LAB REPORT — 2WHELLCAR

AUTHOR(S):

KIN CHANG
LIN LI
AMIR OMIDFAR
ANGEL JIMENEZ

Contents

Lab Overview:

In this lab, we use extended Kalman filter that utilize the state estimation with sensor measurements. Then, a motion planning algorithm use the state estimation result to plan and parallel park a two-wheel car. A block diagram of our system is shown below.

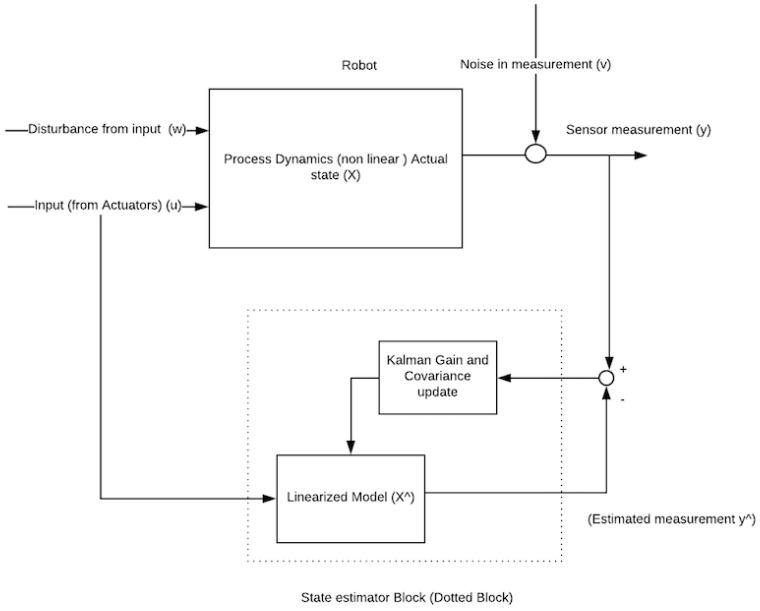

Figure 1. System Block Diagram

In the first half of this report, we will derive everything needed to do Kalman procedure, namely the linearized state-to-measurement and state evolution function. The remaining half of the report will explain the motion planning algorithm we developed.

1. Symbols

Here is the list of all symbols we used for deriving our system functions used in Kalman Filter

L_x	length of the box
L_y	width of the box
W	distance between rotating center and sensors center
C_v	coefficient, distance traveled by the wheel per unit of delay time
C_r	coefficient, body orientation changed by the wheel per unit of delay time
l_x	first range sensor reading (located on front of the car)
l_y	second range sensor reading (located on right side of the car)
α	MPU angle measurement
r_x	absolute x coordinate of the car
r_y	absolute y coordinate of the car
θ	absolute orientation of the car
τ_L	Servo input: delay time of the left wheel
τ_R	Servo input: delay time of the right wheel
θ_{tht}	top half threshold angle
θ_{thb}	bottom half threshold angle
θ_{thr}	bottom right threshold angle
θ_{thl}	bottom left threshold angle
ϵ_{str}	picking variable, 1 if going straight line, 0 otherwise
w_t	process noise
v_t	measurement noise

2. Assumption

For the sake of simplicity, we have the following assumption in the derivation below

(1) θ takes value only (-90, 90)
(2) τ_L and τ_R has the same magnitude, opposite sign if turning, otherwise same sign
(3) $\theta = \alpha$ assuming we have the the MPU calibrated at the 0 point
(4) The reference point where r_x, r_y are defined is the top right corner of the car
(5) The rotational center is the middle point between the two wheels
(6) There are no process noise: $Q_t = 0$

State

$$x = \begin{bmatrix} r_x \\ r_y \\ \theta \end{bmatrix}$$

Sensor Measurements

$$y = \begin{bmatrix} l_x \\ l_y \\ \alpha \end{bmatrix}$$

Input

$$u = \begin{bmatrix} \tau_L \\ \tau_R \end{bmatrix}$$

3. Sensor Measurement

As shown in the picture below, θ is the angle the head of the car would have with respect to the direction of L_x. In order to do the calculations required there are four more threshold angles defined as tht (top), thb (bottom), thr (right) and thl(left).

These angles are then used to find a function such that $y = h(x)$
Threshold angles can be shown using trigonometry relationships:

$$\theta_{tht} = \tan^{-1}(\frac{L_y - r_y}{L_x - r_x})$$
$$\theta_{thb} = \tan^{-1}(\frac{r_y}{L_x - r_x})$$
$$\theta_{thr} = \tan^{-1}(\frac{L_x - r_x}{r_y})$$
$$\theta_{thl} = \tan^{-1}(\frac{r_x}{r_y})$$

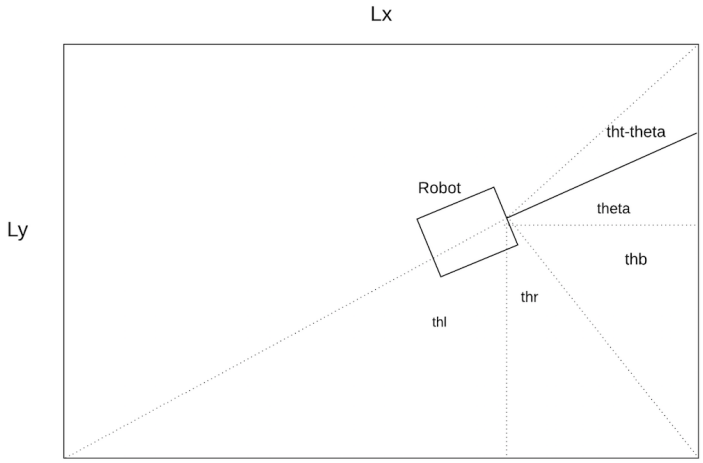

FIGURE 2. Threshold angles and theta (θ)

Under our assumption, our expression for l_x changes under these four cases:

$$l_x = \frac{L_x - r_x}{\cos|\theta|} \quad \theta > 0, |\theta| < \theta_{tht}$$

$$l_x = \frac{L_y - r_y}{\sin|\theta|} \quad \theta > 0, |\theta| > \theta_{tht}$$

$$l_x = \frac{L_x - r_x}{\cos|\theta|} \quad \theta < 0, |\theta| < \theta_{thb}$$

$$l_x = \frac{r_y}{\sin|\theta|} \quad \theta < 0, |\theta| > \theta_{thb}$$

Our expression for l_y changes under these four cases:

$$l_y = \frac{r_y}{\cos|\theta|} \quad \theta > 0, |\theta| < \theta_{thr}$$

$$l_y = \frac{L_x - r_x}{\sin|\theta|} \quad \theta > 0, |\theta| > \theta_{thr}$$

$$l_y = \frac{r_y}{\cos|\theta|} \quad \theta < 0, |\theta| < \theta_{thl}$$

$$l_y = \frac{r_x}{\sin|\theta|} \quad \theta < 0, |\theta| > \theta_{thl}$$

Our expression for $\alpha = \theta$ doesn't change under our assumption.

To summarize, there are total of 8 cases as indicated below:

$\theta \geq 0, |\theta| < \theta_{tht}, |\theta| < \theta_{thr}$

$$y = \begin{bmatrix} \frac{L_x - r_x}{\cos|\theta|} \\ \frac{r_y}{\cos|\theta|} \\ \alpha = \theta \end{bmatrix}$$

$\theta > 0, |\theta| < \theta_{tht}, |\theta| > \theta_{thr}$

$$y = \begin{bmatrix} \frac{L_x - r_x}{\cos|\theta|} \\ \frac{L_x - r_x}{\sin|\theta|} \\ \alpha = \theta \end{bmatrix}$$

$\theta > 0, |\theta| > \theta_{tht}, |\theta| < \theta_{thr}$

$$y = \begin{bmatrix} \frac{L_y - r_y}{\sin|\theta|} \\ \frac{r_y}{\cos|\theta|} \\ \alpha = \theta \end{bmatrix}$$

$\theta > 0, |\theta| > \theta_{tht}, |\theta| > \theta_{thr}$

$$y = \begin{bmatrix} \frac{L_y - r_y}{\sin|\theta|} \\ \frac{L_x - r_x}{\sin|\theta|} \\ \alpha = \theta \end{bmatrix}$$

$\theta < 0, |\theta| < \theta_{thb}, |\theta| < \theta_{thl}$

$$y = \begin{bmatrix} \frac{L_x - r_x}{\cos|\theta|} \\ \frac{r_y}{\cos|\theta|} \\ \alpha = \theta \end{bmatrix}$$

$\theta < 0, |\theta| < \theta_{thb}, |\theta| > \theta_{thl}$

$$y = \begin{bmatrix} \frac{L_x - r_x}{\cos|\theta|} \\ \frac{r_x}{\sin|\theta|} \\ \alpha = \theta \end{bmatrix}$$

$\theta < 0, |\theta| > \theta_{thb}, |\theta| < \theta_{thl}$

$$y = \begin{bmatrix} \frac{r_y}{\sin|\theta|} \\ \frac{r_y}{\cos|\theta|} \\ \alpha = \theta \end{bmatrix}$$

$\theta < 0, |\theta| > \theta_{thb}, |\theta| > \theta_{thl}$

$$y = \begin{bmatrix} \frac{r_y}{\sin|\theta|} \\ \frac{r_x}{\sin|\theta|} \\ \alpha = \theta \end{bmatrix}$$

The above expressions for y is not linear. Linearizing the above eight cases so that: $y = Hx + C$ gives us the extended Kalman Filter. As for demonstration purposes the linearization of case 1 where $\theta > 0, |\theta| < \theta_{tht}, |\theta| < \theta_{thr}$, is shown below,

$$l_x = \frac{L_x - r_x}{\cos\theta}$$
$$l_y = \frac{r_y}{\cos\theta}$$

Linearization around $l_{x0}, l_{y0}, \alpha_0 = h(r_{x0}, r_{y0}, \theta_0)$

$$l_x = l_{x0} + \frac{-1}{\cos\theta_0}(r_x - r_{x0}) + \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2}(\theta - \theta_0)$$
$$l_y = l_{y0} + \frac{1}{\cos\theta_0}(r_y - r_{y0}) + \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2}(\theta - \theta_0)$$
$$l_x = \frac{-1}{\cos\theta_0}r_x + \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2}\theta + [\frac{r_{x0}}{\cos\theta_0} - \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{x0}]$$
$$l_y = \frac{1}{\cos\theta_0}r_y + \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2}\theta + [\frac{-r_{y0}}{\cos\theta_0} - \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{y0}]$$

In matrix form, the complete form for this case is

$$y = \begin{bmatrix} \frac{-1}{\cos\theta_0} & 0 & \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2} \\ 0 & \frac{1}{\cos\theta_0} & \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{r_{x0}}{\cos\theta_0} - \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{x0} \\ \frac{-r_{y0}}{\cos\theta_0} - \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{y0} \\ 0 \end{bmatrix}$$

4. Complete Case Analysis

$\theta \geq 0, |\theta| < \theta_{tht}, |\theta| < \theta_{thr}$

$$H = \begin{bmatrix} \frac{-1}{\cos\theta_0} & 0 & \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2} \\ 0 & \frac{1}{\cos\theta_0} & \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2} \\ 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} \frac{r_{x0}}{\cos\theta_0} - \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{x0} \\ \frac{-r_{y0}}{\cos\theta_0} - \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{y0} \\ 0 \end{bmatrix}$$

$\theta \geq 0, |\theta| < \theta_{tht}, |\theta| > \theta_{thr}$

$$H = \begin{bmatrix} \frac{-1}{\cos\theta_0} & 0 & \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2} \\ \frac{-1}{\sin|\theta_0|} & 0 & -\frac{\theta_0}{|\theta_0|}\frac{L_x - r_{x0}\cos(\theta_0)}{(\sin|\theta_0|)^2} \\ 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} \frac{r_{x0}}{\cos\theta_0} - \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{x0} \\ \frac{r_{x0}}{\sin|\theta_0|} + \frac{(L_x - r_{x0})\cos(\theta_0)}{(\sin|\theta_0|)^2}|\theta_0| + l_{y0} \\ 0 \end{bmatrix}$$

$\theta > 0, |\theta| > \theta_{tht}, |\theta| < \theta_{thr}$

$$H = \begin{bmatrix} 0 & \frac{-1}{\sin|\theta_0|} & -\frac{\theta_0}{|\theta_0|}\frac{L_y - r_{y0}\cos(\theta_0)}{(\sin|\theta_0|)^2} \\ 0 & \frac{1}{\cos\theta_0} & \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2} \\ 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} \frac{r_{y0}}{\sin|\theta_0|} + \frac{(L_y - r_{y0})\cos(\theta_0)}{(\sin|\theta_0|)^2}|\theta_0| + l_{x0} \\ \frac{-r_{y0}}{\cos\theta_0} - \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{y0} \\ 0 \end{bmatrix}$$

$\theta > 0, |\theta| > \theta_{tht}, |\theta| > \theta_{thr}$

$$H = \begin{bmatrix} 0 & \frac{-1}{\sin|\theta_0|} & -\frac{\theta_0}{|\theta_0|}\frac{L_y - r_{y0}\cos(\theta_0)}{(\sin|\theta_0|)^2} \\ \frac{-1}{\sin|\theta_0|} & 0 & -\frac{\theta_0}{|\theta_0|}\frac{L_x - r_{x0}\cos(\theta_0)}{(\sin|\theta_0|)^2} \\ 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} \frac{r_{y0}}{\sin|\theta_0|} + \frac{(L_y - r_{y0})\cos(\theta_0)}{(\sin|\theta_0|)^2}|\theta_0| + l_{x0} \\ \frac{r_{x0}}{\sin|\theta_0|} + \frac{(L_x - r_{x0})\cos(\theta_0)}{(\sin|\theta_0|)^2}|\theta_0| + l_{y0} \\ 0 \end{bmatrix}$$

$\theta < 0, |\theta| < \theta_{thb}, |\theta| < \theta_{thl}$

$$H = \begin{bmatrix} \frac{-1}{\cos\theta_0} & 0 & \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2} \\ 0 & \frac{1}{\cos\theta_0} & \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2} \\ 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} \frac{r_{x0}}{\cos\theta_0} - \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{x0} \\ \frac{-r_{y0}}{\cos\theta_0} - \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{y0} \\ 0 \end{bmatrix}$$

$\theta < 0, |\theta| < \theta_{thb}, |\theta| > \theta_{thl}$

$$H = \begin{bmatrix} \frac{-1}{\cos\theta_0} & 0 & \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2} \\ \frac{1}{\sin|\theta_0|} & 0 & -\frac{\theta_0}{|\theta_0|}\frac{r_{x0}\cos(\theta_0)}{(\sin|\theta_0|)^2} \\ 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} \frac{r_{x0}}{\cos\theta_0} - \frac{(L_x - r_{x0})\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{x0} \\ \frac{-r_{x0}}{\sin|\theta_0|} + \frac{r_{x0}\cos(\theta_0)}{(\sin|\theta_0|)^2}|\theta_0| + l_{y0} \\ 0 \end{bmatrix}$$

$\theta < 0, |\theta| > \theta_{thb}, |\theta| < \theta_{thl}$

$$H = \begin{bmatrix} 0 & \frac{1}{\sin|\theta_0|} & -\frac{\theta_0}{|\theta_0|}\frac{r_{y0}\cos(\theta_0)}{(\sin|\theta_0|)^2} \\ 0 & \frac{1}{\cos\theta_0} & \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2} \\ 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} -\frac{r_{y0}}{\sin|\theta_0|} + \frac{r_{y0}\cos(\theta_0)}{(\sin|\theta_0|)^2}|\theta_0| + l_{x0} \\ \frac{-r_{y0}}{\cos\theta_0} - \frac{r_{y0}\sin(\theta_0)}{(\cos\theta_0)^2}\theta_0 + l_{y0} \\ 0 \end{bmatrix}$$

$\theta < 0, |\theta| > \theta_{thb}, |\theta| > \theta_{thl}$

$$H = \begin{bmatrix} 0 & \frac{1}{\sin|\theta_0|} & -\frac{\theta_0}{|\theta_0|}\frac{r_{y0}\cos(\theta_0)}{(\sin|\theta_0|)^2} \\ \frac{1}{\sin|\theta_0|} & 0 & -\frac{\theta_0}{|\theta_0|}\frac{r_{x0}\cos(\theta_0)}{(\sin|\theta_0|)^2} \\ 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} -\frac{r_{y0}}{\sin|\theta_0|} + \frac{r_{y0}\cos(\theta_0)}{(\sin|\theta_0|)^2}|\theta_0| + l_{x0} \\ -\frac{r_{x0}}{\sin|\theta_0|} + \frac{r_{x0}\cos(\theta_0)}{(\sin|\theta_0|)^2}|\theta_0| + l_{y0} \\ 0 \end{bmatrix}$$

5. State Evolution

The goal for this section is to find a function such that $x_{t+1} = f(x_t, u_t) = Ax_t + B(u_t)$. To begin with, there are two cases to consider:

Case 1: the car is going in a straight line (either backward or forward): $(\tau_{L,t} * \tau_{R,t}) > 0$

$$\begin{aligned} r_{x,t+1} &= r_{x,t} + C_v * \tau_{R,t}\cos\theta \\ r_{y,t+1} &= r_{y,t} + C_v * \tau_{R,t}\sin\theta \\ \theta_{t+1} &= \theta_t \end{aligned}$$

Case 2: the car is turning (either left or right): $(\tau_{L,t} * \tau_{R,t}) > 0$

Note that $r_{x,t+1} \neq r_{x,t}$ because the rotation center point is not the same as the absolute reference point. To derive the relationship, define $\theta_0 = \theta_t =$ orientation before turning, and $\theta_f = \theta_{t+1} =$ orientation after turning.

$$\begin{aligned} r_{x,t+1} &= r_{x,t} - W[\cos\theta_0 - \cos\theta_f] \\ r_{y,t+1} &= r_{y,t} + W[\sin\theta_f - \sin\theta_0] \\ \theta_{t+1} &= \theta_t + C_r\tau_{R,t} \end{aligned}$$

notice that all the signs worked out in the above expressions either going forward or backward, or turning left or right

To combine these two cases into one expression, we define a picking variable $\epsilon_{str} = \tau_{L,t} * \tau_{R,t} > 0)$ taking value 0 or 1. Then,

$$\begin{aligned} r_{x,t+1} &= r_{x,t} + \epsilon_{str}C_v\tau_{R,t}\cos\theta - (1-\epsilon_{str})W[\cos\theta_t - \cos(\theta_t + C_r\tau_{R,t})] \\ r_{y,t+1} &= r_{y,t} + \epsilon_{str}C_v\tau_{R,t}\sin\theta + (1-\epsilon_{str})W[\sin(\theta_t + C_r\tau_{R,t}) - \sin(\theta_t)] \\ \theta_{t+1} &= \theta_t + (1-\epsilon_{str}) * C_r\tau_{R,t} \end{aligned}$$

In matrix form,

$$\begin{aligned} x_{t+1} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{x,t} \\ r_{y,t} \\ \theta_t \end{bmatrix} + \begin{bmatrix} \epsilon_{str}C_v\tau_{R,t}\cos\theta - (1-\epsilon_{str})W[\cos\theta_t - \cos(\theta_t + C_r\tau_{R,t})] \\ \epsilon_{str}C_v\tau_{R,t}\sin\theta + (1-\epsilon_{str})W[\sin(\theta_t + C_r\tau_{R,t}) - \sin(\theta_t)] \\ (1-\epsilon_{str}) * C_r\tau_{R,t} \end{bmatrix} \\ x_{t+1} &= Ax_t + B(u_t) \end{aligned}$$

6. Kalmen Filter Procedure

We have everything we need to do Kalman Filter Procedure, as shown below

6.1. **Kalman Gain Update.**

(1) Initial Error Covariance $P_{1|0} = P_0, t = 1$
(2) Compute Gain: $K_t = P_{t|t-1}H_t^T[H_t^T P_{t|t-1}H_t^T + R_t]^{-1}$
(3) Update error covariance
  (a) $P_t = (I - K_t H_t)P_{t|t-1}$
  (b) $P_{t+1|t} = A_t P A_t^T + G_t Q_t G_t^T$
(4) t = t+1
(5) Go back to (2) until stop condition

where

$$Q_t = E[w_t w_t^T] = 0$$
$$R_t = E[v_t v_t^T]$$

6.2. **State Estimation.**

(1) Initialize estimated state $\hat{x}_0$
(2) Collect new measurement y_t
(3) Update State Estimate with new measurement, with updated Kalman gain from above
  (a) $\hat{x}_{t|t-1} = A_{t-1}\hat{x}_{t-1}$
  (b) $\hat{x}_t = \hat{x}_{t|t-1} + K_t(y_t - H_t\hat{x}_{t|t-1})$
(4) t = t+1
(5) Go back to (2) until stop condition

6.3. **Pseudo-code.**

(1) Initialize $P_{pri}, x_{pri}, x_{pos}$
(2) Get Measurements y from sensors
(3) $[H, C] = hmatrix(x_{pos})$
(4) $K = P_{pri} * H.' * inv(H * P_{pri} * H.' + R)$
(5) $P_{pos} = (I - K * H) * P_{pri}; P_{pri} = P_{pos} + Q;$
(6) $b = bfunction(x_{pri}, u, Cv, Cr);$
(7) $x_{pri} = x_{pos} + b;$
(8) $[H, C] = hmatrix(x_{pri}, Lx, Ly);$
(9) $x_{pos} = x_{pri} + K * (y - H * x_{pri} - C)$
(10) Loop back to (2)

7. Communication Method

The communication method used in this lab is through the access point created by the ESP8266 WIFI chip. Since all our computations (Kalman Filter and Motion Planning Algorithm) are done in Matlab on personal computer, the computer needs to be connected to this access point for data transmission. Matlab requires measurement data from ESP by initiating a GET request to "192.168.4.1:100". ESP will then serve the current measurement data in JSON format, using an Arduino JSON library developed by MIT.

To transmit computed control input from Matlab to ESP, however, the information is itself encoded in the GET request, after / separated by commas with no spaces between. For example, a turn, move, turn signal will be sent as follow,

Matlab initiates a GET request to "192.168.4.1:100/100,-100,200,200,-300,300". The ESP will get this request string and serve some garbage, while decodes the information in the string and executes the according actions.

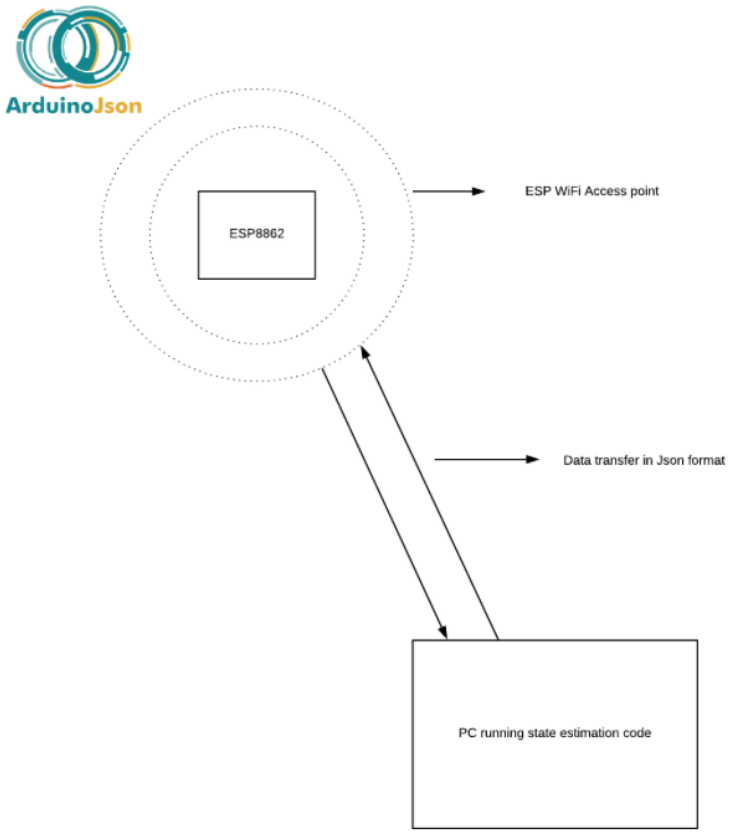

Figure 3. Communication diagram

8. Motion Planning

8.1. **Introduction.** In Lab 4, the two wheel robot car will perform parallel parking between two rectangular shape obstacles. First the ideal behavior of the car was simulated in Matlab and then implemented on the car. In the simulation, assumption was the obstacles are located at right top corner and left top corner of the box and the car would start somewhere in the left half space of the box(Figure 4).

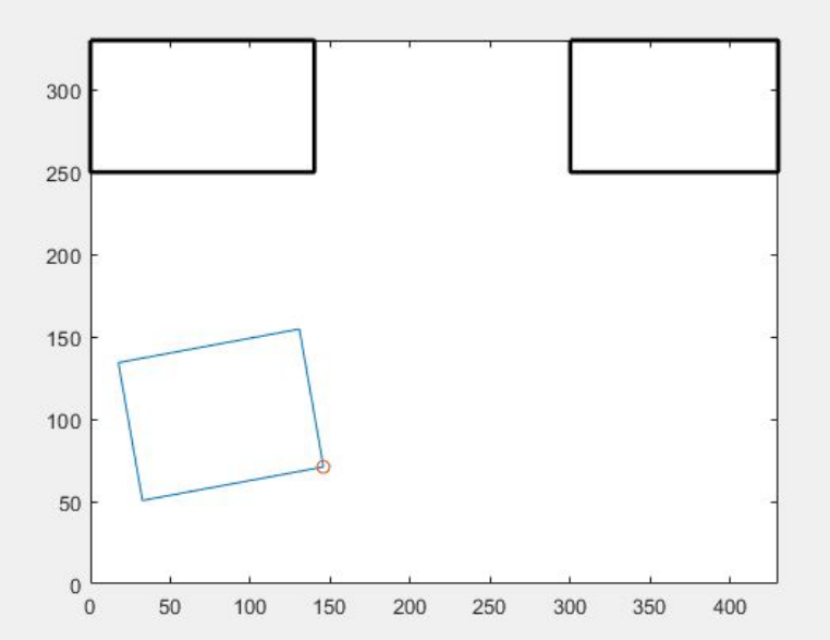

Figure 4. Position of the robot and obstacles

As shown in the figure, the origin coordinates are set to be the left bottom corner. The measurement scale is 1mm. The obstacles are marked as the black bordered rectangles and the blue one is the car. The small red circle at right front corner of the car is where the range sensors are placed. In this set up, the motion planning algorithm then calculates an optimal path for the car to successfully finish its parallel parking without hitting the obstacles or the walls.

9. Algorithm

This section goes the implemented algorithm in this lab. The motion planning procedure applied consists of two main functions :

- Get Milestones
- Get Control Inputs

9.1. **Get Milestones.** This function, as the name suggests, finds a series of milestone positions that the car must reach on its way the destination. In the very beginning of the function, all the required constants and dimensions are input which then will be used to calculate the milestones(Figure 5). These constants are previously defined in page 4 "Symbols" section.

```
global Cv; %Translational coefficient of the robot
global Cr; %Rotational coefficient of the robot
global box_length; %The dimension of the box, this is the longer side
global box_width;  %This is the shorter side
global left_obstacle_length; %The distance from left to right of the obstacle located on the left top corner
global left_obstacle_width;  %The distance from up to down of the obstacle located on the left top corner
global right_obstacle_length; %The distance from left to right of the obstacle located on the right top corner
global right_obstacle_width; %The distance from up to down of the obstacle located on the right top corner
global car_length; %The dimensions of the robot
global car_width;
global car_semidiagonal; %The distance from the rotating center to the further corner of the car
global safety_distance; %The distance we want the car to keep from the walls and obstacles
global N; %A constant that is used to control the simulation speed, the smaller N is, the faster the simulation.

% Constants in mm or radians
Cv = 120;
Cr = 70 * pi/180;
box_length=430;
box_width=330;
left_obstacle_length=140;
left_obstacle_width=80;
right_obstacle_length=130;
right_obstacle_width=80;
car_length=115;
car_width=85;
car_semidiagonal=sqrt(car_length^2+(car_width/2)^2);
safety_distance=5;
N=30;
```

Figure 5. Get Milestones function initialization

The "safety" constant initialized in the code is used to create a range of safe distance shield around obstacles and the walls so the car moving in the box would not hit any of them.

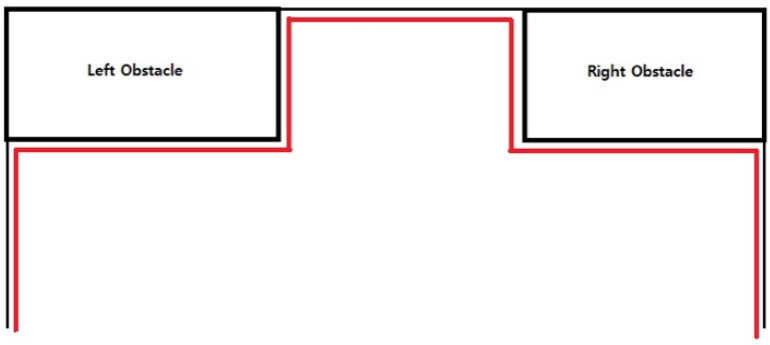

Figure 6. Indication of the safety distance around obstacles and the walls

Safety distance for the code shown in Figure-5 is set to be equal to 5, which means the car would always keep a 5 millimeter distance from the obstacles and walls. The corresponding distance is measured as the gap between the red line and black lines in Figure-6.

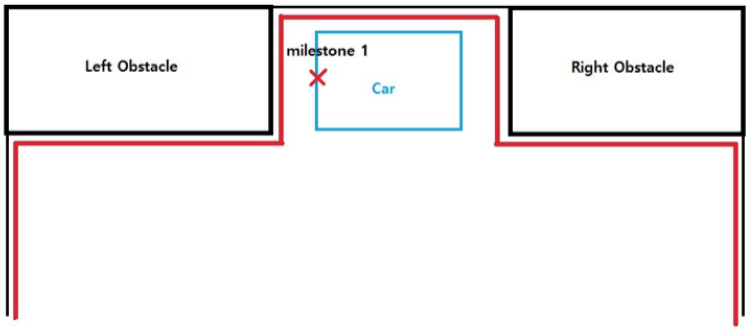

FIGURE 7. Finding the first Milestone

Then the function will find the first milestone position such that when the rotating center(the middle point of the segment that connects the centers of two wheels) of the car reach that position, the car will be the same distance away from both obstacles.(Figure 7) This milestone 1 is what we call the destination milestone. In other words, if the car successfully gets to this position, the parallel parking is completed.

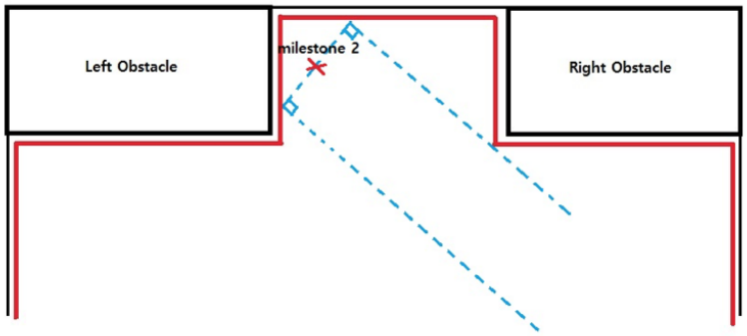

FIGURE 8. Finding the first Milestone

The next step is to find the milestone position that the car must reach before getting into the parking space, where the car starts to back up somewhere close to the right obstacle. Seemingly, the milestone 2 might look very close to the milestone 1 in this specific case when the parking space is pretty narrow. Unlike Milestone 1 that can be directly found by looking at the dimensions of the objects,milestone 2 has to be derived from solving the equation shown in next figure.

```
%////////////////////////////////////////////////////////////////////////////////////////////////////////////////
syms x;
d=box_length-right_obstacle_length-safety_distance;
e=box_width-right_obstacle_width-safety_distance;
c=destination_y;
f=left_obstacle_length+safety_distance;
R=half_car_width;

%This equation expresses the relationships between important parameters on the optimal backward parking path
eqn = ((x-f)/R)*sqrt((x-d)^2+(c-e)^2-R^2)-R*sqrt(1-((x-f)/R)^2) == c-e;

solx = vpa(solve(eqn, x));
%////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

FIGURE 9. Deriving Milestone 2 position

Figure 6 shows the equation we used to find the x coordinate of the milestone 2. The equation describes the geometric surrounding of the parking space and the relationships between important constant parameters. This equation will guarantee that the car won't hit any of the obstacles or the wall while it is moving backward into the parking space and eventually rotate to face the obstacle. As it can be seen in Figure 6, all of the parameters are defined by the global constants. So if we change the settings for any of the objects, this equation will change accordingly, and gives out different answer. This way the algorithm can be utilized on any different set of box and obstacles as long as the constants and global variables modified accordingly. The milestone 2 is the most important milestone in the algorithm. Once this milestone calculated, the rest of the milestones are easier to be calculated.

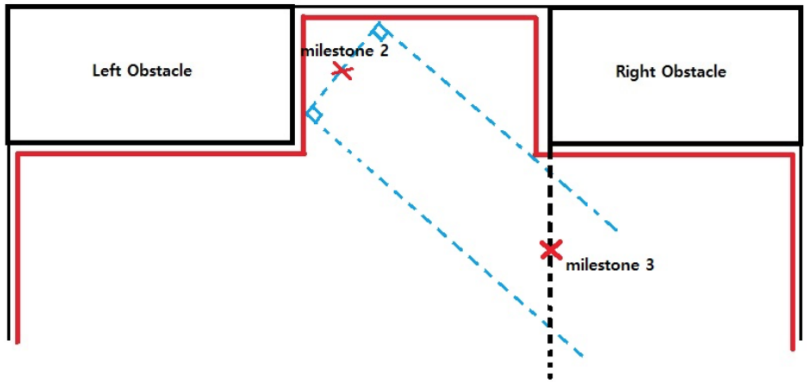

FIGURE 10. Deriving Milestone 3 position

The milestone 3 is the position that the car must reach right before starting to move into the parking space. It can be found based on the coordinates of the milestone 2. Draw the expected trajectory of the car(blue dash line) and then the black dash line by extending the left boundary of the right obstacle. The intersection of these two lines is what will be used to calculate milestone 3.

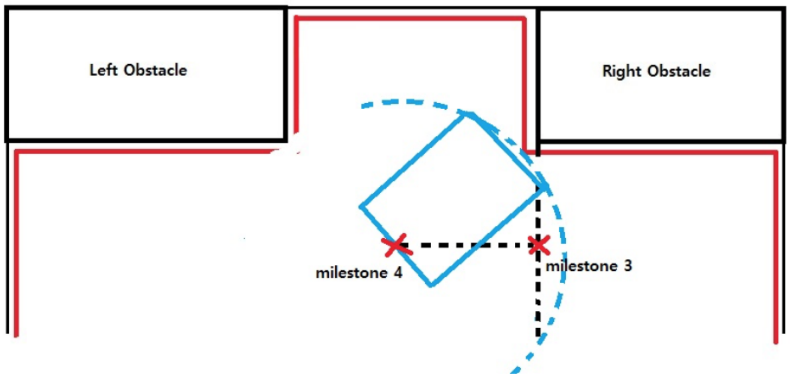

FIGURE 11. Deriving Milestone 4 position

To find the milestone 4, the function will first make a horizontal line that starts from the milestone 3 to the left. Then it will try to make a circle that has the center on the drawn line with the radius of the "semi-diagonal" of the car and check if the circle has any intersections with the red line.(red line being the defined safety distance) It will define the center of the circle as the milestone 4 when the circle is tangent with the red line corner of the right obstacle. The "Get Milestones" will then, finally, output these four milestone coordinates so that the "Get Control Inputs" function can use them as its guidance to generate the proper control inputs for the car.

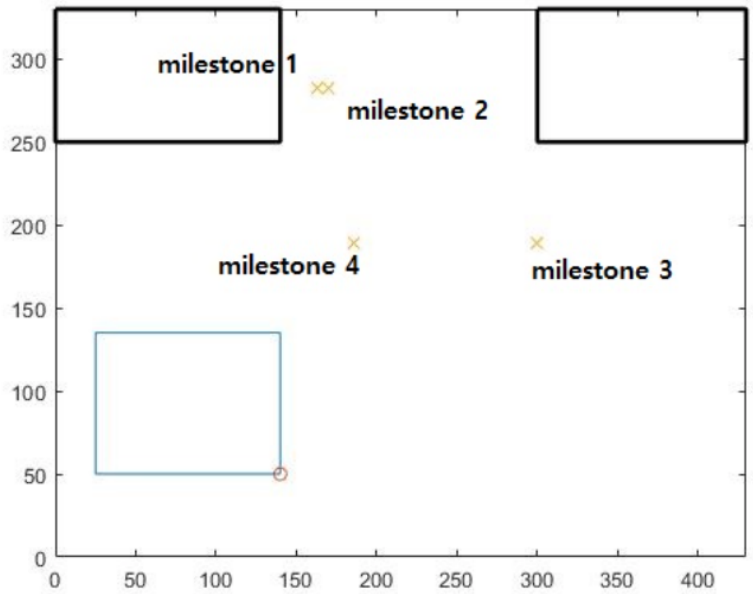

FIGURE 12. All Milestones overview

The actual result from calling the "Get Milestones" is the shown in figure 12.

9.2. **Get Control Inputs.** Running "Get Milestones" gives the positions the car needs to visit in order to perform the parallel parking. Now calling "Get Control Inputs" will generate 3 required control inputs in order to move the car in the right path considering milestones. These 3 inputs are sorted as : "turn, move, turn". Meaning, the first input tells the car to turn a certain degree and the second input tells the car to move by a certain distance, and the last input tells the car to turn by another certain degree. Now, suppose we initially put the car at the position shown in Figure 12, calling the "Get Control Inputs" function we will generate the following inputs.(Figure10)

```
pre_parking_s0_input_1 =

    0.3616


pre_parking_s0_input_2 =

    1.2986


pre_parking_s0_input_3 =

   -0.3616
```

FIGURE 13. "Get Control Inputs" function call

If we check Figure 5, there are two constants $Cv = 120$ and $Cr = 70$. (Defined in "Symbols" section 1)
These are the translational and the rotational constant of the car. It means, given $input = 1$ the car will move by 120 millimeter or rotate by about 70 degrees. Considering these two constants and if we look at Figure 13, we can see that the 3 inputs will do "turning the car counter clock wise by 70×0.3616 °, move the car forward by $120 \times 1.2986mm$ and turn the car clock wise by 70×0.3616 °.

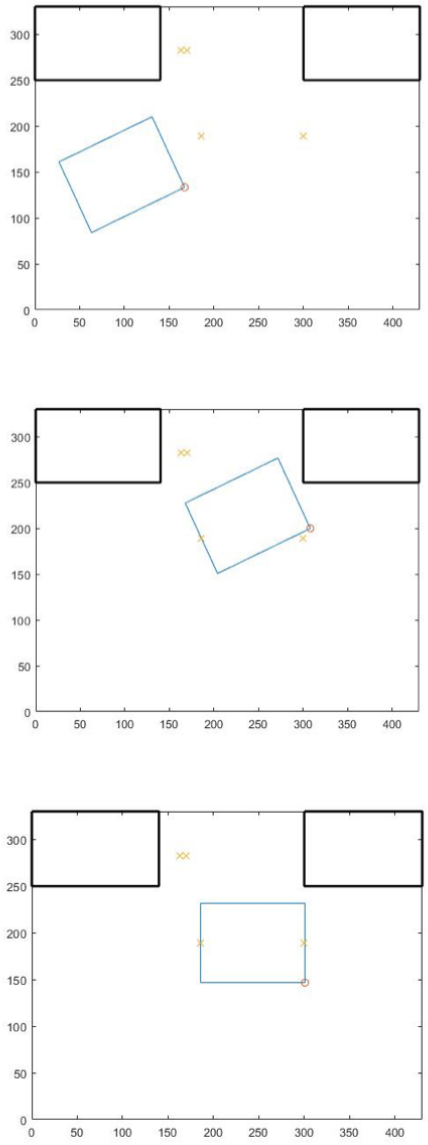

FIGURE 14. sequential motion commands

The first picture in Figure 14 shows the result after plugging the first turning input, the second picture shows the result after plugging the moving input, the last picture is the result after plugging the second turning input. Similarly, when the function has been noticed that the car has reached the target milestone, it will generate the next series of inputs to lead the car to the next milestone position and it continues the same procedure until the car gets to the destination milestone to complete its parallel parking (Figure 15).

For more about the dynamic Matlab simulation of our algorithm please check the "Dynamic Simulation" folder on our GitHub page. (Link given in References)
Below is the result of running the complete simulation in Matlab.

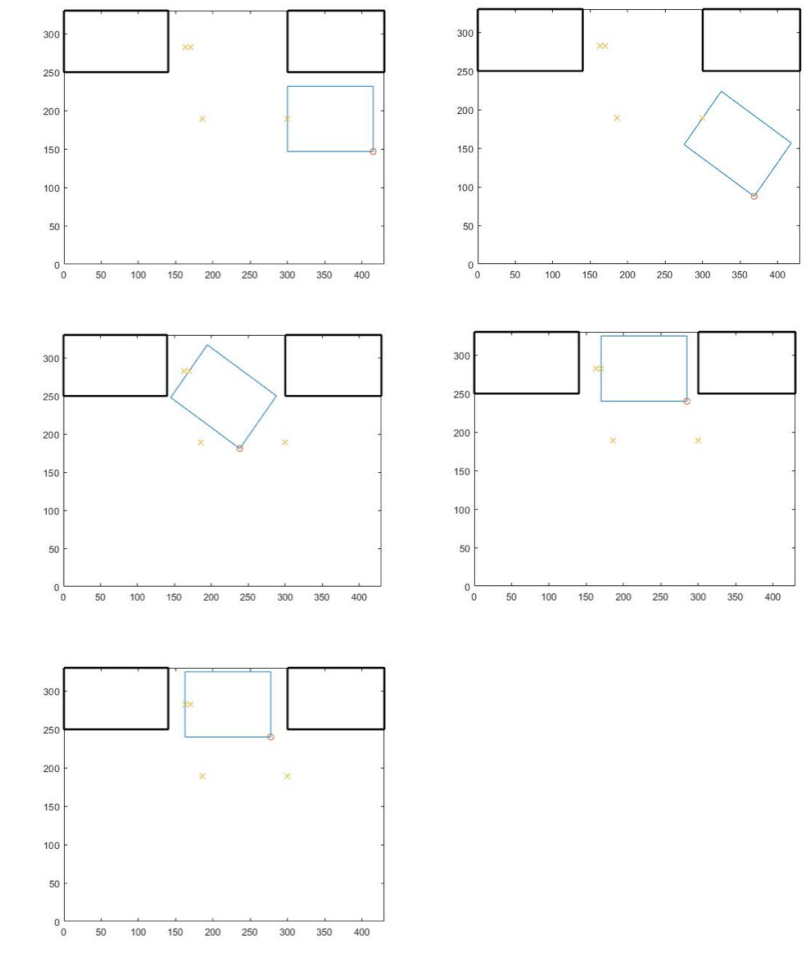

FIGURE 15. Complete Parallel Parking Process

10. Additional variable

For testing the algorithm on the actual module we got help from the state estimation Kalman filter derived above. In addition to Kalman filter, we added two more variables to improve the performance of the car, which are the "acceptable radius" and "Close Enough".(Figure 16)

```
%These are the acceptable radius,
%the car will keep moving until it gets to a point within this radius from the target point,
%then move to the next milestone.
global acceptable_radius1;
global acceptable_radius2;
global acceptable_radius3;
global acceptable_radius4;
global Close_Enough;%Determine if it's close enough to the target position.
```

Figure 16. Motion Planning Helper variables

For example, assuming set $acceptable_{radius1} = 10$ if then the car gets to a position that is within 10mm from the first milestone, it will consider it as "close enough" to the target.
The "Close Enough" boolean variable will be set to 1 if the distance is within defined range,it will otherwise be set to 0. This variable is used to determine if the car is close enough to the current target milestone so that it can start moving to the next milestone in the next step.

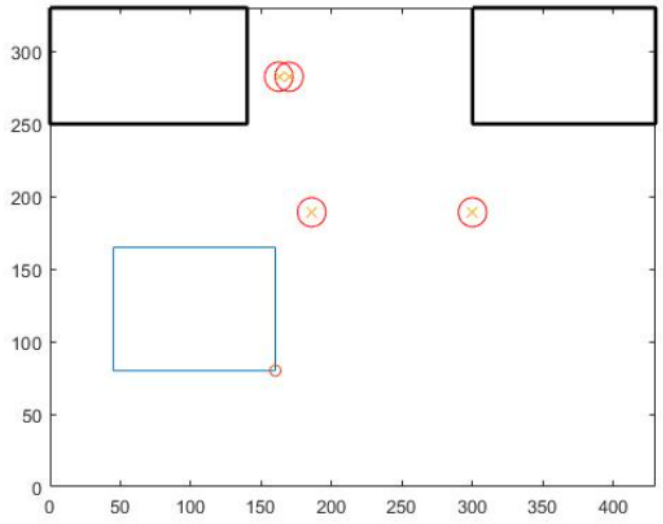

Figure 17. acceptable radius variables for different milestones

In the main control code, we put a "while" statement (Figure 18) for each of the milestone points. Inside the loop, the "Get Control Inputs" function will keep generating the inputs to order the car to get closer and closer to the target spot until the car is distanced within, specified range, from the milestone. Therefore the variable "Close Enough" is set to 1.

```
%Get the milestones according to the current dimension settings
Milestones=Get_Milestones;
N=length(Milestones)/2+1;
for i=1:1:N
Close_Enough=0;

%Estimated_State=[];%Replace this with the real initial reading.

while Close_Enough==0
[u_turn,u_move,u_extra_turn]=Get_Control_Inputs(Estimated_State,i,Milestones);

%Send u_turn,u_move,u_extra_turn to the car.
%Wait util the motion is done.

figure_index=figure_index+1;
if Close_Enough==0

%Estimated_State=[];%Replace this with the real estimation reading.

end
end
end
```

FIGURE 18. while loop statement for control system

As it can be seen in Figure 18, the new inputs are recalculated in the iteration loops based on the current estimated states. If the estimated states are reliable and the process errors are decently small, we can expect this algorithm to give a pretty good result on the real world parallel parking. And it indeed did a good job in our actual tests. In our real world parking experiments, we have recorded the expected positions and the estimated positions around all of the milestone points and plotted them as figures when the parking is finished. Figure 19 shows the result of one the run tests we did. For more information about the real time tests, please refer to section 11 ("Demonstrations").

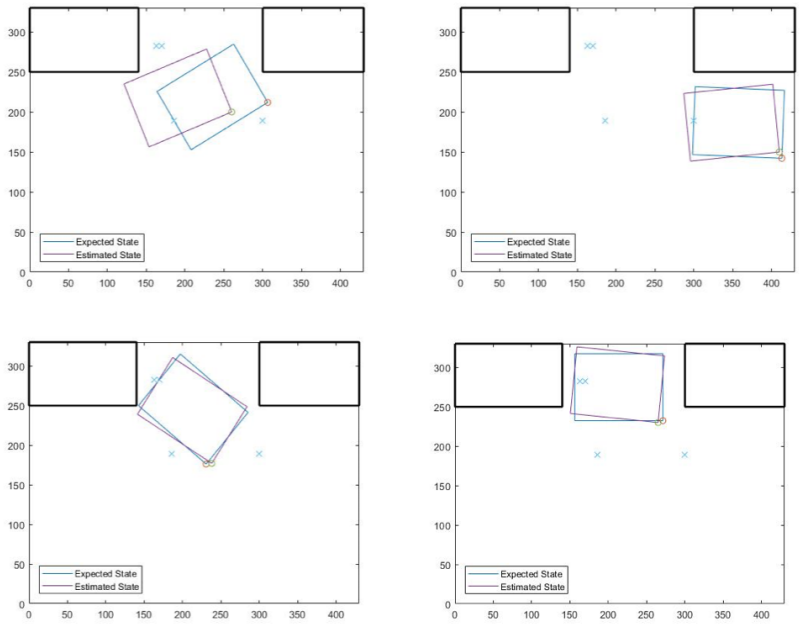

FIGURE 19. Real time simulation, Results recorded in Matlab

11. RESULTS/DEMONSTRATION

All codes and related documents can be found on GitHub
For demonstration videos please visit links below:

- Full demo (motion planning and state estimation combined) `https://youtu.be/30bCeLLuHEQ`
- State Estimation Demo `https://youtu.be/0avIQL3_okI`
- Motion Planning Demo test case 1 `https://youtu.be/0AI09jDCzT4`
- Motion Planning Demo test case 2 `https://youtu.be/vqe-3CE-YhE`
- Motion Planning Demo test case 3 `https://youtu.be/mMm7KnLYAuQ`
- Motion Planning Demo test case 4 `https://youtu.be/OZEuJw4zu0I`