

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

```
1
2 %
3 %
4 %   Extended Kalman Filter
5 %   by J rgen Sturm, Tijn Schmits, Arnoud Visser
6 %   April 2008
7 %
8 % Based on:
9 %
10 % Wolfram Burgard 's
11 % http://ais.informatik.uni-freiburg.de/teaching/ss07/
12 %   robotics/slides/
13 %   —> 09.pdf
14 %
15 % Thrun 's
16 % http://robots.stanford.edu/probabilistic-robotics/ppt/
17 %   slam.ppt
18 %
19 % Dataset dlog.dat provided by Steffen Gutmann, 6.5.2004
20 % http://cres.usc.edu/radishrepository/view-one.php?name=
21 %   comparison_of_self-localization_methods_continued
22 %
23 %
```

¹<https://nl.mathworks.com/matlabcentral/fileexchange/4705-error-ellipse>

```

21 |
22 | %
    | _____
    |
    |     init
23 | %
24 | clear;
25 |
26 | % N is number of observations in dlog.dat
27 |
28 | logfilename = 'dlog_firstmark.dat'; N = 758;
29 | %logfilename = 'dlog_secondmark.dat'; N = 1159;
30 | %logfilename = 'dlog_thirdmark.dat'; N = 1434;
31 |
32 |
33 | % expected user input noise
34 | u_err = .15;
35 | M = u_err*eye(2);
36 |
37 | % expected robot location noise
38 | m_err = .1;
39 | Q = m_err*eye(2);
40 |
41 | %Custom, for plotting the position
42 | figure
43 | % _____
44 |
45 | %
    | _____
    |
    |     data creation
46 | %
47 | % true robot position at t = 1
48 | xt(:,1) = [0 0 0]'; dim = 3; % x = [x y angle]'
49 |
50 | % user input at t = 1
51 | u(:,1) = [0 0]'; % u = [speed delta_angle]'
52 |
53 | % Landmark locations
54 | L2006 = [20 20 -20 -20;...
55 |          20 -20 20 -20];
56 |
57 | % You also need the following information about the
    | landmark positions:
58 | % cyan:magenta -1500 -1000 magenta:cyan -1500 1000
    | magenta:green 0 -1000 green:magenta 0 1000 yellow:
    | magenta 1500 -1000 magenta:yellow 1500 1000
59 | % 0 -> green 1 -> magenta 2 -> yellow 3 -> blue

```

```

60 L = [-15 -15 0 0 15 15;-10 10 -10 10 -10 10];
61 LID = [3 1 1 0 2 1;1 3 0 1 1 2];
62 U = M; % user input noise (set to be equal to
        expected input noise)
63
64 angle = 0;
65
66 logfile = true;
67
68 if ~logfile
69     for t=2:N
70
71         % fabricate user input
72         u(2,t) = randn;
73         if abs(u(2,t)) > 0.4 % P(steering) = 0.4
74             u(2,t) = 0;
75         end
76         u(1,t) = .5*(1 - u(2,t)/0.4); % high delta_angle
77             --> low speed
78
79         % create noisy user input
80         un = U*randn(2,1) +u(:,t);
81
82         % calculate true robot position t+1
83         xt(:,t) = [xt(1,t-1)+ un(1)*cos(xt(3,t-1)) ; ...
84                 xt(2,t-1)+ un(1)*sin(xt(3,t-1)) ; ...
85                 xt(3,t-1)+ un(2)];
86
87     end
88
89     %
90
91     %
92     % measurements
93     perc = .7; % percentage of Landmark measurement loss
94     for t=1:N
95         for landmark=1:size(L,2)
96             if rand > perc
97                 % z = [distance angle]'
98                 z(:,t,landmark) = [ sqrt((L(1,landmark)-
99                                     xt(1,t))^2 + (L(2,landmark)-xt(2,t))
100                                 ^2)+randn*m_err;...
101                 atan2(L(2,landmark)-xt(2,t),L(1,
102                     landmark)-xt(1,t)) - xt(3,t)+randn
103                     *m_err];

```

```

98         else
99             z(:,t,landmark) = [0;0];
100         end
101     end
102 end
103
104 else % logfile
105
106     fid = fopen(logfilename, 'r');
107     for t=1:N
108         tline = fgetl(fid);
109         [type,success] = sscanf(tline, '%s', 1);
110         if strcmp(type, 'mark')
111             fprintf(1, '*')
112             continue
113         end
114
115         [xt(:,t),success] = sscanf(tline, 'obs: %*d %f %f
116             %f', 3);
117         xt(1,t)=xt(1,t)/100; % millimeters to decimeters
118         xt(2,t)=xt(2,t)/100; % degrees to radians
119         xt(3,t)=xt(3,t)*pi/180;
120         if t > 1
121             dx=xt(1,t)-xt(1,t-1);
122             dy=xt(2,t)-xt(2,t-1);
123
124             u(2,t) = xt(3,t)-xt(3,t-1); % diff_angle
125             u(1,t) = sqrt(dx*dx+dy*dy); % speed
126         end
127         for landmark=1:6
128             z(:,t,landmark) = [0;0];
129         end
130
131         [obs_landmarks, success,errmsg,nextindex] =
132             sscanf(tline, 'obs: %*d %*f %*f %*f %*d', 1);
133         for observation=1:obs_landmarks
134             tline=tline(1,nextindex:size(tline,2));
135             [signature, success] = sscanf(tline, '( %d:%d
136                 %d', 2);
137             for landmark = 1:6
138                 if signature(1) == LID(1,landmark) &&
139                     signature(2) == LID(2,landmark)
140                     [z(:,t,landmark),success,errmsg,
141                         nextindex] = sscanf(tline, '( %*d
142                             %*d %f %f )', 2);
143                     z(1,t,landmark) = z(1,t,landmark) /

```

```

138         100; % millimeters to decimeters
           z(2,t,landmark) = z(2,t,landmark) *
139             pi / 180; % degrees to radians
           end
140     end % for landmarks
141 end % for observations
142
143 end % for t=1:N
144 fclose(fid)
145 end % if logfile
146
147
148 %


---


149 % a priori
150 %
151 x_ = xt(1:3,1); % a priori x = true robot position
152 P_ = 0*eye(3); % a priori P = very certain (no error)
153 %


---


154 % EKF
155 x = zeros( dim, N );
156 P = zeros( dim, dim, N );
157 I = eye(dim);
158 match = ones(1, N);
159
160 for t = 1:N
161     %


---


162     % prediction
163
164     %get user input
165     v = u(1,t); % velocity
166     da = u(2,t); % delta angle
167
168     % Jacobian with respect to robot location
169     G = [1          0 -v*sin(x_(3)+da);...
170          0          1  v*cos(x_(3)+da);...
171          0          0          1];
172
173     % Jacobian with respect to control
174     V = [cos(x_(3)+da) -v*sin(x_(3)+da);...
175          sin(x_(3)+da)  v*cos(x_(3)+da);...

```

```

176         0           1];
177
178 % predicted robot position mean
179 x_ = [x_(1) + v*cos(x_(3)+da);...
180       x_(2) + v*sin(x_(3)+da);...
181       x_(3)+da];
182
183 % predicted covariance
184 P_ = G*P_*G' + V*M*V';
185
186
187 %Custom, for plotting the positions, and uncertainty:
188 %quiver(x_(1), x_(2), 0.3*cos(x_(3)), 0.3*sin(x_(3)))
189 ;
190 %hold on;
191 %-----
192 %
193
194 % correction
195 for landmark = 1:size(z,3)
196     if z(1,t,landmark) ~= 0 % if Landmark is
197         measured
198         % predicted measurement
199         z_ = [sqrt((L(1,landmark)-x_(1))^2 + (L(2,
200             landmark)-x_(2))^2);...
201             atan2(L(2,landmark)-x_(2),L(1,landmark)-
202                 x_(1)) - x_(3)];
203
204         % Jacobian of H with respect to location
205         H(:, :, landmark) = [ -(L(1,landmark)-x_(1))/(L
206             (1,landmark)^2-2*L(1,landmark)*x_(1)+x_(1)
207             ^2+L(2,landmark)^2-2*L(2,landmark)*x_(2)+
208             x_(2)^2)^(1/2), -(L(2,landmark)-x_(2))/(L
209             (1,landmark)^2-2*L(1,landmark)*x_(1)+x_(1)
210             ^2+L(2,landmark)^2-2*L(2,landmark)*x_(2)+
211             x_(2)^2)^(1/2), 0;
212             (L(2,landmark)-x_(2))/(L(1,landmark)^2-2*
213             L(1,landmark)*x_(1)+x_(1)^2+L(2,
214             landmark)^2-2*L(2,landmark)*x_(2)+x_(
215             2)^2), -(L(1,landmark)-x_(1))/(
216             L(1,landmark)^2-2*L(1,landmark)*x_(1)+
217             x_(1)^2+L(2,landmark)^2-2*L(2,landmark
218             )*x_(2)+x_(2)^2), -1];

```

```

204
205 % predicted measurement covariance
206 S = H(:, :, landmark)*P_*H(:, :, landmark)' + Q;
207
208 %Kalman gain
209 K(:, :, landmark) = P_* H(:, :, landmark)' / S;
210
211 %innovation
212 nu = z(:, t, landmark) - z_-;
213
214 %validation gate
215 ro = nu'/S*nu; % From Kristensen IROS'03,
                section III.A
216
217 if ro < 2
218     %updated mean and covariance
219     foundx(:, landmark) = x_- + K(:, :, landmark)
                *nu;
220     foundP_(:, :, landmark) = (I-K(:, :, landmark)
                )*H(:, :, landmark))*P_-;
221 else
222     %propagate known mean and covariance
223     foundx(:, landmark) = x_-;
224     foundP_(:, :, landmark) = P_-;
225     z(:, t, landmark)=[0; 0];
226 end
227
228 else
229     %propagate known mean and covariance
230     foundx(:, landmark) = x_-;
231     foundP_(:, :, landmark) = P_-;
232 end
233 end
234
235 % determine mean
236 x_- = mean(foundx, 2);
237 P_- = mean(foundP_, 3);
238
239 % create history
240 x(:, t) = x_-;
241 P(:, :, t) = P_-;
242
243 %Custom, for plotting the positions, and uncertainty:
244 quiver(x_-(1), x_-(2), 0.3*cos(x_-(3)), 0.3*sin(x_-(3)));
245 hold on;
246 %show the uncertainty for every 20th position

```

```

247     %if mod(t, 20) == 0
248         %use a covariance of 2D
249         error_ellipse(diag([P_(1,1);P(3,3)]), x-)
250         hold on;
251     %end
252     %—————
253 end

```

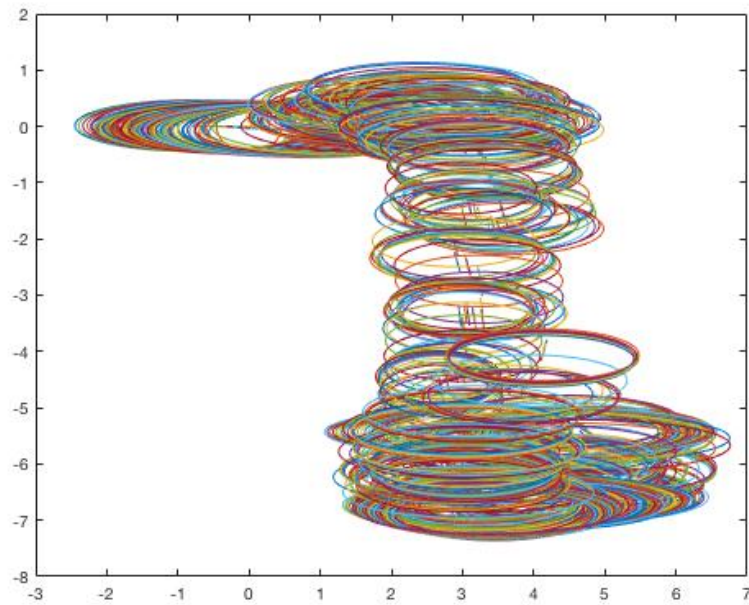


Figure 1: All locations with their uncertainty for the first mark.

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).

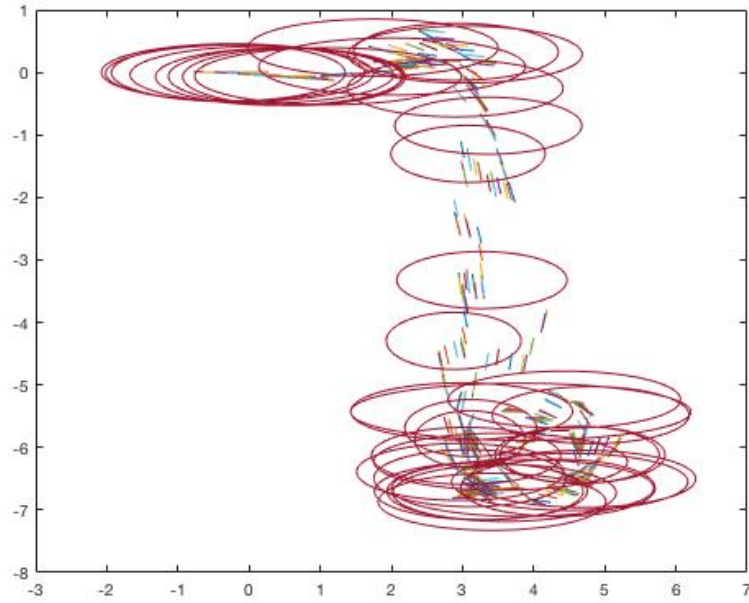
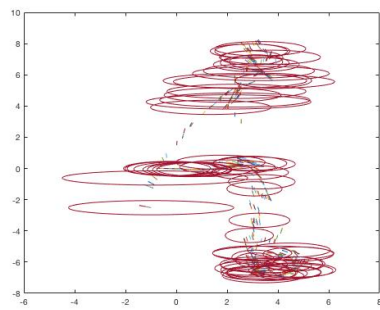
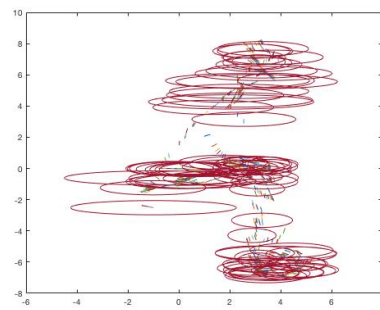


Figure 2: First mark. Error ellipse shown for location at every 20th time step.

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

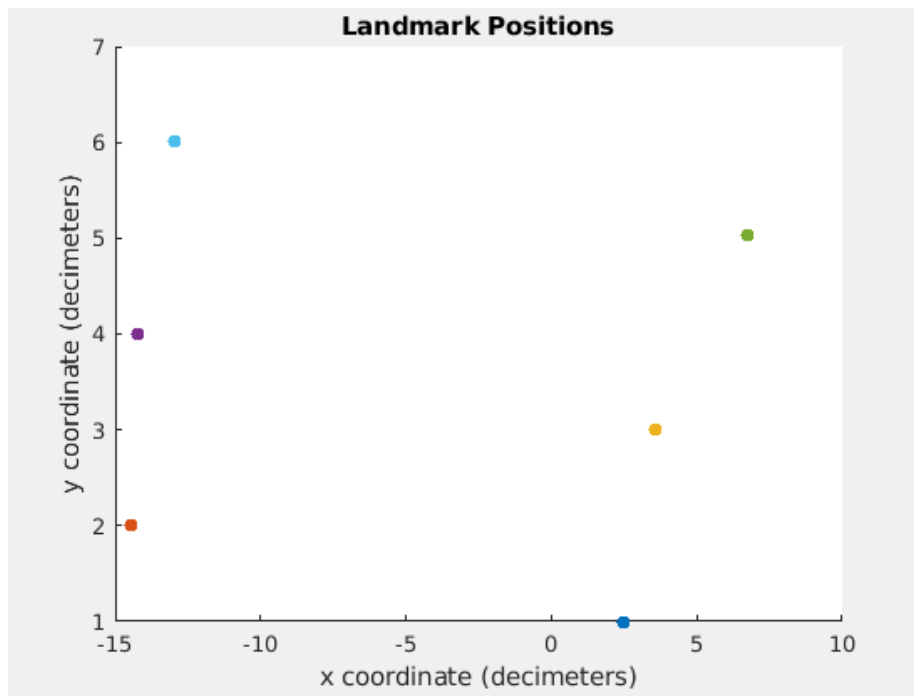
decrease in uncertainty about the pose as the robot gets increasingly certain about landmark positions.

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.

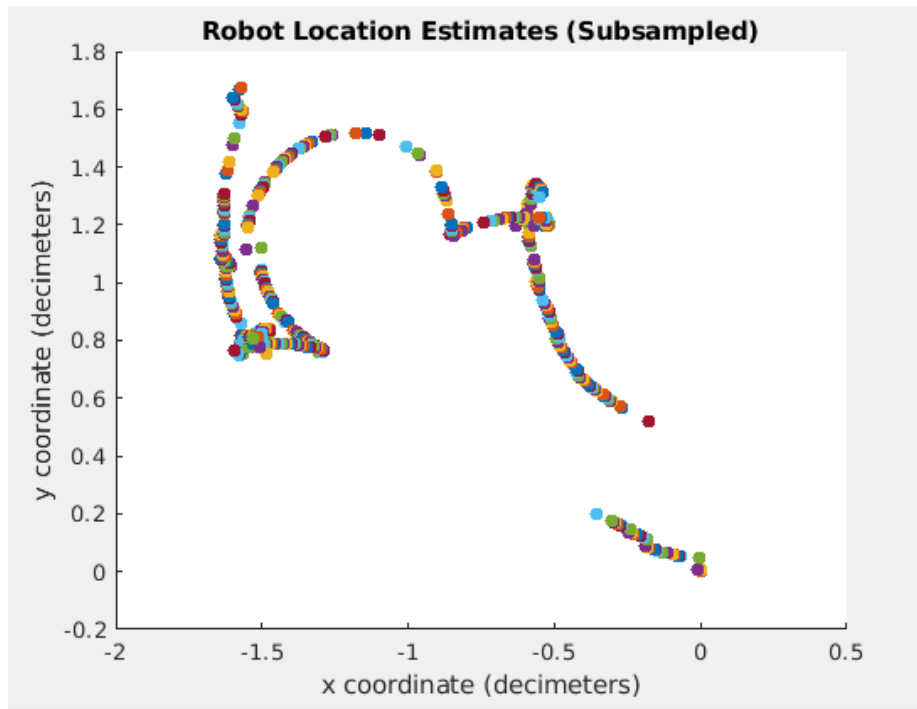
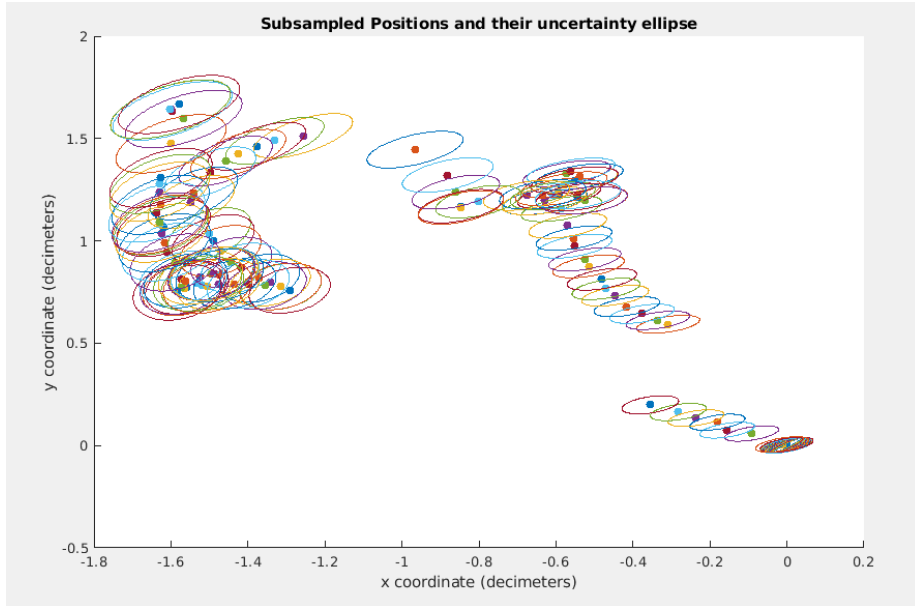


Figure 5: Pose Estimates for Robot (subsampled)



(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

```
1
2 %
3 %
4 %   Extended Kalman Filter
5 %   by J rgen Sturm, Tijn Schmits, Arnoud Visser
6 %   April 2008 – October 2017
7 %
8 % Based on:
9 %
10 % Wolfram Burgard 's
11 % http://ais.informatik.uni-freiburg.de/teaching/ss07/
12 %   robotics/slides/
13 % --> 09.pdf
14 % Thrun 's
15 % http://robots.stanford.edu/probabilistic-robotics/ppt/
16 %   slam.ppt
17 % Dataset dlog.dat provided by Steffen Gutmann, 6.5.2004
18 % http://cres.usc.edu/radishrepository/view-one.php?name=
19 %   comparison_of_self-localization_methods_continued
20 %
21 %
22 %
23 %   _____
24 %   init
25 %
26 % clear;
27 %
28 % N is number of observations in dlog.dat
29 % logfilename = 'dlog_firstmark.dat'; N = 758;
30 % logfilename = 'dlog_secondmark.dat'; N = 1159;
31 % logfilename = 'dlog_thirdmark.dat'; N = 1434;
```

```

32 |
33 | % expected user input noise
34 | u_err = 0.0015;
35 | M = u_err*eye(2);
36 |
37 | % expected robot location noise
38 | m_err = .01;
39 | Q = m_err*eye(3);
40 |
41 | %
42 |     data creation
43 | %
44 | % true robot position at t = 1
45 | xt(:,1) = [0 0 0]'; dim = 3; % x = [x y angle]'
46 | % user input at t = 1
47 | u(:,1) = [0 0]'; % u = [speed delta_angle]'
48 |
49 | % Landmark locations
50 | L2006 = [20 20 -20 -20;...
51 |         20 -20 20 -20];
52 |
53 | % You also need the following information about the
54 | % landmark positions:
55 | % cyan:magenta -1500 -1000 magenta:cyan -1500 1000
56 | % magenta:green 0 -1000 green:magenta 0 1000 yellow:
57 | % magenta 1500 -1000 magenta:yellow 1500 1000
58 | % 0 -> green 1 -> magenta 2 -> yellow 3 -> blue
59 | L = [-15 -15 0 0 15 15;-10 10 -10 10 -10 10];
60 | LID = [3 1 1 0 2 1;1 3 0 1 1 2];
61 | U = M; % user input noise (set to be equal to
62 | % expected input noise)
63 |
64 | angle = 0;
65 | logfile = true;
66 |
67 | if ~logfile
68 |     for t=2:N
69 |         % fabricate user input
70 |         u(2,t) = randn;
71 |         if abs(u(2,t)) > 0.4 % P(steering) = 0.4
72 |             u(2,t) = 0;

```

```

72     end
73     u(1,t) = .5*(1 - u(2,t)/0.4); % high delta_angle
74         → low speed
75
76     % create noisy user input
77     un = U*randn(2,1) +u(:,t);
78
79     % calculate true robot position t+1
80     xt(:,t) = [xt(1,t-1)+ un(1)*cos(xt(3,t-1)) ; ...
81               xt(2,t-1)+ un(1)*sin(xt(3,t-1)) ; ...
82               xt(3,t-1)+ un(2) ];
83
84     end
85
86     %
87     %
88     %
89     %
90     %
91     %
92     %
93     %
94     %
95     %
96     %
97     %
98     %
99     %
100    else % logfile
101
102    fid = fopen(logfilename, 'r');
103    for t=1:N
104        tline = fgetl(fid);
105        [type, success] = sscanf(tline, '%s', 1);
106        if strcmp(type, 'mark')
107            fprintf(1, '*')
108            continue
109        end
110

```

```

111     [xt(:,t),success] = sscanf(tline, 'obs: %*d %f %f
112         %f', 3);
113     xt(1,t)=xt(1,t)/100; % millimeters to decimeters
114     xt(2,t)=xt(2,t)/100; % degrees to radians
115     xt(3,t)=xt(3,t)*pi/180;
116     if t > 1
117         dx=xt(1,t)-xt(1,t-1);
118         dy=xt(2,t)-xt(2,t-1);
119         u(3,t) = atan2(dy,dx) - xt(3,t-1); %
120             start_angle
121         u(3,t) = mod(u(3,t) + pi, 2*pi) - pi;
122         u(2,t) = xt(3,t)-xt(3,t-1); % diff_angle
123         u(2,t) = mod(u(2,t) + pi, 2*pi) - pi; % FIX
124         angle difference range
125         u(1,t) = sqrt(dx*dx+dy*dy); % speed
126     end
127     for landmark=1:6
128         z(:,t,landmark) = [0;0];
129     end
130     [obs_landmarks, success,errmsg,nextindex] =
131         sscanf(tline, 'obs: %*d %*f %*f %*f %d', 1);
132     for observation=1:obs_landmarks
133         tline=tline(1,nextindex:size(tline,2));
134         [signature, success] = sscanf(tline, ' ( %d:%
135             d', 2);
136         for landmark = 1:6
137             if signature(1) == LID(1,landmark) &&
138                 signature(2) == LID(2,landmark)
139                 [z(:,t,landmark),success,errmsg,
140                     nextindex] = sscanf(tline, ' ( %*d
141                         %*d %f %f )', 2);
142                 z(1,t,landmark) = z(1,t,landmark) /
143                     100; % millimeters to decimeters
144                 z(2,t,landmark) = z(2,t,landmark) *
145                     pi / 180; % degrees to radians
146             end
147         end % for landmarks
148     end % for observations
149 end % if logfile
150
151     end % for t=1:N
152     fclose(fid)
153 end % if logfile
154
155

```



```

147 %
    a priori
148 %
149 % x_ = xt(1:3,1); % a priori x = true robot position
150 % P_ = 0*eye(3); % a priori P = very certain (no error)
151 figure
152 numLandmarks = 6;
153 x_ = zeros(3*numLandmarks+3,1); % a priori x = true robot
    position
154 P_ = diag(horzcat([0,0,0],ones(1,(3*numLandmarks))
    *10000000)); % a priori P = very certain (no error)
155
156
157 %
    EKF
158 %
159 % x = zeros( dim, N );
160 % P = zeros( dim, dim, N );
161 % I = eye(dim);
162 % match = ones(1, N);
163
164 combined_dim = 3*numLandmarks + 3;
165 x = zeros( combined_dim, N );
166 P = zeros( combined_dim, combined_dim, N );
167 I = eye(combined_dim);
168 match = ones(1, N);
169
170
171 for t = 1:N
172 %
    prediction
173 %
174
175 %get user input
176 v = u(1,t); % velocity
177 da = u(2,t); % delta angle
178 sa = u(3,t); % start angle
179
180 % Create F matrix
181
182 F = horzcat(eye(3), zeros(3, 3*numLandmarks));
183
184 % Jacobian with respect to robot location

```

```

185 G = I + F'*[0 0 -(v)*cos(x_(3))+(v)*cos(x_(3)+(sa));
      ...
186           0 0 -(v)*sin(x_(3))+(v)*sin(x_(3)+(sa));...
187           0 0 0]*F;
188
189 % Jacobian with respect to control
190 V = [cos(x_(3)+sa) -v*sin(x_(3)+sa);...
191       sin(x_(3)+sa) -v*cos(x_(3)+sa);...
192              0 1];
193
194 x_ = x_ + F'*[-(v)*sin(x_(3))+(v)*sin(x_(3)+(sa));...
195              +(v)*cos(x_(3))-(v)*cos(x_(3)+(sa));...
196              (da)];
197
198 R = V*M*V';
199
200 % predicted covariance
201 P_ = G*P_*G' + F'*R*F;
202
203 %

```

```

204 % correction
205 % for landmark = 1:size(z,3)
206 % if z(1,t,landmark) ~= 0 % if Landmark is
measured
207
208
209 %x_landmarks = zeros(6,3);
210 seenLandMarks = [];
211
212 for landmark = 1:size(z,3)
213 if z(1,t,landmark) ~= 0 % if Landmark is
measured
214
215 if ~ismember(landmark,seenLandMarks)
216 x_(3*landmark) = x_(1) + z(1,t,landmark)*
cos(z(2,t,landmark)+x_(3));
217 x_(3*landmark+1) = x_(2) + z(1,t,landmark
)*sin(z(2,t,landmark)+x_(3));
218 x_(3*landmark+2) = landmark;
219 seenLandMarks = [seenLandMarks,landmark];
220 end
221
222
223 % Generate Q Matrix

```

```

224
225 bearing = z(2,t,landmark);
226 bearing_error_percentage = abs((pi/9)/(
    bearing));
227 Q = [0.15,0,0;0,bearing_error_percentage
    ,0;0,0,0.000001];
228
229
230 delta_x = x_(3*landmark) - x_(1);
231 delta_y = x_(3*landmark+1) - x_(2);
232
233 delta = [delta_x;delta_y];
234
235 q = delta'*delta;
236 % predicted measurement
237 z_cap = [sqrt(q);...
238     atan2(delta_y,delta_x)-x_(3);...
239     x_(3*landmark+2)];
240
241 F_xj = [[eye(3);zeros(3,3)],zeros(6,3*
    landmark-3),[zeros(3,3);eye(3)],zeros
    (6,(3*numLandmarks-3*landmark))];
242
243 % Jacobian of H with respect to location
244 H(:, :, landmark) = (1/q) * [-sqrt(q)*delta_x ,
    -sqrt(q)*delta_y , 0 , +sqrt(q)*delta_x ,
    sqrt(q)*delta_y , 0;...
245     delta_y , -delta_x , -q , -
    delta_y , +delta_x , 0;...
246     0 , 0 , 0 , 0 , 0 , q] * F_xj;
247
248 % predicted measurement covariance
249 S = H(:, :, landmark)*P_*H(:, :, landmark)' + Q;
250
251 %Kalman gain
252 K(:, :, landmark) = P_* H(:, :, landmark)' / S;
253
254 %innovation
255 nu = [z(:, t, landmark);landmark] - z_cap;
256
257 %validation gate
258 ro = nu'/S*nu; % From Kristensen IROS'03,
    section III.A
259
260 if ro < 2
261     %updated mean and covariance

```

```

262         foundx(:, landmark) = x_ + K(:, :, landmark)
           *nu;
263         foundP_(:, :, landmark) = (I-K(:, :, landmark)
           )*H(:, :, landmark))*P_;
264     else
265         %propagate known mean and covariance
266         foundx(:, landmark) = x_;
267         foundP_(:, :, landmark) = P_;
268         z(:, t, landmark)=[0; 0];
269     end
270
271     else
272         %propagate known mean and covariance
273         foundx(:, landmark) = x_;
274         foundP_(:, :, landmark) = P_;
275     end
276 end
277
278 % determine mean
279 x_ = mean(foundx, 2);
280 P_ = mean(foundP_, 3);
281
282 % create history
283 x(:, t) = x_;
284 P(:, :, t) = P_;
285 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.