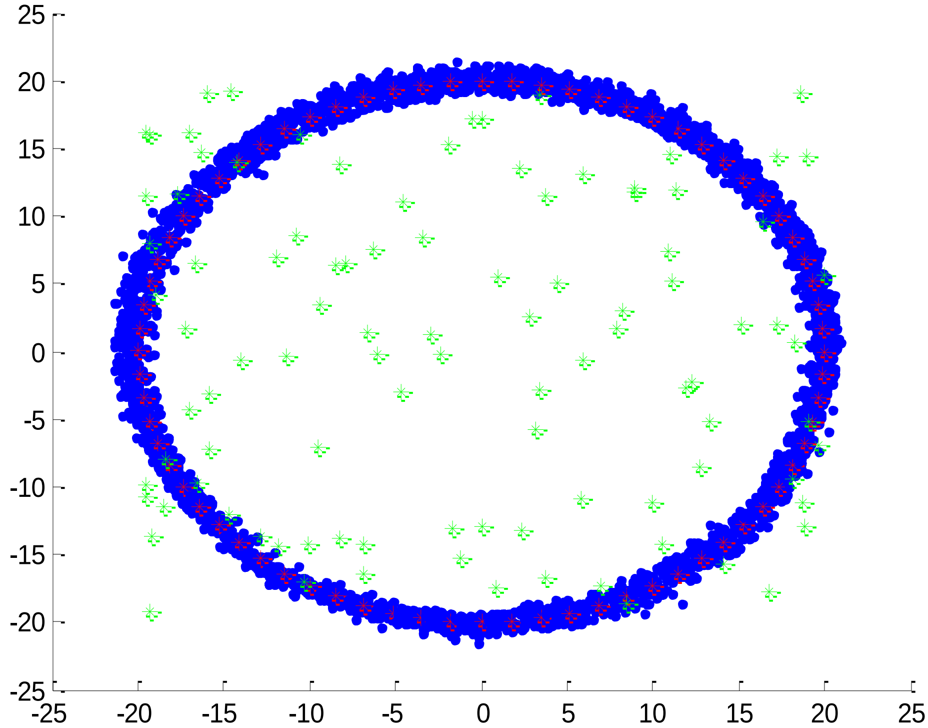
(c) Chapter 13, Exercise 7.

Fast slam simulation

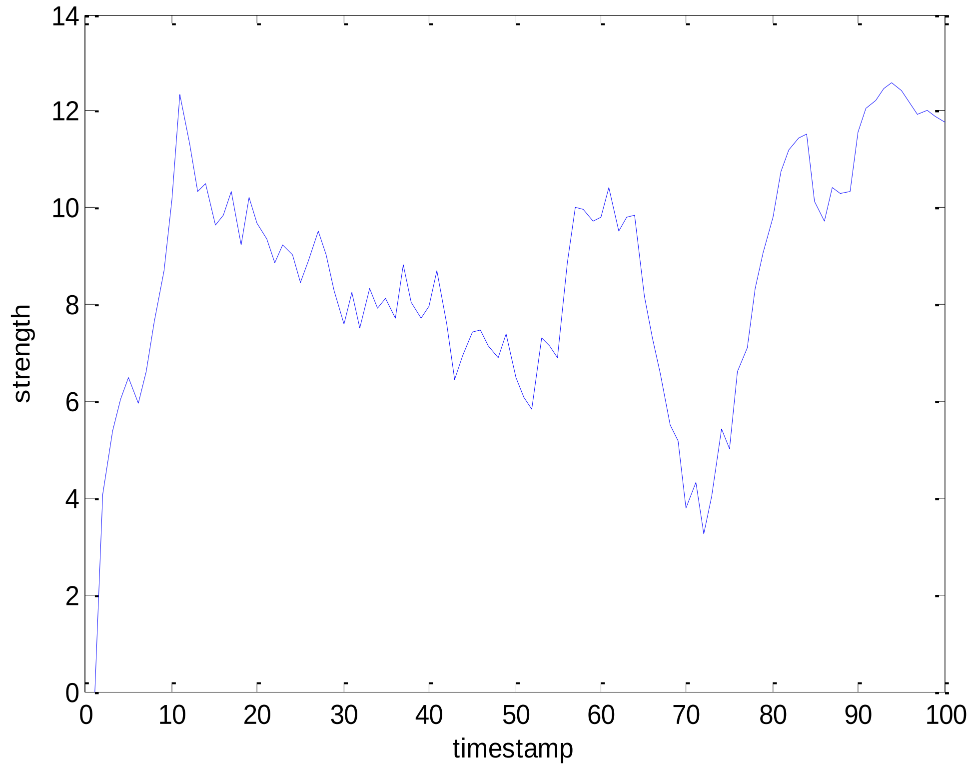
Red: ground truth

Blue: particles

Green: landmarks



The following figure shows the strength of the correlations w.r.t timestep. At the beginning, the strength increases with time. This means the uncertainty increases. At time=72, the robot back to the starting point (20, 0), the strength reach the nadir. The situation is similar to the decrease of variance when closing a loop in EKF.




```
function fastslamProcedure
    global Y;
    global setting;
    global landmark;
    global x_groundtruth;

    setting.nstep = 100;
    setting.nparticle = 100;
    setting.zrange = 10000;
    setting.nlandmark = 100;
    setting.Usigma = 0.1;
    setting.Zsigma = 0.1;
    %setting.Usigmath = 0.01;
    %setting.R = [0.1 0 0; 0 0.1 0; 0 0 0.01];
    setting.R = [0.5 0; 0 0.5];
    initLandmark();
    initParticle(); %draw init samples
    x_groundtruth = generate_x_groundtruth();
```

```
    for i=2:setting.nstep
        [u z c] = simulateOneStep(x_groundtruth,i);
        %for j=1:length(c)
            Y{i} = FastSlam(z,c,u,Y{i-1});
        %end
    end
    drawY();
    fitGaussian();
end
function fitGaussian()
global Y setting

f = zeros(0,2);
for kk = 1:setting.nstep
    data = zeros(setting.nparticle,2+setting.nlandmark*2);
    for i=1:setting.nparticle
        data(i,1:2) = Y{kk}.p{i}.xt;
        for j=1:setting.nlandmark
            if(Y{kk}.p{i}.landmark{j}.init==1)
                data(i,2+(j-1)*2 : 2+(j-1)*2+1) =
Y{kk}.p{i}.landmark{j}.mu;
            end
        end
    end

    avg = mean(data);
    Cov = zeros(2+setting.nlandmark*2);
    for i=1:(2+setting.nlandmark*2)
        for j=1:(2+setting.nlandmark*2)
            for k=1:setting.nparticle
                Cov(i,j) = Cov(i,j) +
(data(k,i)-avg(i))*(data(k,j)-avg(j));
            end
        end
    end
    Cov = Cov/setting.nparticle;
    f = [f;kk norm(Cov)];
end
```

```
figure
plot(f(:,1),f(:,2));
end

function drawY()
global Y  x_groundtruth landmark
    figure
    hold on;
    for i=1:length(Y)
        for j=1:length(Y{i}.p)
            plot(Y{i}.p{j}.xt(1),Y{i}.p{j}.xt(2), 'b.');
        end
        pause(0.1);
        plot(x_groundtruth(i,1),x_groundtruth(i,2), 'r*');
    end
    for i=1:size(landmark,1);
        plot(landmark{i}(1),landmark{i}(2),'g*');
    end
end

function Yt = FastSlam(z, c, u, Yt_1)
    global setting;
    Yt.p = cell(length(Yt_1.p),1);
    for k=1:length(Yt_1.p)  % loop over particles
        pstate = Yt_1.p{k};
        xt = pstate.xt + u+ randn(1,2)*setting.Usigma*2;

        Yt.p{k}.landmark = pstate.landmark;
        Yt.p{k}.w = 0;
        for j=1:length(c)
            j_lmk = c(j);
            if Yt.p{k}.landmark{j_lmk}.init == 0 % j never seen before
                mu = z(j,1:2) + xt;   % initialize mean
                invG = invgp(xt, mu);
                Cov = invG*setting.R*invG';
                Yt.p{k}.landmark{j_lmk}.mu = mu;
                Yt.p{k}.landmark{j_lmk}.Cov = Cov;
                Yt.p{k}.landmark{j_lmk}.init = 1;
```

```
            Yt.p{k}.w = Yt.p{k}.w+0.9;
        else
            Cov = Yt.p{k}.landmark{j_lmk}.Cov;
            mu = Yt.p{k}.landmark{j_lmk}.mu;
            zt = z(j,1:2);
            zh = mu-xt; %g(mu, xt);
            G = gp(xt,mu);
            Q = G'*Cov*G + setting.R;
            K = Cov*G*Q;
            mu = mu + (K* (zt-zh)')';
            Cov = (eye(2) - K*G') * Cov;

           Yt.p{k}.landmark{j_lmk}.Cov =  Cov;
           Yt.p{k}.landmark{j_lmk}.mu = mu;
           %Yt.p{k}.w = exp(-(zt-zh)*(zt-zh)');
           Yt.p{k}.w = Yt.p{k}.w+(1/sqrt(det(2*pi*Q))) *
exp(-0.5*(zt-zh)*inv(Q)*(zt-zh)');
        end

        Yt.p{k}.xt = xt;
    end
end

Yttmp = Yt;
corr = zeros(setting.nparticle, 1);
corr(1) = Yt.p{1}.w;
for i=2:setting.nparticle
    corr(i) = corr(i-1) + Yt.p{i}.w;
end
RAND = rand(setting.nparticle,1)*corr(setting.nparticle);
for i=1:setting.nparticle
    j=0;
    for j=1:setting.nparticle
        if j==1, ub = 0;
        else ub = corr(j-1);
        end
        if RAND(i) < corr(j) && RAND(i) >= ub;
            break;
```

```
            end
        end
        Yt.p{i} = Yttmp.p{j};
    end
end

function x = gp(xt, mu)
    x =  [1 0; 0 1];

end

function x = invgp(xt, mu)
    x =  [1 0; 0 1];
end
function x_groundtruth = generate_x_groundtruth()
global setting
    x_groundtruth = zeros(setting.nstep, 2);
    for i=1:setting.nstep
        x_groundtruth(i,1:2) = [cos((i-1)*pi/36), sin((i-1)*pi/36)] * 20;
        %x_groundtruth(i,3) = i*pi/36;   % 5 degree each step
    end
end

function [u,z,c] = simulateOneStep(x_groundtruth, index)
    global landmark;
    global setting;
    u = x_groundtruth(index,:) - x_groundtruth(index-1,:)+
[randn(1,2)*setting.Usigma]; % R
    c = zeros(0,1);
    z = zeros(0,2);
    for i=1:length(landmark)
        dist = norm( landmark{i} - x_groundtruth(index,1:2) );
        if(dist < setting.zrange) %% visible
            z = [z; (landmark{i} - x_groundtruth(index,1:2))] ;
            c = [c; i];
        end
    end
end
```

```
function initLandmark()
    global setting landmark;
    landmark = cell(setting.nlandmark,1);
    rnd = rand(setting.nlandmark,2);
    for i=1:setting.nlandmark
        landmark{i} = rnd(i,:)*40-[20,20];
    end

end
function initParticle()
    global Y;
    global setting;
    global landmark;
    Y = cell(setting.nstep,1);
    Y{1}.p = cell(setting.nparticle,1);
    initLandmarkSigma = [0.1^2 , 0, 0, 0.1^2];
    for k=1:setting.nparticle
        Y{1}.p{k}.xt = [20 0];

        Y{1}.p{k}.landmark = cell(setting.nlandmark,1);
        Y{1}.p{k}.w = 1/setting.nparticle;
        for i=1:setting.nlandmark
            Y{1}.p{k}.landmark{i}.init = 0;
            %Y{1}.p{k}.landmark{i}.u = landmark{i};
            %Y{1}.p{k}.landmark{i}.Sigma =initLandmarkSigma;
        end
    end

end
```