Odometry Data

Authors: Pol Puigdemont & Marc Sardà

Team: 11A

Table of Contents

Load Data

"L_acu": The first column is time, and the second column represents the distance covered to the left in 60 seconds.

"R_acu": The same as "L_acu", but for the right.

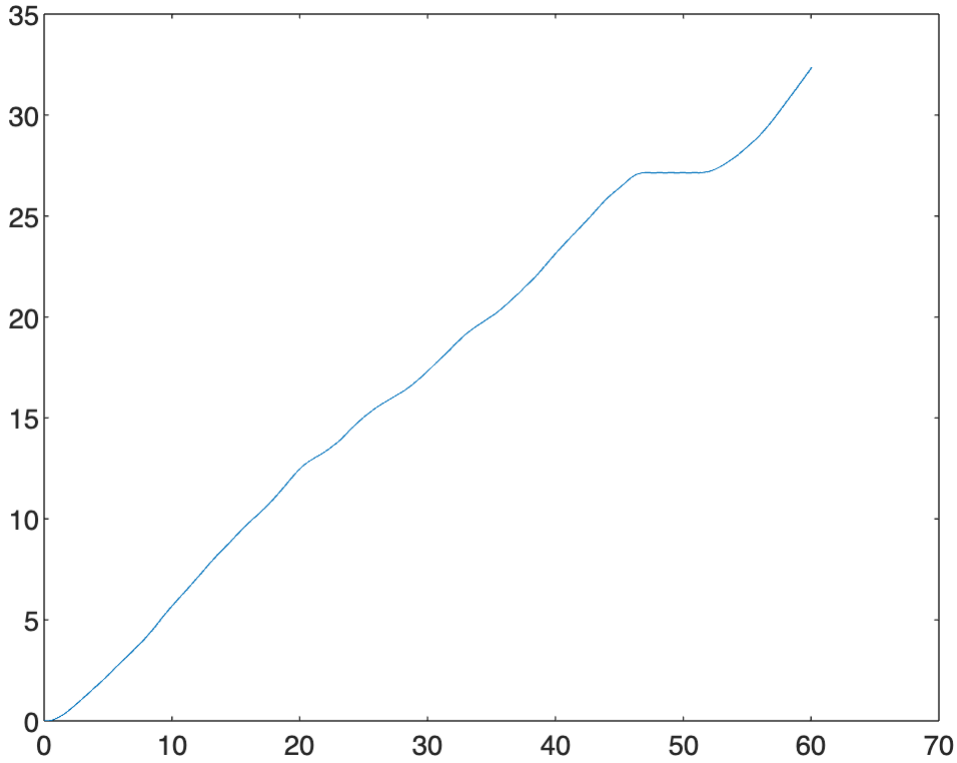
There are 3000 samples with a distance of 20 ms between samples.

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

Visualizing L&R info

Using time values, we create a graph of "L_acu" and "R_acu" over time.

```
%left_dist = L_acu(:, 2)
%right_dist = R_acu;
xlabel();
ylabel();
title('');
plot(R_acu(:, 1), R_acu(:, 2))
```



Visualize increment displacement

```
L_acu
```

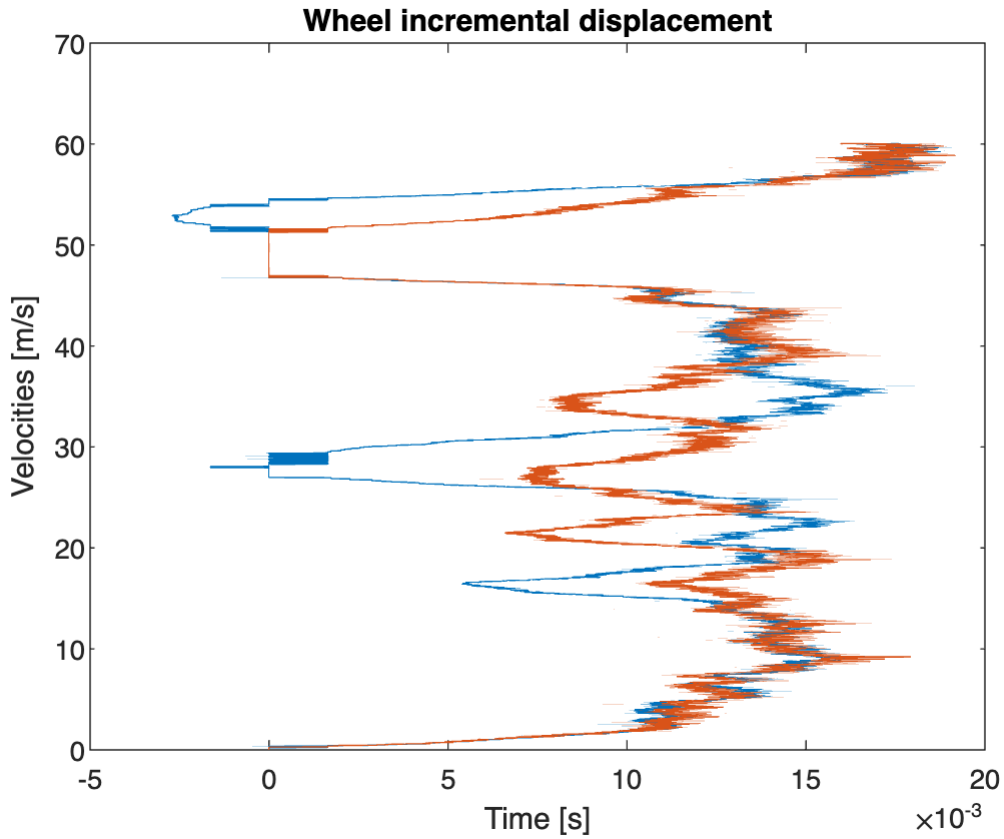
```
L_acu = 3004×2
         0         0
    0.0200         0
    0.0400         0
    0.0600         0
    0.0800         0
    0.1000         0
    0.1200         0
    0.1400         0
    0.1600         0
    0.1800         0
     ⋮
```

```
L_acu(:,2)
```

```
ans = 3004×1
     0
     0
     0
     0
     0
     0
     0
     0
     0
```

```
     0
     ⋮
```

```
l_inc = diff(L_acu(:, 2));
r_inc = diff(R_acu(:, 2));
time = L_acu(1:3003, 1);
plot([l_inc(:,1), r_inc(:, 1)], time)
title('Wheel incremental displacement')
xlabel('Time [s]')
ylabel('Velocities [m/s]')
```



Visualize increment velocity

```
plot()
```

Odometry

Compute δ_x and δ_ψ

```
delta_x = (r_inc + l_inc) / 2 % (Rinc + Linc) / 2
```

```
delta_x = 3003×1
     0
     0
     0
     0
```

```
     0
     0
     0
     0
     0
     0
     ⋮
```

```
delta_psi = (r_inc - l_inc) / W % (Rinc - Linc) / 2*S  [2*S es la separacion
entre las ruedas]
```

```
delta_psi = 3003×1
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     ⋮
```

Pose integration

Compare your results

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

```
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
```

or using

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

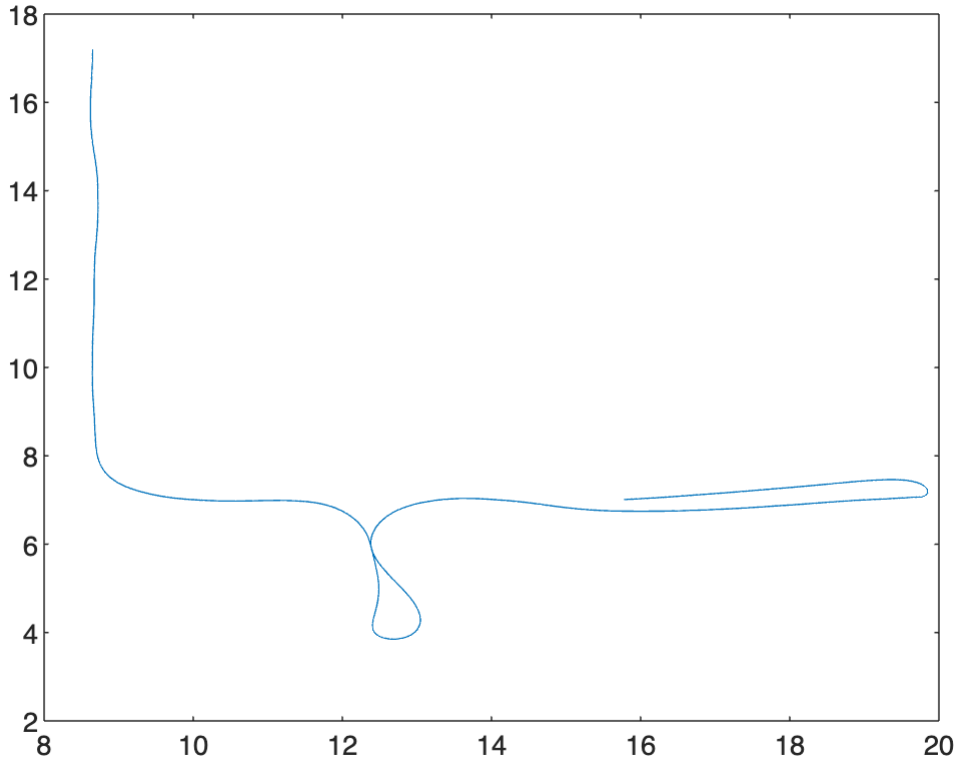
```
result_x = zeros(length(l_inc), 1);
```

```
result_y = zeros(length(l_inc), 1);

x = 8.65;
y = 17.2;
theta = -pi/2;

for i = 1:length(l_inc)
    result_x(i) = x;
    result_y(i) = y;
    x = x + delta_x(i) * cos(theta);
    y = y + delta_x(i) * sin(theta);
    theta = theta + delta_psi(i);
end
```

```
plot(result_x, result_y)
```

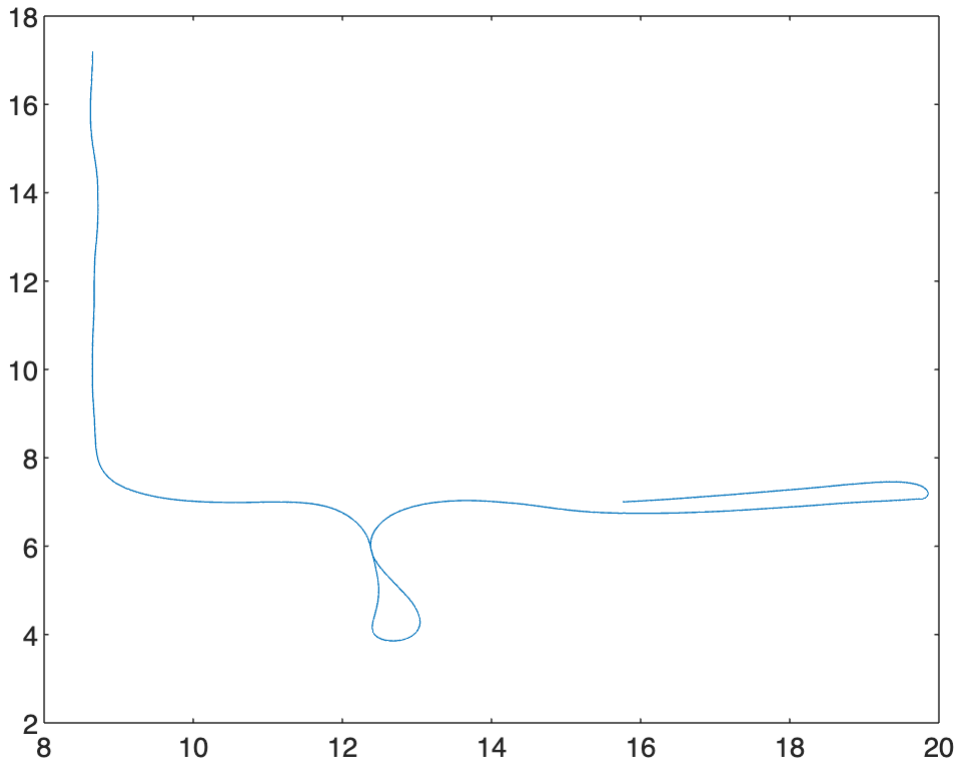



```
pose = [0 1 0 8.65; -1 0 0 17.2; 0 0 1 0; 0 0 0 1];
result_x_2 = zeros(length(l_inc), 1);
result_y_2 = zeros(length(l_inc), 1);

for i = 1:length(l_inc)
    pose = pose * trotz(delta_psi(i)) * transl(delta_x(i), 0, 0);
    result_x_2(i) = pose(1, 4);
    result_y_2(i) = pose(2, 4);
```

```
end
```

```
plot(result_x_2, result_y_2)
```



Adding noise

Add noise to odometry

See the effect on the trajectory

```
noise_range = max(delta_x) * 2;
noisy_delta_x = delta_x + randn(length(delta_x), 1) * noise_range
```

```
noisy_delta_x = 3003×1
   -0.0201
    0.0076
   -0.0125
    0.0289
   -0.0136
   -0.0352
    0.0326
   -0.0538
   -0.0162
   -0.0178
     ⋮
```

```
noisy_delta_psi = delta_psi + randn(length(delta_psi), 1) * max(delta_psi)
```

```
noisy_delta_psi = 3003×1
    0.0120
   -0.0216
    0.0130
    0.0017
    0.0144
    0.0006
   -0.0365
   -0.0418
    0.0142
   -0.0160
     ⋮
```

```
pose = [0 1 0 8.65; -1 0 0 17.2; 0 0 1 0; 0 0 0 1];
result_x_3 = zeros(length(l_inc), 1);
result_y_3 = zeros(length(l_inc), 1);

for i = 1:length(l_inc)
    pose = pose * trotz(delta_psi(i)) * transl(noisy_delta_x(i), 0, 0);
    result_x_3(i) = pose(1, 4);
    result_y_3(i) = pose(2, 4);
end
```

```
plot(result_x_3, result_y_3)
```