Probabilistic Robotics Homework 4

Ujjwal Sharma and Arjan van der Linden, 10982108

October 2017

Task 1 - Localisation

For the first part, the uncertainty in the locations has to be visualized. We used the function error_ellipse [1] to plot the location and the covariance for every observation (figure 1).

Listing 1: Localization using Extended Kalman Filter

```

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Extended Kalman Filter
%   by J rgen Sturm, Tijn Schmits, Arnoud Visser
%   April 2008
%
% Based on:
%
% Wolfram Burgard's
% http://ais.informatik.uni-freiburg.de/teaching/ss07/robotics/slides/
% --> 09.pdf
%
% Thrun's
% http://robots.stanford.edu/probabilistic-robotics/ppt/slam.ppt
%
% Dataset dlog.dat provided by Steffen Gutmann, 6.5.2004
% http://cres.usc.edu/radishrepository/view-one.php?name=comparison_of_self-localization_methods_continued
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

[1] https://nl.mathworks.com/matlabcentral/fileexchange/4705-error-ellipse

```

% ------------------------------------------------------------------------ init
%
clear;

% N is number of observations in dlog.dat

logfilename = 'dlog_firstmark.dat'; N = 758;
%logfilename = 'dlog_secondmark.dat'; N = 1159;
%logfilename = 'dlog_thirdmark.dat'; N = 1434;


% expected user input noise
u_err = .15;
M = u_err*eye(2);

% expected robot location noise
m_err = .1;
Q = m_err*eye(2);

%Custom, for plotting the position
figure
%--------------------------

% --------------------------------------------------------------- data creation
%
% true robot position at t = 1
xt(:,1) = [0 0 0]'; dim = 3;  % x = [x y angle]'

% user input at t = 1
u(:,1) = [0 0]';              % u = [speed delta_angle]'

% Landmark locations
L2006 = [20  20 -20 -20;...
     20 -20  20 -20];

% You also need the following information about the landmark positions:
% cyan:magenta -1500 -1000 magenta:cyan -1500 1000 magenta:green 0 -1000 green:magenta 0 1000 yellow:magenta 1500 -1000 magenta:yellow 1500 1000
% 0 -> green 1 -> magenta 2 -> yellow 3 -> blue
```

```matlab
L = [-15 -15 0 0 15 15;-10 10 -10 10 -10 10];
LID = [3 1 1 0 2 1;1 3 0 1 1 2];
U = M;          % user input noise (set to be equal to expected input noise)

angle = 0;

logfile = true;

if ~logfile

    for t=2:N

        % fabricate user input
        u(2,t) = randn;
        if abs(u(2,t)) > 0.4 %  P(steering) = 0.4
            u(2,t) = 0;
        end
        u(1,t) = .5*(1 - u(2,t)/0.4); % high delta_angle --> low speed

        % create noisy user input
        un = U*randn(2,1) +u(:,t);

        % calculate true robot position t+1
        xt(:,t) =  [xt(1,t-1)+ un(1)*cos(xt(3,t-1)) ; ...
            xt(2,t-1)+ un(1)*sin(xt(3,t-1)) ; ...
            xt(3,t-1)+ un(2)];

    end

    % ---------------------------------------------------------------- measurements
    %
    perc = .7; % percentage of Landmark measurement loss
    for t=1:N
        for landmark=1:size(L,2)
            if rand > perc
                % z = [distance angle]'
                z(:,t,landmark) = [ sqrt((L(1,landmark)-xt(1,t))^2 + (L(2,landmark)-xt(2,t))^2)+randn*m_err;...
                    atan2(L(2,landmark)-xt(2,t),L(1,landmark)-xt(1,t)) - xt(3,t)+randn*m_err];
```

```
            else
                z(:,t,landmark) = [0;0];
            end
        end
    end

else % logfile

 fid = fopen(logfilename,'r');
    for t=1:N
        tline = fgetl(fid);
        [type,success] = sscanf(tline, '%s', 1);
        if strcmp(type,'mark')
            fprintf(1,'*')
            continue
        end

        [xt(:,t),success] = sscanf(tline, 'obs: %*d %f %f %f', 3);
        xt(1,t)=xt(1,t)/100; % milimeters to decimeters
        xt(2,t)=xt(2,t)/100; % degrees to radians
        xt(3,t)=xt(3,t)*pi/180;
        if t > 1
            dx=xt(1,t)-xt(1,t-1);
            dy=xt(2,t)-xt(2,t-1);

            u(2,t) = xt(3,t)-xt(3,t-1); % diff_angle
            u(1,t) = sqrt (dx*dx+dy*dy); % speed
        end
        for landmark=1:6
            z(:,t,landmark) = [0;0];
        end

        [obs_landmarks, success,errmsg,nextindex] = sscanf(tline, 'obs: %*d %*f %*f %*f %d', 1);
        for observation=1:obs_landmarks
            tline=tline(1,nextindex:size(tline,2));
            [signature, success] = sscanf(tline, ' ( %d:%d', 2);
            for landmark = 1:6
                if signature(1) == LID(1,landmark) && signature(2) == LID(2,landmark)
                    [z(:,t,landmark),success,errmsg,nextindex] = sscanf(tline, ' ( %*d:%*d %f %f )', 2);
                    z(1,t,landmark) = z(1,t,landmark) /
```

```
                        100; % milimeters to decimeters
                    z(2,t,landmark) = z(2,t,landmark) *
                        pi / 180; % degrees to radians
                end
            end % for landmarks
        end % for observations

    end % for t=1:N
    fclose(fid)
end % if logfile


% ------------------------------------------------------------------------------ a prioris
%
x_ = xt(1:3,1); % a priori x = true robot position
P_ =  0*eye(3); % a priori P = very certain (no error)

% ------------------------------------------------------------------------------------------ EKF
 %
x = zeros( dim, N );
P = zeros( dim, dim, N );
I = eye(dim);
match = ones(1, N);

for t = 1:N
    % ------------------------------------------------------------------------ prediction
    %

    %get user input
    v = u(1,t);  % velocity
    da = u(2,t); % delta angle

    % Jacobian with respect to robot location
    G = [1          0 -v*sin(x_(3)+da);...
         0          1  v*cos(x_(3)+da);...
         0          0                1];

    % Jacobian with respect to control
    V = [cos(x_(3)+da) -v*sin(x_(3)+da);...
         sin(x_(3)+da)  v*cos(x_(3)+da);...
```

```
                         0                      1];

    % predicted robot position mean
    x_ = [x_(1) + v*cos(x_(3)+da);...
          x_(2) + v*sin(x_(3)+da);...
                         x_(3)+da];

    % predicted covariance
    P_ = G*P_*G' + V*M*V';


    %Custom, for plotting the positions, and uncertainty:
    %quiver(x_(1), x_(2), 0.3*cos(x_(3)), 0.3*sin(x_(3)));
    %hold on;
    %--------

    % ------------------------------------------------------------------ correction
    %
    for landmark = 1:size(z,3)
        if z(1,t,landmark) ~= 0   % if Landmark is measured

            % predicted measurement
            z_ = [sqrt((L(1,landmark)-x_(1))^2 + (L(2,landmark)-x_(2))^2);...
                 atan2(L(2,landmark)-x_(2),L(1,landmark)-x_(1)) - x_(3)];

            % Jacobian of H with respect to location
            H(:,:,landmark) = [ -(L(1,landmark)-x_(1))/(L(1,landmark)^2-2*L(1,landmark)*x_(1)+x_(1)^2+L(2,landmark)^2-2*L(2,landmark)*x_(2)+x_(2)^2)^(1/2), -(L(2,landmark)-x_(2))/(L(1,landmark)^2-2*L(1,landmark)*x_(1)+x_(1)^2+L(2,landmark)^2-2*L(2,landmark)*x_(2)+x_(2)^2)^(1/2),   0;
                 (L(2,landmark)-x_(2))/(L(1,landmark)^2-2*L(1,landmark)*x_(1)+x_(1)^2+L(2,landmark)^2-2*L(2,landmark)*x_(2)+x_(2)^2),          -(L(1,landmark)-x_(1))/(L(1,landmark)^2-2*L(1,landmark)*x_(1)+x_(1)^2+L(2,landmark)^2-2*L(2,landmark)*x_(2)+x_(2)^2),  -1];
```

```
            % predicted  measurement covariance
            S = H(:,:,landmark)*P_*H(:,:,landmark)' + Q;

            %Kalman gain
            K(:,:,landmark) = P_* H(:,:,landmark)' / S;

            %innovation
            nu = z(:,t,landmark) - z_;

            %validation gate
            ro = nu'/S*nu; % From Kristensen IROS'03, section III.A

            if ro < 2
                %updated mean and covariance
                foundx(:,landmark) = x_ + K(:,:,landmark)*nu;
                foundP_(:,:,landmark) = (I-K(:,:,landmark)*H(:,:,landmark))*P_;
            else
                %propagate known mean and covariance
                foundx(:,landmark) = x_;
                foundP_(:,:,landmark) = P_;
                z(:,t,landmark)=[0; 0];
            end

        else
            %propagate known mean and covariance
            foundx(:,landmark) = x_;
            foundP_(:,:,landmark) = P_;
        end
    end

    % determine mean
    x_ = mean(foundx,2);
    P_ = mean(foundP_,3);

    % create history
    x(:,t) = x_;
    P(:,:,t) = P_;

    %Custom, for plotting the positions, and uncertainty:
    quiver(x_(1), x_(2), 0.3*cos(x_(3)), 0.3*sin(x_(3)));
    hold on;
    %show the uncertainty for every 20th position
```

```
    %if mod(t, 20) == 0
        %use a covariance of 2D
        error_ellipse(diag([P_(1,1);P(3,3)]), x_)
        hold on;
    %end
    %--------
end
```

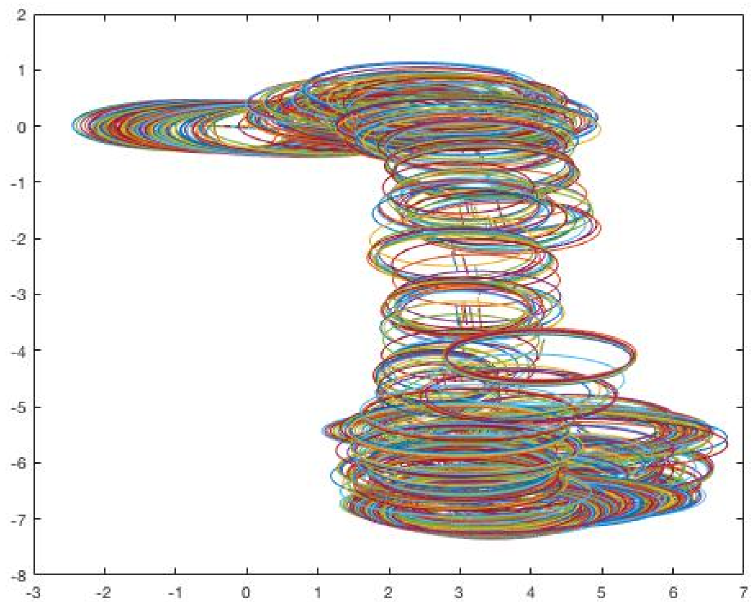


Figure 1: All locations with their uncertainty for the first mark.

To make the figures more readable, we plotted the uncertainty in the upcoming figures for every 20th observation (as can be seen in figure 2).

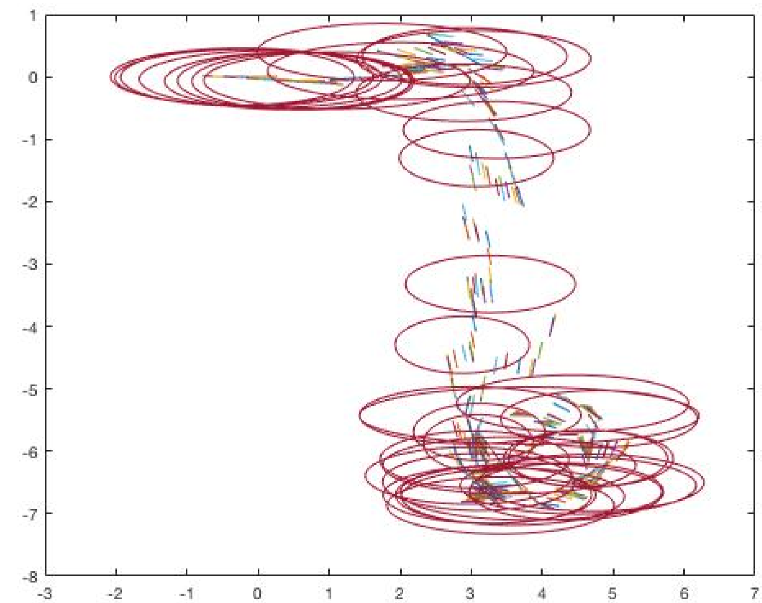
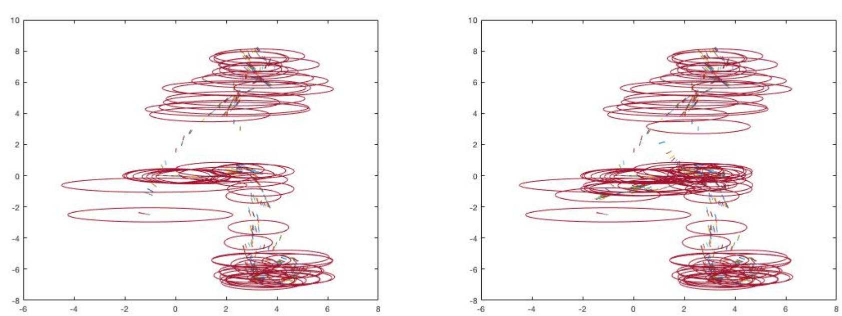


Figure 2: First mark. Error ellipse shown for location at every 20th time step.

The figures for the other marks are shown in figures 3a and 3b.



(a) Second mark.

(b) Third mark.

Figure 3: Error ellipse shown for every 20th location.

0.1 Observations

1. For this case, we observe how certain ellipses have smaller covariance when they are closer to a landmark. This indicates that there is a considerable

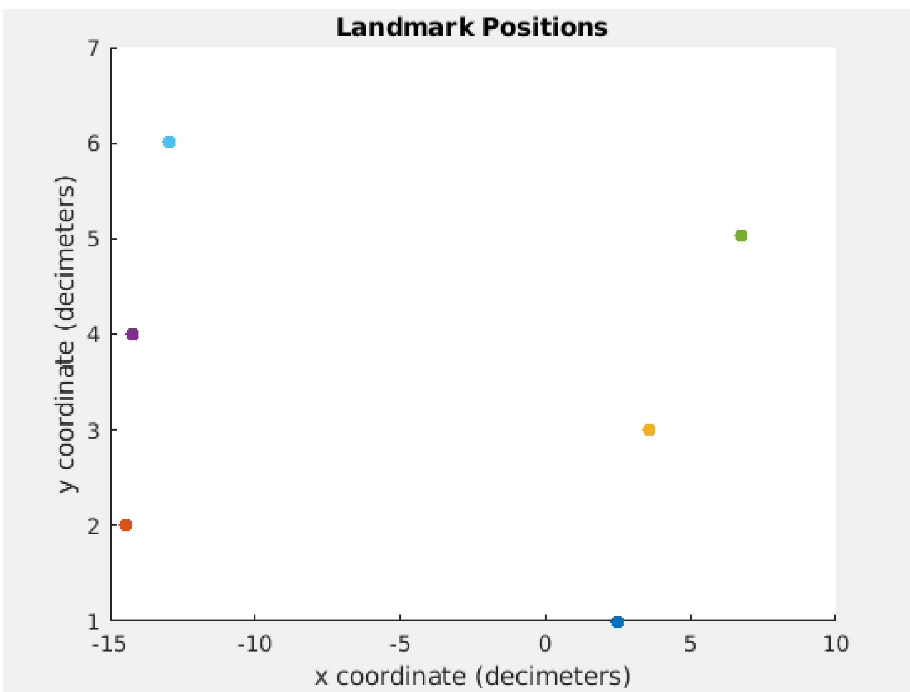
decrease in uncertainty about the pose as the robot gets increasingly certain about landmark positions.

2. It is also important to note that when executing angular displacements, the uncertainty in both x and y increases showing that angular movements cause greater uncertainty in both parameters as compared to linear motions wherein only one parameter drastically changes.

   This increase can be seen at the points where the robot is rapidly turning to make a loop for the first mark.

3. While 6 landmarks is not a very high overhead, a larger number of landmarks could involve more iterations over the central loop for each time interval and could be more computationally complex. Also worth noting is that the AIBO bot only observes 1 or 2 landmarks at a given time, however, the update process must run over all possible values of

Task 2 - SLAM



(a) Landmarks, in the left the path of the observations is visible.

Figure 4: Landmarks.

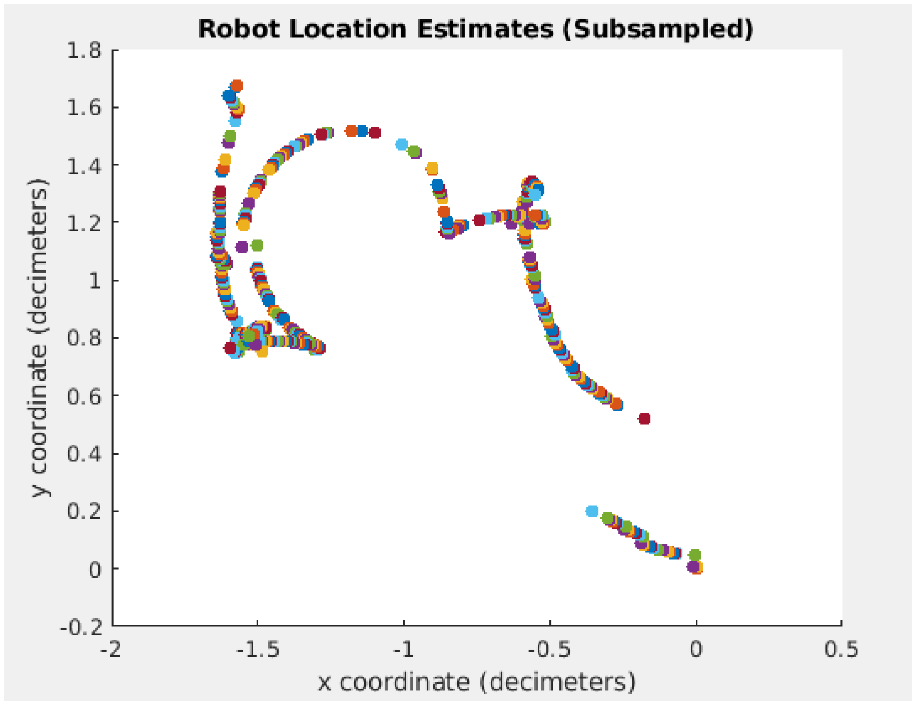
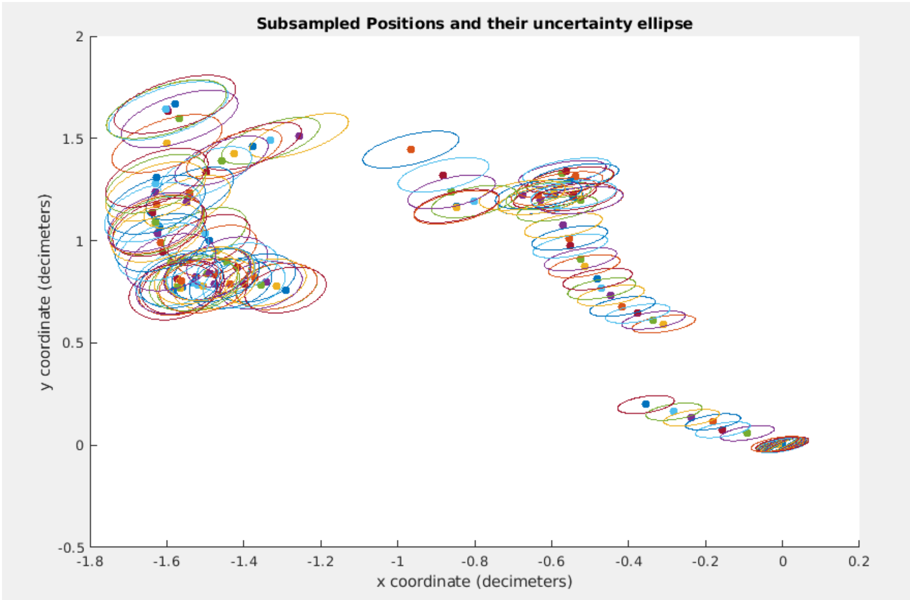


Figure 5: Pose Estimates for Robot (subsampled)



(a) Poses with Uncertainty Ellipses

Figure 6: Poses with Uncertainty Ellipses (subsampled for clarity)

1 Assumptions and Approximations

For the SLAM model, we make the following assumptions

1. For the control noise we assume a noise of 15% on the range measurements and 10°on the bearing measurement. Since the bearing measurement is an offset in degrees we express it as a percentage of the current bearing measurement in order to integrate it into the measurement noise Q_t. This is done by taking the percentage that 10°or $\frac{\pi}{9}$ radians is off the bearing measurement and then generating a new Q matrix at each iteration for every new measurement.

2. To keep the model complete and in accordance with the algorithm given in *Probabilistic Robotics*, we also include the signature variable. It is worth noting that the signature does not play any part in this example since observations are definitive as to which landmark they have been received from. Accordingly, the signature noise is zero. In order to retain the signature, we instead provide its value to be the same as the current landmark we are inspecting. Since signature recognition is assumed to be perfect, this is sufficient.

3. For certain poses, the covariance is positive singular and hence cannot be used as a valid covariance. This is resolved by adding slight noise to all parameters to resolve this.

4. For the control error of 0.1, we observed a strong deviation from the current position and extremely large variance scores. To correct this, we use the control noise as 10^{-3}

Listing 2: State Predictions using EKF-SLAM

```

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Extended Kalman Filter
%   by  J rgen  Sturm, Tijn Schmits, Arnoud Visser
%   April 2008 - October 2017
%
% Based on:
%
% Wolfram Burgard's
% http://ais.informatik.uni-freiburg.de/teaching/ss07/robotics/slides/
% --> 09.pdf
%
% Thrun's
% http://robots.stanford.edu/probabilistic-robotics/ppt/slam.ppt
%
% Dataset dlog.dat provided by Steffen Gutmann, 6.5.2004
% http://cres.usc.edu/radishrepository/view-one.php?name=comparison_of_self-localization_methods_continued
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ---------------------------------------------------------------------------- init
%
clear;

% N is number of observations in dlog.dat

logfilename = 'dlog_firstmark.dat'; N = 758;
%logfilename = 'dlog_secondmark.dat'; N = 1159;
%logfilename = 'dlog_thirdmark.dat'; N = 1434;

```

```
% expected user input noise
u_err = 0.0015;
M = u_err*eye(2);

% expected robot location noise
m_err = .01;
Q = m_err*eye(3);

% ---------------------------------------------------------------- data creation
%
% true robot position at t = 1
xt(:,1) = [0 0 0]'; dim = 3;  % x = [x y angle]'

% user input at t = 1
u(:,1) = [0 0]';              % u = [speed delta_angle]'

% Landmark locations
L2006 = [20  20 -20 -20;...
         20 -20  20 -20];

% You also need the following information about the landmark positions:
% cyan:magenta -1500 -1000 magenta:cyan -1500 1000 magenta:green 0 -1000 green:magenta 0 1000 yellow:magenta 1500 -1000 magenta:yellow 1500 1000
% 0 -> green 1 -> magenta 2 -> yellow 3 -> blue
L = [-15 -15 0 0 15 15;-10 10 -10 10 -10 10];
LID = [3 1 1 0 2 1;1 3 0 1 1 2];
U = M;          % user input noise (set to be equal to expected input noise)

angle = 0;

logfile = true;

if ~logfile

    for t=2:N

        % fabricate user input
        u(2,t) = randn;
        if abs(u(2,t)) > 0.4 %  P(steering) = 0.4
            u(2,t) = 0;
```

```
        end
        u(1,t) = .5*(1 - u(2,t)/0.4); % high delta_angle --> low speed

        % create noisy user input
        un = U*randn(2,1) +u(:,t);

        % calculate true robot position t+1
        xt(:,t) =  [xt(1,t-1)+ un(1)*cos(xt(3,t-1)) ; ...
            xt(2,t-1)+ un(1)*sin(xt(3,t-1)) ; ...
            xt(3,t-1)+ un(2)];

    end

    % ------------------------------------------------------------------ measurements
    %
    perc = .7; % percentage of Landmark measurement loss
    for t=1:N
        for landmark=1:size(L,2)
            if rand > perc
                % z = [distance angle]'
                z(:,t,landmark) = [ sqrt((L(1,landmark)-xt(1,t))^2 + (L(2,landmark)-xt(2,t))^2)+randn*m_err;...
                 atan2(L(2,landmark)-xt(2,t),L(1,landmark)-xt(1,t)) - xt(3,t)+randn*m_err];
            else
                z(:,t,landmark) = [0;0];
            end
        end
    end

else % logfile

 fid = fopen(logfilename,'r');
    for t=1:N
        tline = fgetl(fid);
        [type,success] = sscanf(tline, '%s', 1);
        if strcmp(type,'mark')
            fprintf(1,'*')
            continue
        end

```

```
        [xt(:,t),success] = sscanf(tline, 'obs: %*d %f %f %f', 3);
        xt(1,t)=xt(1,t)/100; % milimeters to decimeters
        xt(2,t)=xt(2,t)/100; % degrees to radians
        xt(3,t)=xt(3,t)*pi/180;
        if t > 1
            dx=xt(1,t)-xt(1,t-1);
            dy=xt(2,t)-xt(2,t-1);

            u(3,t) = atan2(dy,dx) - xt(3,t-1); % start_angle
            u(3,t) = mod(u(3,t) + pi, 2*pi) - pi;
            u(2,t) = xt(3,t)-xt(3,t-1); % diff_angle
            u(2,t) = mod(u(2,t) + pi, 2*pi) - pi; % FIX angle difference range
            u(1,t) = sqrt (dx*dx+dy*dy); % speed
        end
        for landmark=1:6
            z(:,t,landmark) = [0;0];
        end

        [obs_landmarks, success,errmsg,nextindex] = sscanf(tline, 'obs: %*d %*f %*f %*f %d', 1);
        for observation=1:obs_landmarks
            tline=tline(1,nextindex:size(tline,2));
            [signature, success] = sscanf(tline, ' ( %d:%d', 2);
            for landmark = 1:6
                if signature(1) == LID(1,landmark) && signature(2) == LID(2,landmark)
                    [z(:,t,landmark),success,errmsg,nextindex] = sscanf(tline, ' ( %*d:%*d %f %f )', 2);
                    z(1,t,landmark) = z(1,t,landmark) / 100; % milimeters to decimeters
                    z(2,t,landmark) = z(2,t,landmark) * pi / 180; % degrees to radians
                end
            end % for landmarks
        end % for observations

    end % for t=1:N
    fclose(fid)
end % if logfile
```

```
% ------------------------------------------------------------ a prioris
%
% x_ = xt(1:3,1); % a priori x = true robot position
% P_ =  0*eye(3); % a priori P = very certain (no error)
figure
numLandmarks = 6;
x_ = zeros(3*numLandmarks+3,1); % a priori x = true robot position
P_ = diag(horzcat([0,0,0],ones(1,(3*numLandmarks))*10000000)); % a priori P = very certain (no error)


% -------------------------------------------------------------------- EKF
 %
% x = zeros( dim, N );
% P = zeros( dim, dim, N );
% I = eye(dim);
% match = ones(1, N);

combined_dim = 3*numLandmarks + 3;
x = zeros( combined_dim, N );
P = zeros( combined_dim, combined_dim, N );
I = eye(combined_dim);
match = ones(1, N);


for t = 1:N
    % ------------------------------------------------------------ prediction
    %

    %get user input
    v = u(1,t);  % velocity
    da = u(2,t); % delta angle
    sa = u(3,t);    % start angle

    % Create F matrix

    F = horzcat(eye(3), zeros(3, 3*numLandmarks));

    % Jacobian with respect to robot location
```

```
    G = I + F'*[0   0  -(v)*cos(x_(3))+(v)*cos(x_(3)+(sa));...
                 0   0  -(v)*sin(x_(3))+(v)*sin(x_(3)+(sa));...
                 0   0     0]*F;

    % Jacobian with respect to control
    V = [cos(x_(3)+sa) -v*sin(x_(3)+sa);...
         sin(x_(3)+sa) -v*cos(x_(3)+sa);...
                   0                  1];

    x_ = x_ + F'*[-(v)*sin(x_(3))+(v)*sin(x_(3)+(sa));...
                  +(v)*cos(x_(3))-(v)*cos(x_(3)+(sa));...
                  (da)];

     R = V*M*V';

    % predicted covariance
    P_ = G*P_*G' + F'*R*F;

    % ------------------------------------------------------------------ correction
    %
%     for landmark = 1:size(z,3)
%         if z(1,t,landmark) ~= 0   % if Landmark is measured


      %x_landmarks = zeros(6,3);
      seenLandMarks = [];

      for landmark = 1:size(z,3)
        if z(1,t,landmark) ~= 0   % if Landmark is measured

            if ~ismember(landmark,seenLandMarks)
                x_(3*landmark) = x_(1) + z(1,t,landmark)*cos(z(2,t,landmark)+x_(3));
                x_(3*landmark+1) = x_(2) + z(1,t,landmark)*sin(z(2,t,landmark)+x_(3));
                x_(3*landmark+2) = landmark;
                seenLandMarks = [seenLandMarks,landmark];
            end


            % Generate Q Matrix
```

```

            bearing = z(2,t,landmark);
            bearing_error_percentage = abs((pi/9)/(bearing));
            Q = [0.15,0,0;0,bearing_error_percentage,0;0,0,0.000001];


            delta_x = x_(3*landmark) - x_(1);
            delta_y = x_(3*landmark+1) - x_(2);

            delta = [delta_x;delta_y];

            q = delta'*delta;
            % predicted measurement
            z_cap = [sqrt(q);...
                atan2(delta_y,delta_x)-x_(3);...
                x_(3*landmark+2)];

            F_xj = [[eye(3);zeros(3,3)],zeros(6,3*landmark-3),[zeros(3,3);eye(3)],zeros(6,(3*numLandmarks-3*landmark))];

            % Jacobian of H with respect to location
            H(:,:,landmark) = (1/q) * [-sqrt(q)*delta_x, -sqrt(q)*delta_y, 0, +sqrt(q)*delta_x, sqrt(q)*delta_y, 0;...
                            delta_y, -delta_x, -q, -delta_y, +delta_x, 0;...
                            0, 0, 0, 0, 0, q] * F_xj;

            % predicted  measurement covariance
            S = H(:,:,landmark)*P_*H(:,:,landmark)' + Q;

            %Kalman gain
            K(:,:,landmark) = P_* H(:,:,landmark)' / S;

            %innovation
            nu = [z(:,t,landmark);landmark] - z_cap;

            %validation gate
            ro = nu'/S*nu; % From Kristensen IROS'03, section III.A

            if ro < 2
                %updated mean and covariance
```

```
                foundx(:,landmark) = x_ + K(:,:,landmark)*nu;
                foundP_(:,:,landmark) = (I-K(:,:,landmark)*H(:,:,landmark))*P_;
            else
                %propagate known mean and covariance
                foundx(:,landmark) = x_;
                foundP_(:,:,landmark) = P_;
                z(:,t,landmark)=[0; 0];
            end

        else
            %propagate known mean and covariance
            foundx(:,landmark) = x_;
            foundP_(:,:,landmark) = P_;
        end
    end

    % determine mean
    x_ = mean(foundx,2);
    P_ = mean(foundP_,3);

    % create history
    x(:,t) = x_;
    P(:,:,t) = P_;
end
```

2 Observations

- The initial pose estimates are only affected by the noise in motion and measurement. It is also worth noting that given the estimate for a landmark, only the values for that landmark are readjusted because of the structure of the conditioning matrix $F_{j,x}$
- The signature is not required for this case. Signatures refer to the characteristics that can help you identify which landmark is currently in sight which like any measurement can be inherently noisy. However, for this case, sensor measurements are definite indicating no ambiguity in determining the signature. Hence the signature need not be updated.
- While *Probabilistic Robotics* suggests using ∞ as the variance for the landmarks positions given the fact that we have no prior knowledge of them (one of the fundamental use-cases for SLAM), we chose to set them to infinity however it leads to singular errors when the simulation tries to invert this matrix.

To remedy this, we use large numbers like 10^{10} instead of infinity. This, however, has a detrimental effect of introducing noise in the landmarks as can be seen in our case where landmarks strongly deviate from their actual position. This is because the large numbers do not effectively block out the effects from a multiplication with G matrix as effectively as a value of ∞ would do.

- For some of the cases, the especially where there is a strong bearing change, the error ellipse looks increasingly like a circle indicating greater uncertainty in the x and y position. For cases where there is only a linear movement indicated by a considerable range change but smaller bearing change, the position is almost certain in one parameter but varies in the other. This can be seen by the drastic difference between the major and minor axis of those error ellipses.

- In cases with large movements or noisy measurements, there is a visible new second line of pose hypotheses visible. This indicates that under non-gaussian noise or sparse measurements the posterior has a greater amount of uncertainty. In this case, the Taylor expansion based linearization can fail.