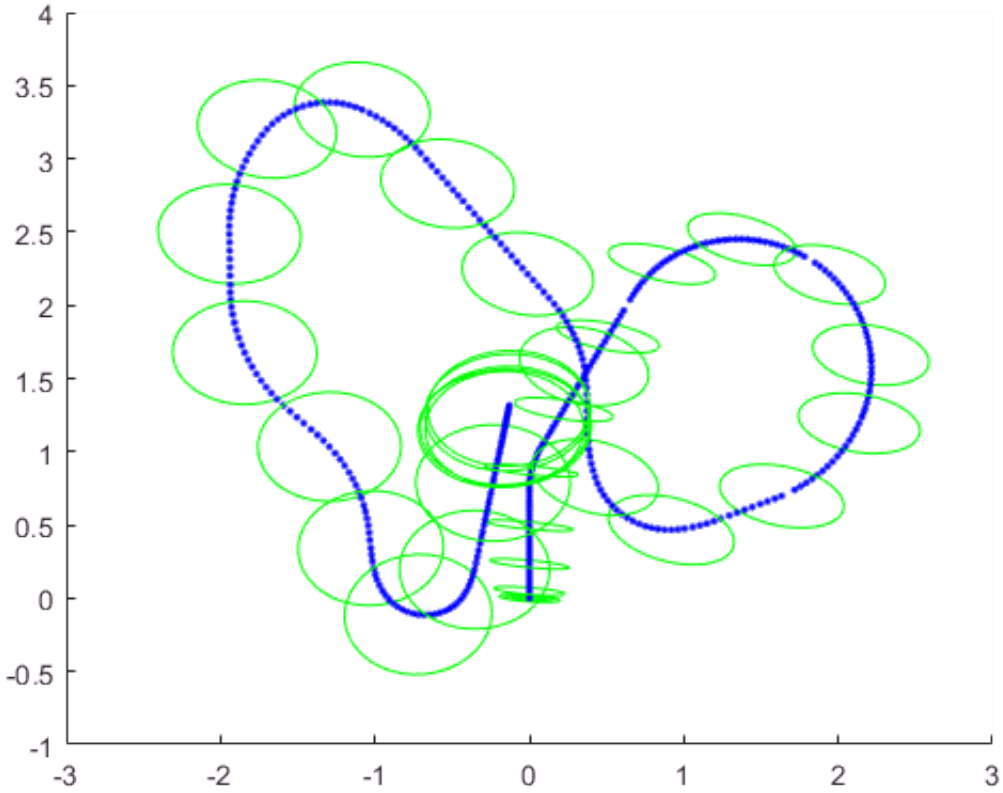
## LOCALIZATION LAB

**1. Pose estimated.** A figure of a noisy trajectory with the ellipses representing the covariance error in position. Make a zoom in to see in detail. Add to the report the commented code you implemented.

```
tita = pi/2;
xt = 0;
yt = 0;

for index=2:522
    % calculo de odometria
    R = data_enc(index, 7) - data_enc(index-1, 7);
    L = data_enc(index, 6) - data_enc(index-1, 6);
    Sc = (( R + L)/2)/1000; % (Right + Left) / 2 y lo pasamos a metros
    titac = ( R - L )/(width); % (Right - Left) / 2*S

    xt = xt + Sc*cos(tita);
    yt = yt + Sc*sin(tita);

    tita =  mod(titac + tita,2*pi);

    traject_calc(index-1,:) = [xt, yt, tita];

    % FIN calculo odometria

    % ploting the calculated trajectory
    hold on;
    plot (traject_calc(index-1,1), traject_calc(index-1,2), 'b.','LineWidth',1.5) % Plotti
    if (mod(index,15)==0)
        plot_ellipse(pk.signals.values(1:2,1:2,index),[trajec(index,1), trajec(index,2)],'
    end
end
hold off
```

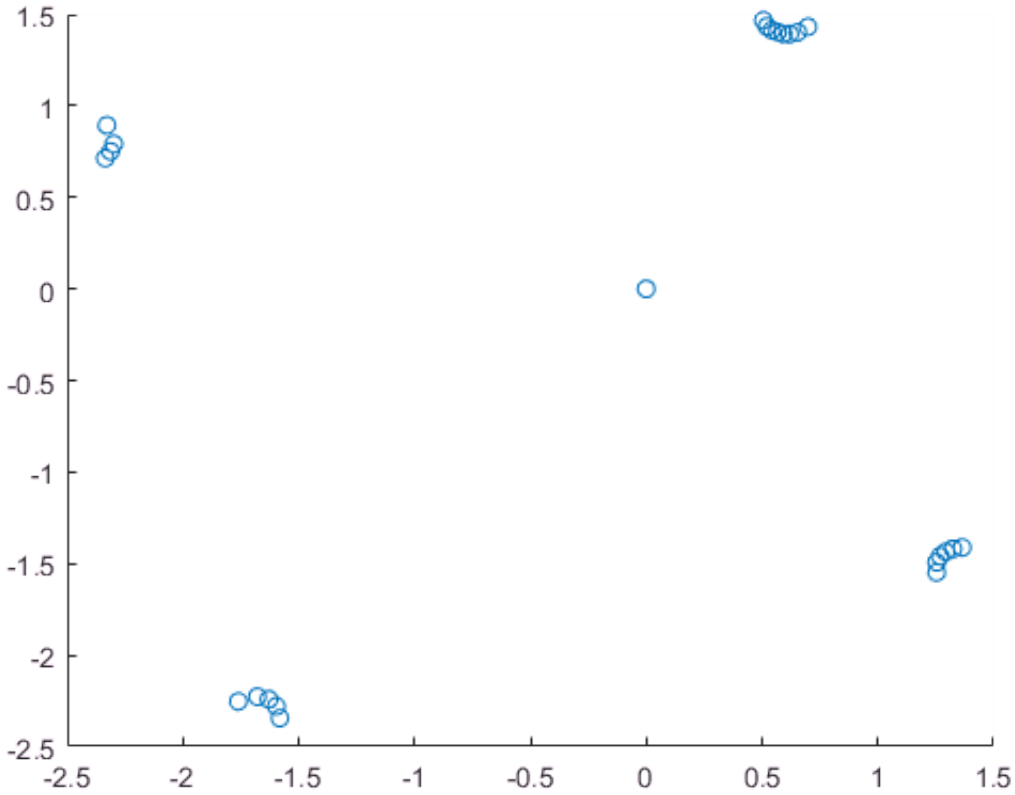



***2. Polar 2 Cartesian.*** A figure of the Land Mark seeing in Robot Reference Frame.

```
    entrada = inputdlg('Enter step time to visualize',... %Introducing the snapshot to visualiz
    'Input', [1 20]);
    temps = str2double(entrada{:})
```

```
temps = 500
```

```
    % Polar2cartesian
    for index=1:522
        for i=1:360
            xt(index,i) = (cosd(i) * lds_dis(index,i+1) )/1000; % y pasamos a metros
            yt(index,i) = (sind(i) * lds_dis(index,i+1) )/1000; % y pasamos a metros
        end
        % FIN Polar2cartesian
    end
    scatter(xt(temps,:),yt(temps,:));
```

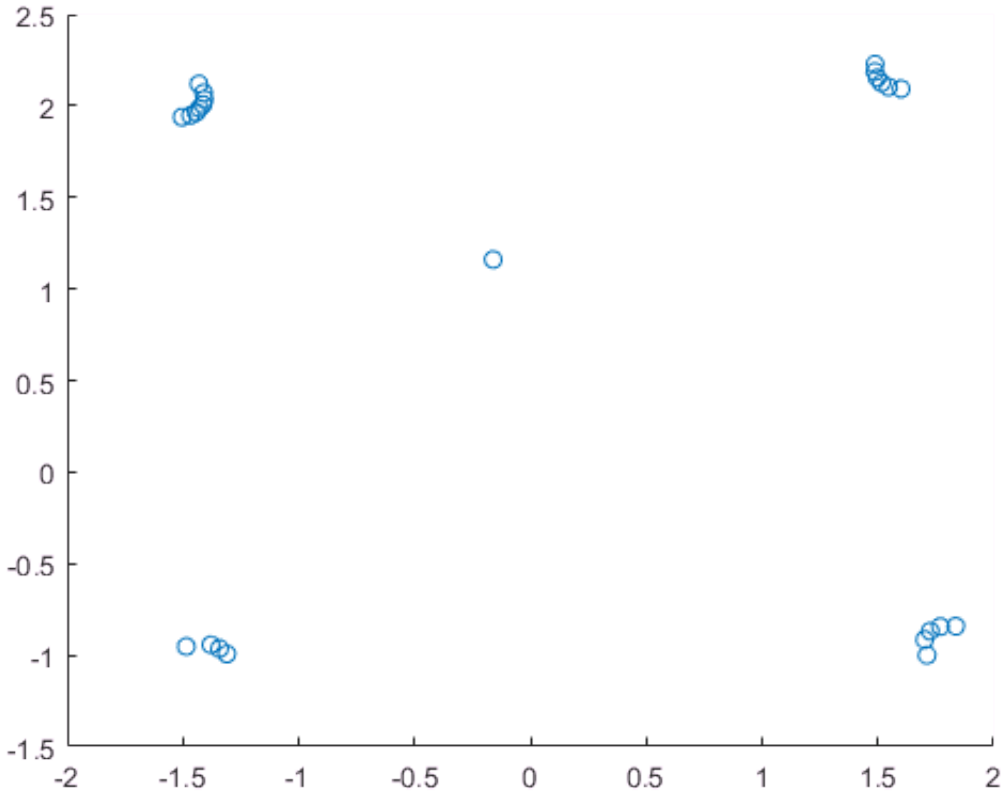



**3. Robot to World.** Generate the workspace 'laserW' variable and include in the report a figure of the Land Mark seeing in World Reference Frame.

```
entrada = inputdlg('Enter step time to visualize',... %Introducing the snapshot to visualiz
'Input', [1 20]);
temps = str2double(entrada{:})
```

```
temps = 500
```

```
for index=1:521
     for i=1:360
         tita_ini = traject_calc(index,3);
         pos_act_x = traject_calc(index,1);
         pos_act_y = traject_calc(index,2);
         tst1 = transl2(pos_act_x ,pos_act_y )*trot2(tita_ini)*[xt(index,i),yt(index,i),1]'
         tst1 = tst1';
         ldx_calc(index,i) = tst1(1,1);
         ldy_calc(index,i) = tst1(1,2);
     end
 end
 scatter(ldx_calc(temps,:),ldy_calc(temps,:));
 hold off;
```



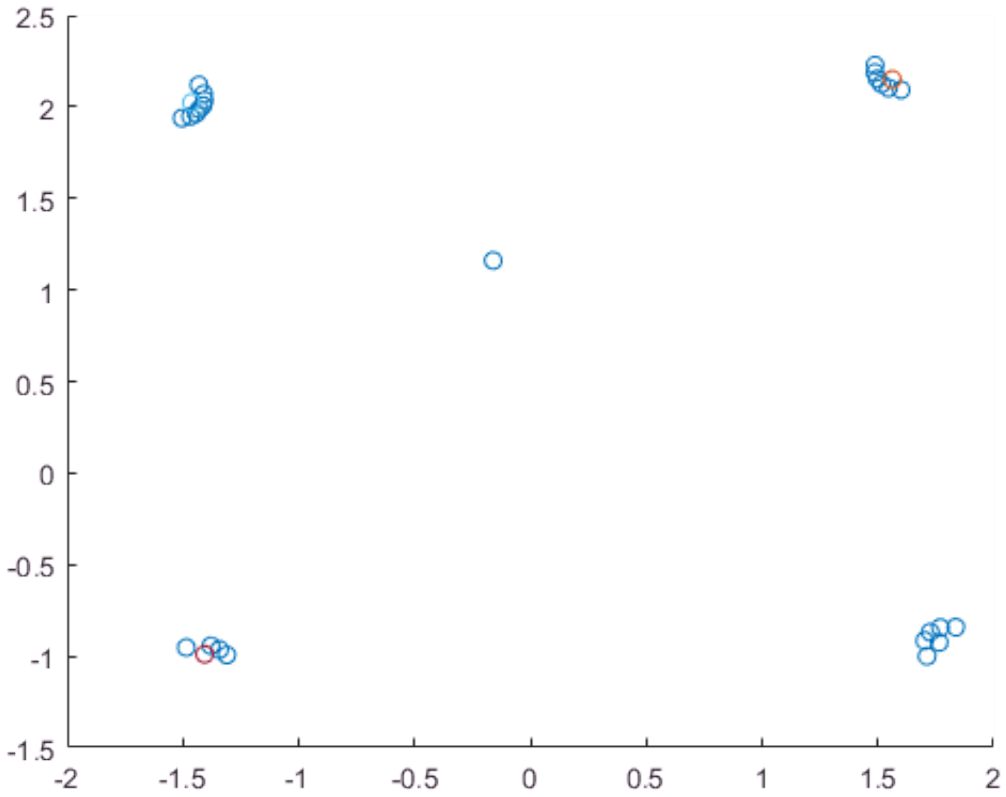

**4. Associated Land Mark.** Filter out the lidar data by detecting the landMark (datacloud) and Identifying the LandMark (nearest_to). Add to the report a figure with colored Land Mark seeing by the Robot.

```
    entrada = inputdlg('Enter step time to visualize',... %Introducing the snapshot to visuali
   'Input', [1 20]);
    temps = str2double(entrada{:})
```

```
temps = 500
```

```
    %Filtering landmarks
    for k=1:length(lds_dis)
        array = lds_dis(k,2:361);
        edgeArray = diff([0; (array(:) ~= 0); 0]);
        indices = [find(edgeArray > 0)-1 find(edgeArray < 0)]-1;
        j=1;
        i=1;
        while i<=length(indices)
            ini = indices(i,1)+2;
            while (i+1 <= length(indices)) && (indices(i,2)==indices(i+1,1) || indices(i,2)+1=
                i = i + 1;
            end
            fin = indices(i,2);
            x = median(nonzeros(ldx(k,ini:fin)));
            y = median(nonzeros(ldy(k,ini:fin)));
            %landmarks{i}{j}(1:2) contiene las coordenadas del landmark detectadas por el lase
            landmarks{k}{j} = [x y];
            j = j + 1;
            i = i + 1;
```

```
    end
end

%Finding associated landmark and printing it as seen by the robot
for t=1:522
    for l=1:length(landmarks{t})
        coordx = landmarks{t}{l}(1);
        coordy = landmarks{t}{l}(2);

        distances = sqrt(sum(bsxfun(@minus, LandMark, landmarks{t}{l}).^2,2));
        %landmarks{i}{j}(3:4) contiene el landmark asociado a las coordenadas de landmarks
        landmarks{t}{l}(3:4) = LandMark(find(distances==min(distances),1),:);
    end
end
for l=1:length(landmarks{temps})
    hold on;
    scatter(landmarks{temps}{l}(1),landmarks{temps}{l}(2));
end
hold off;
```




**5. Similarity Transform.** Adapt the Similarity Transform to output the error in pose given a time.

```
%Similarity Transform, error calculation
Error = [];
for t=1:522
    LandMark = [];
    detected = [];
    for m=1:length(landmarks{t})
        LandMark = [ LandMark'; landmarks{t}{m}(3:4)]';
```

```
            detected = [detected'; landmarks{t}{m}(1:2)]';
        end

        %Build Matrix A
        A = [];
        for i=1:size( LandMark , 2)
         A = [A;[ LandMark(1,i), LandMark(2,i),1,0]];
         A = [A;[ LandMark(2,i),-LandMark(1,i),0,1]];
        end
        B = [];%Build Matrix B
        for i=1:size( detected , 2)
         B = [B; detected(1,i); detected(2,i)];
        end
        %Compute tx ty i tita
        X = pinv(A'*A)*A'*B;
        Error(t,1) = X(3);
        Error(t,2)= X(4);
        Error(t,3) = atan2(X(2),X(1))*180/pi;
    end
```