```
%EKF SLAM 2D
% Loading landmarks
load('W.mat')
%W = W;


% System parameters
% R -> robot pose (x, y, theta)
% u -> control (delta_t, delta_theta)
% y -> map (W ¡ú landmarks positions from enclosed file)

R = [0;-2.5;0];
u = [0.1;0.05];
y_olds = zeros(3,size(W,2)); % will be explained later
y_news = zeros(3,size(W,2));

% noise assumption
% Model evolution noise
q = [0.005;0.003];
Q = diag(q.^2);
% measurment noise
m = [.25; 1*pi/180];
M = diag(m.^2);

% sensor radius
sensor_r = 5.2;  % determins how fast the robot will find all
 landmarks
Id = zeros(1,size(W,2));

%  EKF-state and covariance matrix
% state (robot pose and landmark positions)
x = zeros(numel(R)+numel(W), 1);
% covariance matrix
P = zeros(numel(x),numel(x));

% id_to_x_map
id_to_x_map = zeros(1,size(W,2));

% predefined states of the robot
r = [1 2 3];
x(r) = R;
P(r,r) = 0;


% Graphics Intialization
s = [4 5];
loop =100;
poses_ = zeros(3,loop);

%
poses = zeros(3,loop);
```

```
 %
mapFig = figure(1);
cla;
axis([-7 7 -7 7])
axis square
%axis equal
% Draw landmarks
WG = line('parent',gca,...
    'linestyle','none',...
    'marker','+',...
    'color','r',...
    'xdata',W(1,:),...
    'ydata',W(2,:));
% Draw initial robot position
RG = line('parent',gca,...
    'marker','>',...
    'color','b',...
    'xdata',R(1),...
    'ydata',R(2));
% Draw initial robot position estimation
rG = line('parent',gca,...
    'linestyle','none',...
    'marker','.',...
    'color','r',...
    'xdata',x(r(1)),...
    'ydata',x(r(2)));
% Initialize objects for future landmarks drawing
lG = line('parent',gca,...
    'linestyle','none',...
    'marker','.',...
    'color','b',...
    'xdata',[],...
    'ydata',[]);

% Estimated covariance of landmark points
eG1 = zeros(1,size(W,2));
for i = 1:numel(eG1)
    eG1(i) = line(...
        'parent', gca,...
        'color','k',...
        'xdata',[],...
        'ydata',[]);
end

% Estimated robot position
reG = line(...
    'parent', gca,...
    'color','r',...
    'xdata',[],...
    'ydata',[]);

% Sensor detection range
%(take the real position as the center of the circle)
sensor1 = line(...
```

```
    'parent', gca,...
    'color','m',...
    'xdata',[],...
    'ydata',[],...
    'LineStyle','--');
sensor2 = line(...
    'parent', gca,...
    'color','m',...
    'xdata',[],...
    'ydata',[],...
    'LineStyle','--');

 true_pose = line(...
    'parent', gca,...
    'color','r',...
    'xdata',[],...
    'ydata',[],...
    'LineWidth',0.8);
     %'LineStyle','--');

 estimate_pose = line(...
    'parent', gca,...
    'color','b',...
    'xdata',[],...
    'ydata',[],...
    'LineWidth',0.8);
    % 'LineStyle','--');

 % II.
 %
for t = 1:loop

    n = q.*randn(2,1);
    % TODO Simulate robot movement and landmarks observation
    R = move(R, u, n);

   % Information obtained by the sensor; i represents the unique ID
   %identification number of the landmark point; yi represents the
   %coordinate of the observed feature point in the current coordinate
 system,
   %if it is zero, it means that the landmark point of this kind is
 not observed.
     % There are two sources of observed landmark points:
     % 1: It has been observed. EKF only needs to correct the
     %current state quantity according to the forward observation
 equation project.
     % 2: It has not been observed before. At this time, it is
     %necessary to expand the state vector, and initialize the new
     %state by using the inverse observation equation backProject.

     % y_olds represents the collection of old landmark points that
 have been observed.
     % y_news represents a collection of newly discovered landmarks.
    i_olds=1;
```

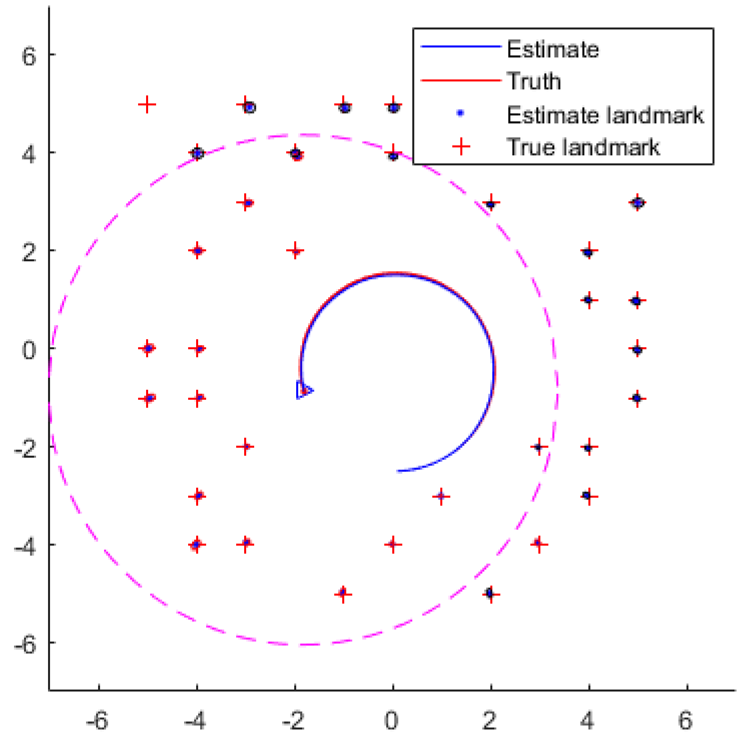
```
    i_news=1;

    for i = 1:size(W,2)
        v = m.*randn(2,1);
         yi= project(R, W(:,i)) + v;
        if yi(1) < sensor_r && Id(i) == 1
               y_olds(:,i_olds) = [yi(1);yi(2);i];
               i_olds = i_olds + 1;
        elseif  yi(1) < sensor_r &&  Id(i) == 0 %&& temp ==1
                y_news(:,i_news) = [yi(1);yi(2);i];
                i_news = i_news + 1;
                Id(i) = 1;
        end
    end

    for i = i_olds:size(W,2)
        y_olds(:,i) = [100;0;0];
    end
    for i = i_news:size(W,2)
        y_news(:,i) = [101;0;0];
    end


  % TODO Extended Kalman Filter
    % Prediction and correction with known landmarks
    % 2. EKF filter
     % prediction
     % x(r) is the one-step prediction position, R_r and R_n are
     %the Jacobian matrix of x(r) to R and n in the current state
    [x(r), R_r, R_n] = move(x(r), u, [0 0]);
    P_rr = P(r,r);
    P(r,:) = R_r*P(r,:);
    P(:,r) = P(r,:)';
    P(r,r) = R_r*P_rr*R_r' + R_n*Q*R_n';

% b. Correction
  % Processing method for multiple observations:
  %the observations are processed one by one,
  %and the status is updated according to the observation of a
 landmark
    end_old = find(y_olds(1,:)==100,1);
    if isempty(end_old)
        end_old=size(y_olds,2)+1;
    end

    for j = 1:(end_old-1)
        % expectation
        if isempty(j)
            break
        end
        id = find(id_to_x_map==y_olds(3,j),1);
        v = [id*2+2 id*2+3];
        [e, E_r, E_l] = project(x(r), x(v));
        E_rl = [E_r E_l];
```

```
        rl   = [r v];
        E    = E_rl * P(rl,rl) * E_rl';

        % measurement
        yi_1 = y_olds(:,j);
        yi1 = yi_1(1:2,1);

        % innovation
        z = yi1 - e;
        if z(2) > pi
            z(2) = z(2) - 2*pi;
        end
        if z(2) < -pi
            z(2) = z(2) + 2*pi;
        end
        Z = M + E;

        % Kalman gain
        K = P(:, rl) * E_rl' * Z^-1;

        % update
        x = x + K * z;
        P = P - K * Z * K';
    end

        % TODO Check if there is new landmarks and addition to the map

    % 3 State augmentation
     % Each big cycle will expand the state and add a new waypoint
     %state quantity; if you wait until all the waypoint points have
been initialized,
     %the initialization part will not be executed.
     end_new = find(y_news(1,:)==101,1);
    if isempty(end_new)
        end_new=size(y_news,2)+1;
    end
    for m1 = 1:(end_new-1)
        if isempty(m1)
            break
        end
        id = find(id_to_x_map==0,1);
        id_to_x_map(id) = y_news(3,m1);

        % measurement
        yi_2 = y_news(:,m1);
        yi2 = yi_2(1:2,1);
        [x(s), L_r, L_y] = backProject(x(r ), yi2);
        P(s,:) = L_r * P(r,:);
        P(:,s) = P(s,:)';
        P(s,s) = L_r * P(r,r) * L_r' + L_y * M * L_y';
        s = s + [2 2];
    end

     %4 getting information
```

```
     %poses info
    poses(1,t) = x(1);
    poses(2,t) = x(2);
    poses(3,t) = x(3);
    poses_(1,t) = R(1);
    poses_(2,t) = R(2);
    poses_(3,t) = R(3);
    %

     % Graphics

     % Robot simulation position snd sensor detection range
    set(RG, 'xdata', R(1), 'ydata', R(2));
    circle_x = linspace((R(1)-0.9999*sensor_r),
(R(1)+0.9999*sensor_r));
    circle_y1 = sqrt(sensor_r^2 - (circle_x - R(1)).^2) + R(2);
    circle_y2 = R(2) - sqrt(sensor_r^2 - (circle_x - R(1)).^2);
    set(sensor1,'xdata',circle_x,'ydata',circle_y1);
    set(sensor2,'xdata',circle_x,'ydata',circle_y2);

    % Detection range
    set(rG, 'xdata', x(r(1)), 'ydata', x(r(2)));
    Circle_x = linspace((x(r(1))-0.9999*sensor_r),
(x(r(1))+0.9999*sensor_r));
    Circle_y1 = sqrt(sensor_r^2 - (Circle_x - x(r(1))).^2) + x(r(2));
    Circle_y2 = x(r(2)) - sqrt(sensor_r^2 - (Circle_x - x(r(1))).^2);
     %set(Sensor1,'xdata',Circle_x,'ydata',Circle_y1);
     %set(Sensor2,'xdata',Circle_x,'ydata',Circle_y2);

    % Location trajectory
    set(estimate_pose,'xdata',poses(1,1:t),'ydata',poses(2,1:t));
    set(true_pose,'xdata',poses_(1,1:t),'ydata',poses_(2,1:t));

    legend([estimate_pose true_pose lG WG],
{'Estimate','Truth' 'Estimate landmark' 'True landmark'})

    % If there is no state augmentation for the first time,
 immediately return to the next cycle
  if s(1)==4
        continue
  end

  % Estimated location of landmarks
  w = 2:((s(1)-2)/2);
  w = 2*w;
  lx = x(w);
  ly = x(w+1);
  set(lG, 'xdata', lx, 'ydata', ly);

% Draw the covariance ellipse of estimated landmark points
   % The estimated road signs are divided into three types:
   % 1: Just discovered
   % 2: met before, met again now
   % 3: Encountered before, but not currently
```

```
%First type: one just discovered ( blue)
  for g1 = 1:(end_new-1)
      if isempty(g1)
            break
      end
      o1 = y_news(3,g1);
      h1 = find(id_to_x_map==o1,1);
      temp1 = [2*h1+2;2*h1+3];
      le = x(temp1);
      LE = P(temp1,temp1);
      [X,Y] = cov2elli(le,LE,3,16);
      set(eG1(o1),'xdata',X,'ydata',Y,'color','b');
  end
  % Second kind: met in between, met again now (red)
  for g2 = 1:(end_old-1)
      if isempty(g2)
            break
      end
      o2 = y_olds(3,g2);
      h2 = find(id_to_x_map==o2,1);
      temp2 = [2*h2+2;2*h2+3];
      le = x(temp2);
      LE = P(temp2,temp2);
      [X,Y] = cov2elli(le,LE,3,16);
      set(eG1(o2),'xdata',X,'ydata',Y,'color','r');
  end
  % The third type: I have met before, but I have not met now (black)
  v = find(id_to_x_map==0,1);
  if isempty(v)
      v = size(id_to_x_map,2)+1;
  end
  for g3 = 1:v-1
      if isempty(g3)
            break
      end
      a = find(y_olds(3,:)==id_to_x_map(g3),1);
      b = find(y_news(3,:)==id_to_x_map(g3),1);
      if (isempty (a)) && (isempty(b))
         temp3 =  [2*g3+2;2*g3+3];
            le = x(temp3);
      LE = P(temp3,temp3);
      [X,Y] = cov2elli(le,LE,3,16);
      set(eG1(id_to_x_map(g3)),'xdata',X,'ydata',Y,'color','k');
      end
  end

% Estimated robot position covariance ellipse (red)
     if t > 1
         re = x(r(1:2));
         RE = P(r(1:2),r(1:2));
         [X,Y] = cov2elli(re,RE,3,16);
         set(reG,'xdata',X,'ydata',Y);
```

```
    end
  drawnow;
  pause(0.1);
end
```



Published with MATLAB® R2020b