Familiarization with odometry data

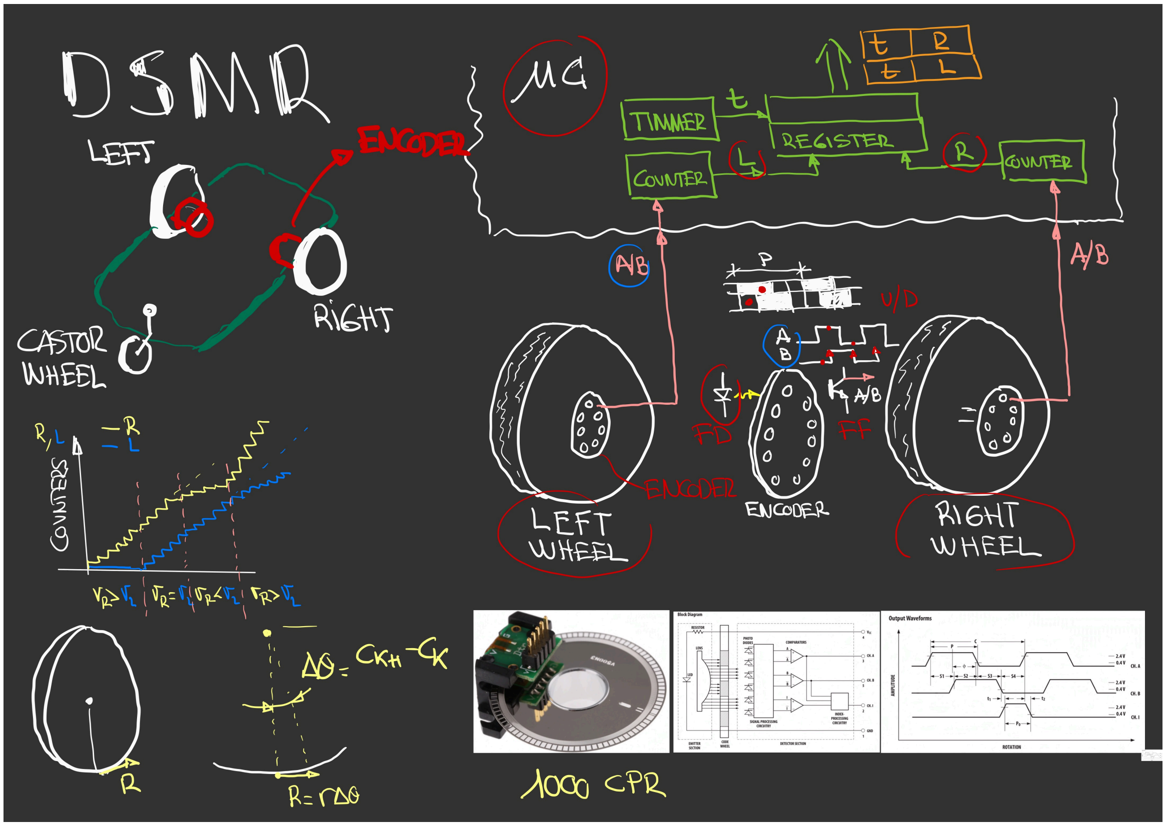


Table of Contents

Load and visualize data

```
clear
```

```
load('Encoder_Data.mat')
```

R_acu & L_acu: The microcontroller counter report; firts column time stamp, second column the total displacement of right and left wheel in meters.

```
R_acu
```

```
R_acu = 3004×2
         0         0
    0.0200         0
    0.0400         0
    0.0600         0
    0.0800         0
    0.1000         0
    0.1200         0
    0.1400         0
    0.1600         0
    0.1800         0
      ⋮
```

```
L_acu
```

```
L_acu = 3004×2
         0         0
    0.0200         0
    0.0400         0
    0.0600         0
    0.0800         0
    0.1000         0
    0.1200         0
    0.1400         0
    0.1600         0
    0.1800         0
      ⋮
```

Ts & Tf : Sample time and total experiment time (Tf=Numbers of rows *Ts) in [s]

```
Ts
```

```
Ts = 0.0200
```

```
Tf
```

```
Tf = 60.0800
```

```
Tf=length(R_acu(:,2))*Ts
```

```
Tf = 60.0800
```

```
ts=diff(R_acu(:,1)) % it is constant
```

```
ts = 3003×1
    0.0200
    0.0200
    0.0200
    0.0200
    0.0200
    0.0200
```

```
    0.0200
    0.0200
    0.0200
    0.0200
      ⋮
```

r_w & W: Mobile Robot paramenters; wheel radius and distance between wheels in [m]

```
r_w
```

```
r_w = 0.0947
```

```
W
```

```
W = 0.5200
```

```
W % 2S
```

```
W = 0.5200
```

Plotting encoders data with respect time

```
t=R_acu(:,1)
```

```
t = 3004×1
         0
    0.0200
    0.0400
    0.0600
    0.0800
    0.1000
    0.1200
    0.1400
    0.1600
    0.1800
      ⋮
```

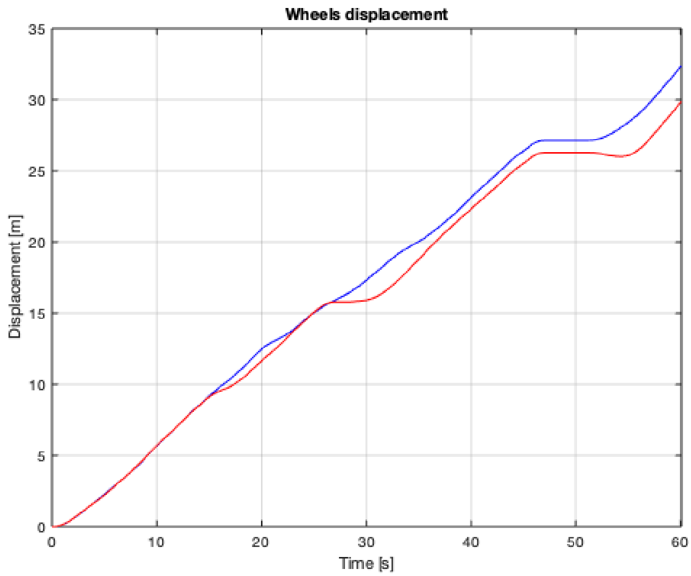
```
t=(0:Ts:Tf-Ts)'
```

```
t = 3004×1
         0
    0.0200
    0.0400
    0.0600
    0.0800
    0.1000
    0.1200
    0.1400
    0.1600
    0.1800
      ⋮
```

Total wheel displacements profile

```
figure
plot(t,R_acu(:,2),'b')
hold on
```

```
plot(t,L_acu(:,2),'r')
xlim ([0 Tf])
grid on
title('Wheels displacement')
xlabel ('Time [s]')
ylabel ('Displacement [m]')
```



Wheel incremental displacements

It tell us how much displacement did the wheel during a sample time.

$$R_{\text{inc}} = R_{k+1} - R_k$$

$$L_{\text{inc}} = L_{k+1} - L_k$$

```
R_inc=diff(R_acu(:,2))
```

```
R_inc = 3003×1
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     ⋮
```

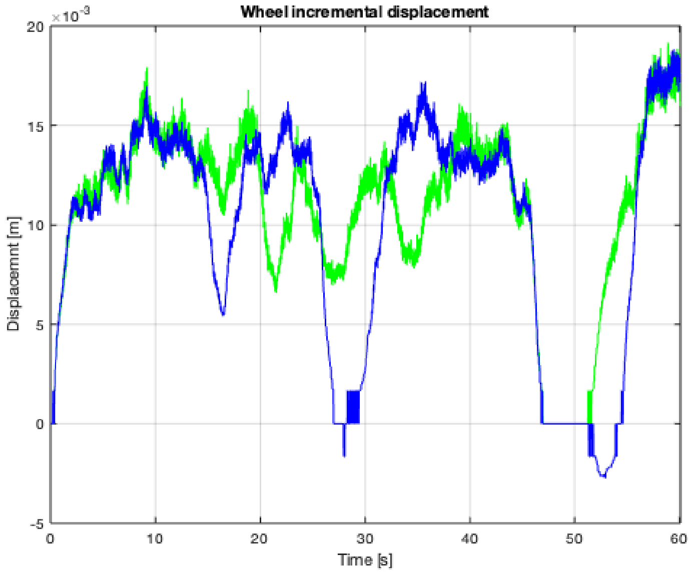
```
L_inc=diff(L_acu(:,2))
```

```
L_inc = 3003×1
     0
     0
     0
```

```
0
0
0
0
0
0
0
⋮
```

Incremental displacements profile

```
figure
plot(t(1:end-1),R_inc,'g') % We losted one sample with 'diff' command
hold on
plot(t(1:end-1),L_inc,'b') % We losted one sample with 'diff' command
hold on
xlim ([0 Tf])
grid on
title('Wheel incremental displacement')
xlabel ('Time [s]')
ylabel ('Displacemnt [m]')
```

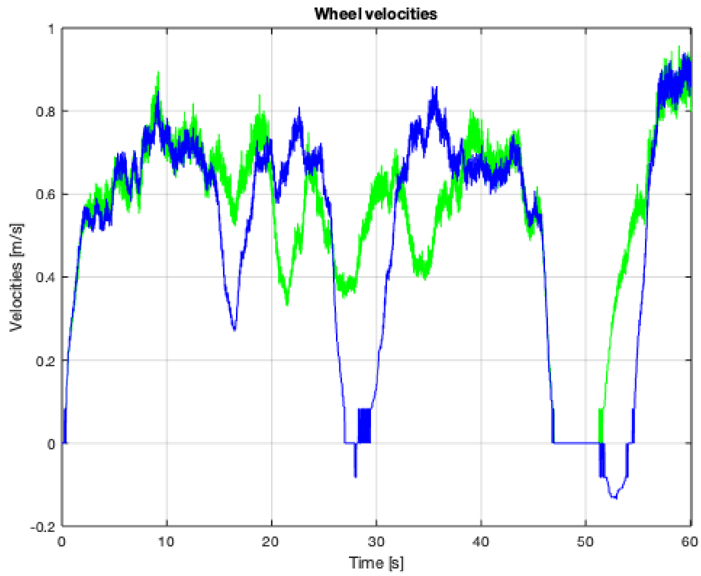


Wheel velocities profile

$$\text{Velocity} = \frac{\Delta\text{displacemts}}{\Delta\text{time}} = \frac{\Delta e}{\Delta t} = \frac{\Delta R}{T_s} = \frac{R_{\text{inc}}}{T_s}$$

```
figure
plot(t(1:end-1),R_inc/Ts,'g')
hold on
plot(t(1:end-1),L_inc/Ts,'b')
xlim ([0 Tf])
grid on
title('Wheel velocities')
xlabel ('Time [s]')
```

```
ylabel ('Velocities [m/s]')
```



Equivalence of Encoder Data

Some time the microcontroller gives wheels increment displacement. To recover total wheel displacement

$$R_{\text{acu}_k} = \int_0^{t_k} R_{\text{inc}}(t)\text{dt} \equiv \sum_i^k R_{\text{inc}_i}$$

```
R_ac=cumsum(R_inc,1)
```

```
R_ac = 3003×1
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      ⋮
```

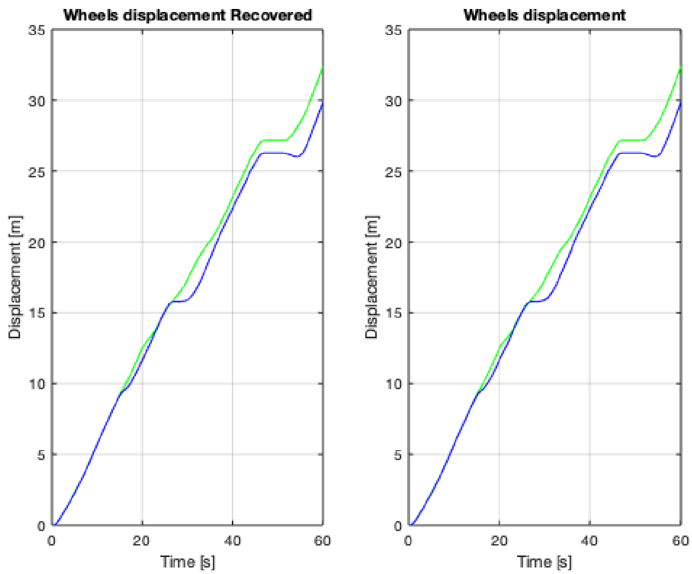
```
L_ac=cumsum(L_inc,1)
```

```
L_ac = 3003×1
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
```

⋮

Recovered wheel displacements profile

```
figure
subplot(1,2,1)
plot(t(1:end-1),R_ac,'g')
hold on
plot(t(1:end-1),L_ac,'b')
xlim ([0 Tf])
grid on
title('Wheels displacement Recovered')
xlabel ('Time [s]')
ylabel ('Displacement [m]')
subplot(1,2,2)
plot(t,R_acu(:,2),'g')
hold on
plot(t,L_acu(:,2),'b')
xlim ([0 Tf])
grid on
title('Wheels displacement')
xlabel ('Time [s]')
ylabel ('Displacement [m]')
```



Odometry

$$\delta_d = \frac{R_{inc} + L_{inc}}{2}$$

$$\delta_\theta = \frac{R_{inc} - L_{inc}}{2S}$$

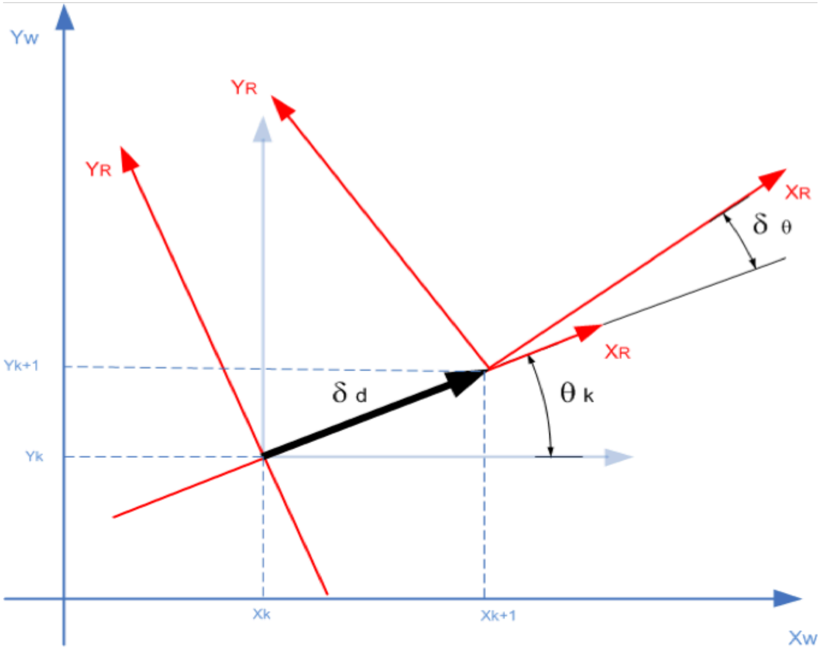
```
delta_d=(R_inc+L_inc)/2
```

```
delta_d = 3003×1
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      ⋮
```

```
delta_t=(R_inc-L_inc)/W
```

```
delta_t = 3003×1
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      ⋮
```

Pose integration



Using post multiplication

Pose; $\xi_k = \begin{pmatrix} c\theta_k & -s\theta_k & x_k \\ s\theta_k & c\theta_k & y_k \\ 0 & 0 & 1 \end{pmatrix}$

Next pose; $\xi_{k+1} = \xi_k \tan sl_x(\delta_d) Rot_Z(\delta_\theta)$

```
Initial_pose=transl(8.65,17.2,0)*trotz(-pi/2)
```

```
Initial_pose = 4×4
         0    1.0000         0    8.6500
   -1.0000         0         0   17.2000
         0         0    1.0000         0
         0         0         0    1.0000
```

```
Pose(:,:,1)=Initial_pose;
 for i=1:length(t)-1
    Pose(:,:,i+1)=Pose(:,:,i)*transl(delta_d(i),0,0)*trotz(delta_t(i));
    Position(:,i+1)=transl(Pose(:,:,i));
    Orientation(:,i+1)=tr2rpy(Pose(:,:,i));
 end
```

or using

$$\xi_{k+1} = \begin{pmatrix} p_{k+1} \\ \theta_{k+1} \end{pmatrix} = \begin{pmatrix} x_k + \delta_d c\theta_k \\ y_k + \delta_d s\theta_k \\ \theta_k + \delta_\theta \end{pmatrix}$$

```
 Initial_position=transl(Initial_pose)
```

```
Initial_position = 3×1
    8.6500
   17.2000
         0
```

```
 Initial_orientation=-pi/2
```

```
Initial_orientation = -1.5708
```

```
 x(1)=Initial_position(1)+0.05 % for comparing reasons we offset x by 5cm
```

```
x = 8.7000
```

```
 y(1)=Initial_position(2)
```

```
y = 17.2000
```

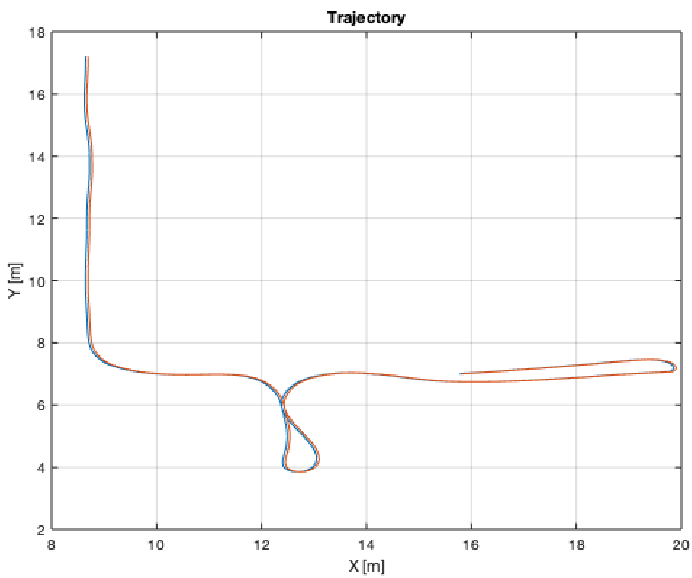
```
 o(1)=Initial_orientation
```

```
o = -1.5708
```

```
  for i=1:(length(t)-1)
     x(i+1)= x(i)+delta_d(i)*cos(o(i));
     y(i+1)= y(i)+delta_d(i)*sin(o(i));
     o(i+1)=o(i)+delta_t(i);
     end
```

Displaying trajectory

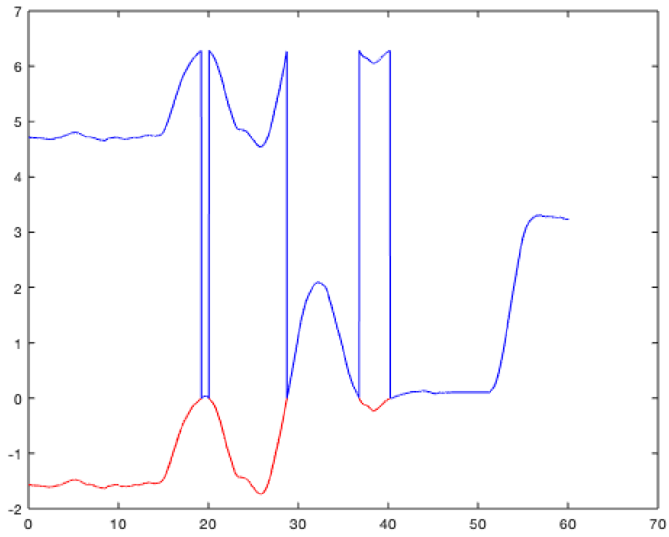
```
figure
plot(Position(1,2:end),Position(2,2:end))
grid on
hold on
title('Trajectory')
xlabel ('X [m]')
ylabel ('Y [m]')
plot(x,y)
```



Understanding Orientation

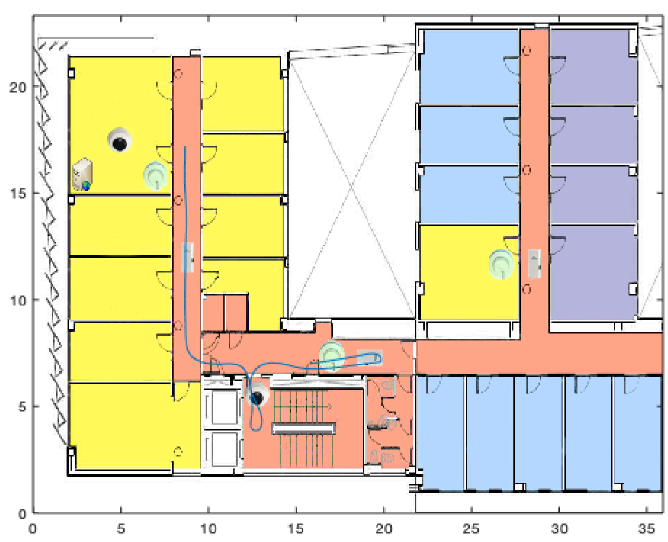
Think about $\sin\left(-\frac{\pi}{2}\right) = \sin\left(\frac{3}{2}\pi\right) = \sin\left(2\pi + \frac{3}{2}\pi\right)$

```
figure
plot(t,o,'r')
hold on
plot(t,mod(o,2*pi),'b')
```



Plotting the enviroment and trajectory

```
figure
I=imread('Enviroment.png');
x_ima=[0 35.9];
y_ima=[23.31 0];
image(I,'XData',x_ima,'YData',y_ima);
axis xy
hold on
plot(Position(1,2:end),Position(2,2:end))
```



Uncertanty: Adding noise

$$\delta_d = \frac{R+L}{2} + \nu_d$$

$$\delta_\theta = \frac{R-L}{2S} + \nu_\theta$$

Noise in the odometry displacement

```
delta_d_n=((R_inc+L_inc)/2)+randn((length(t)-1),1)*0.005
```

```
delta_d_n = 3003×1
    0.0026
    0.0096
   -0.0043
    0.0074
   -0.0017
    0.0017
   -0.0082
   -0.0059
    0.0023
    0.0042
      ⋮
```

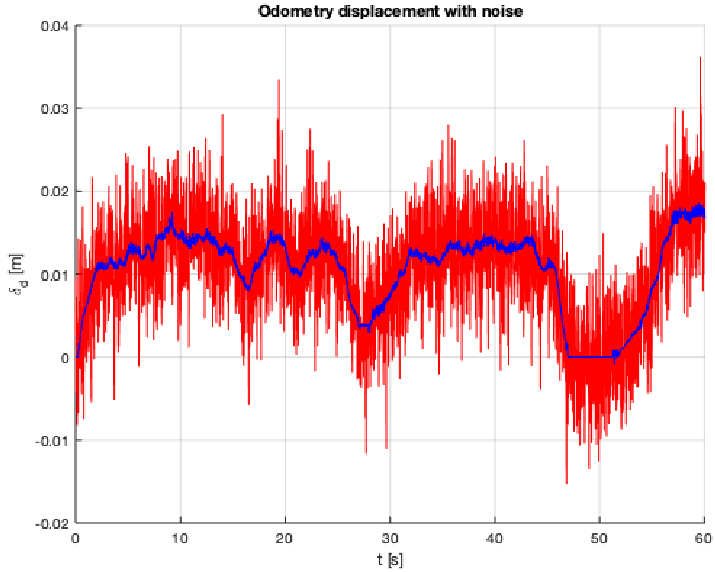
Noise in the odometry change of orientation

```
delta_t_n=((R_inc-L_inc)/W)+randn((length(t)-1),1)*0.005
```

```
delta_t_n = 3003×1
    0.0035
   -0.0028
    0.0059
   -0.0005
    0.0059
    0.0069
   -0.0016
   -0.0008
    0.0022
   -0.0053
      ⋮
```

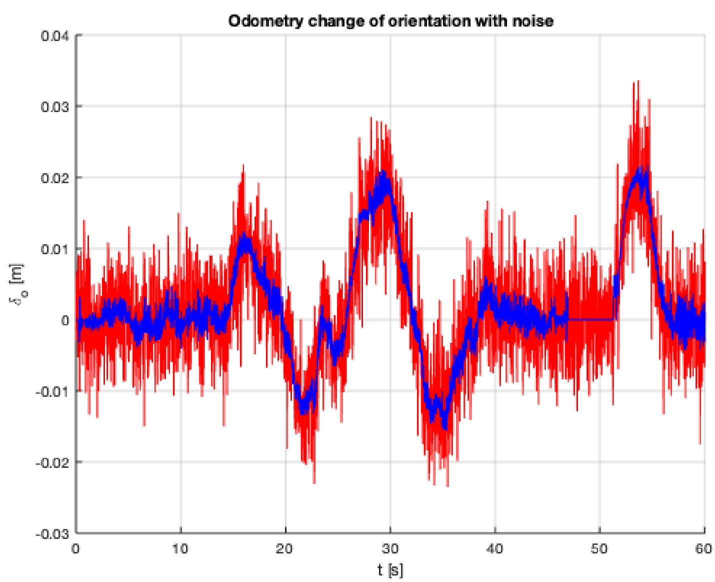
Displacement noise visualization

```
figure
hold on
plot(t(1:end-1),delta_d_n,'r')
xlim ([0 Tf])
grid on
title('Odometry displacement with noise')
xlabel ('t [s]')
ylabel ('\delta_d [m]')
plot(t(1:end-1),delta_d,'b')
```



Orientation noise visualization

```
figure
hold on
plot(t(1:end-1),delta_t_n,'r')
xlim ([0 Tf])
grid on
title('Odometry change of orientation with noise')
xlabel ('t [s]')
ylabel ('\delta_o [m]')
plot(t(1:end-1),delta_t,'b')
```



Pose integration with noise

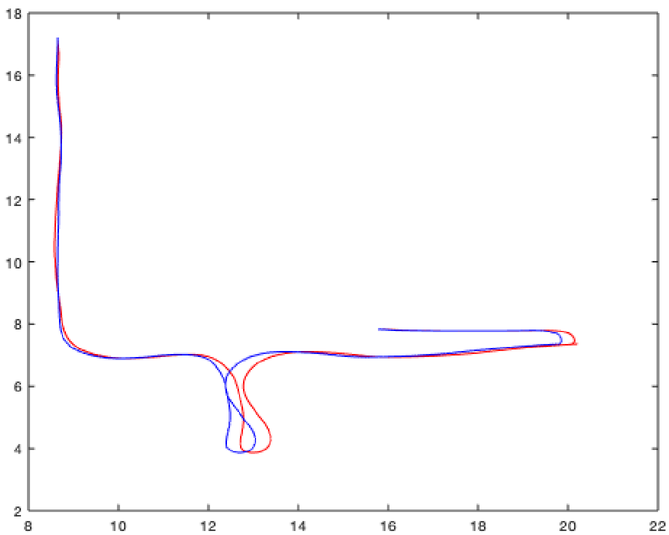
```
Initial_pose=transl(8.65,17.2,0)*trotz(-pi/2)
```

```
Initial_pose = 4×4
         0    1.0000         0    8.6500
   -1.0000         0         0   17.2000
         0         0    1.0000         0
         0         0         0    1.0000
```

```
Pose_n(:,:,1)=Initial_pose;
 for i=1:length(t)-1
    Pose_n(:,:,i+1)=Pose_n(:,:,i)*transl(delta_d_n(i),0,0)*trotz(delta_t_n(i));
    Position_n(:,i+1)=transl(Pose_n(:,:,i));
    Orientation_n(:,i+1)=tr2rpy(Pose_n(:,:,i));
 end
```

Comparing trajectories

```
 figure
 plot(Position_n(1,2:end),Position_n(2,2:end),'r')
 hold on
 plot(Position(1,2:end),Position_n(2,2:end),'b')
```



Ellipse error

It is the of interest to launch many time the dices (our trajectory with noise) and check for the las position and orientation

```
Initial_pose=transl(8.65,17.2,0)*trotz(-pi/2)
```

```
Initial_pose = 4×4
         0    1.0000         0    8.6500
   -1.0000         0         0   17.2000
         0         0    1.0000         0
         0         0         0    1.0000
```

```
Pose_n(:,:,1)=Initial_pose;
for j=1:1000
```

```
delta_d_n=((R_inc+L_inc)/2)+randn((length(t)-1),1)*0.005;
delta_t_n=((R_inc-L_inc)/W)+randn((length(t)-1),1)*0.005;
 for i=1:length(t)-1
    Pose_n(:,:,i+1)=Pose_n(:,:,i)*transl(delta_d_n(i),0,0)*trotz(delta_t_n(i));
    Position_n(:,i+1)=transl(Pose_n(:,:,i));
    Orientation_n(:,i+1)=tr2rpy(Pose_n(:,:,i));
 end
 Positions_n(:,j)=Position_n(:,end);
 Orientations_n(:,j)=Orientation_n(:,end);
end
```

Visualizing the experimental ellipse error of final position

```
figure
scatter(Positions_n(1,:),Positions_n(2,:))
```