FINAL PROJECT REPORT
MAE 5810

# Pinhole Camera Tracking and Path Planning Based on Two-wheeled Vehicle Differential Steering Model

Yang Jiao
NetID: yj286
10 Dec 2021

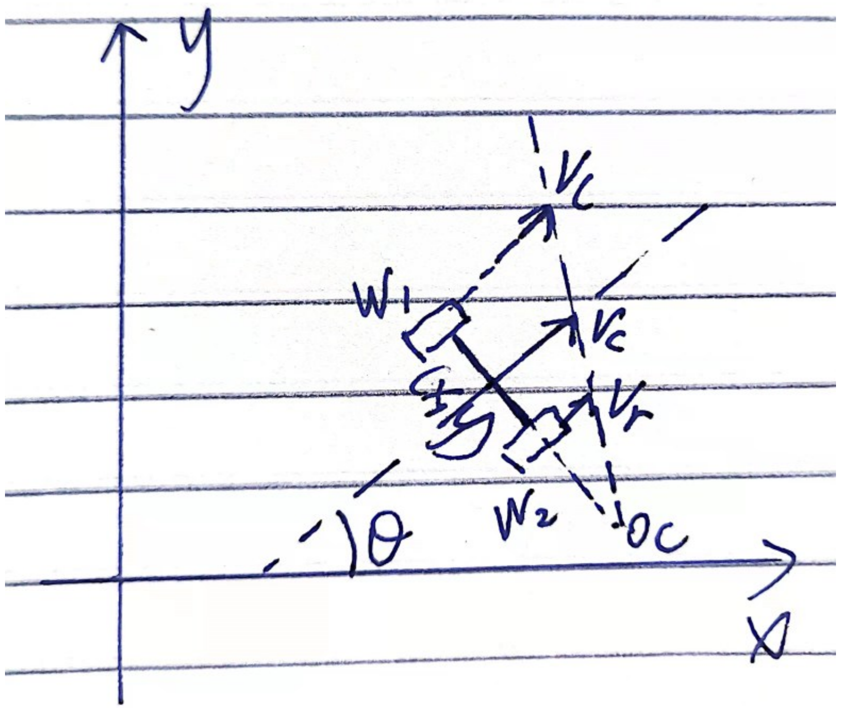
- The results are only briefly shown here in the demo powerpoint
- For detailed derivation process and descriptions can be found in the pdf version of the report as:

Final_Project_Report-yj286.pdf

# ABSTRACT

- This article introduced the establishment of differential steering model for two-wheeled vehicles, and realized the trajectory tracking and path planning for two-wheeled vehicles by combining the pinhole camera model constructed in the mid-term project. At the same time, the A*-based heuristic search algorithm was explained considering the possible environmental obstacles in the actual application scenarios, and the simulation results were illustrated.

# *1. Differential steering model for two-wheeled vehicles*




$$v_l = \frac{\phi_l \cos(\theta)}{2}$$

$$v_r = \frac{\phi_r \cos(\theta)}{2}$$

$$l = \frac{v_r}{w_r} - \frac{v_l}{w_l}$$

$$w_c = \frac{v_r - v_l}{l}$$

$$R = \frac{v_c}{w_c} = \frac{l(v_r + v_l)}{2(v_r - v_l)}$$

$$x' = v_c \cos\theta = \frac{v_r + v_l}{2} \cos\theta$$

$$y' = v_c \sin\theta = \frac{v_r + v_l}{2} \sin\theta$$

$$\theta' = w_c = \frac{v_r - v_l}{l}$$

$$x(t+1) = x(t) + v_c \cos\theta \times T$$
$$y(t+1) = y(t) + v_c \sin\theta \times T$$
$$\theta(t+1) = \text{theta } (t) + w_c \times T$$

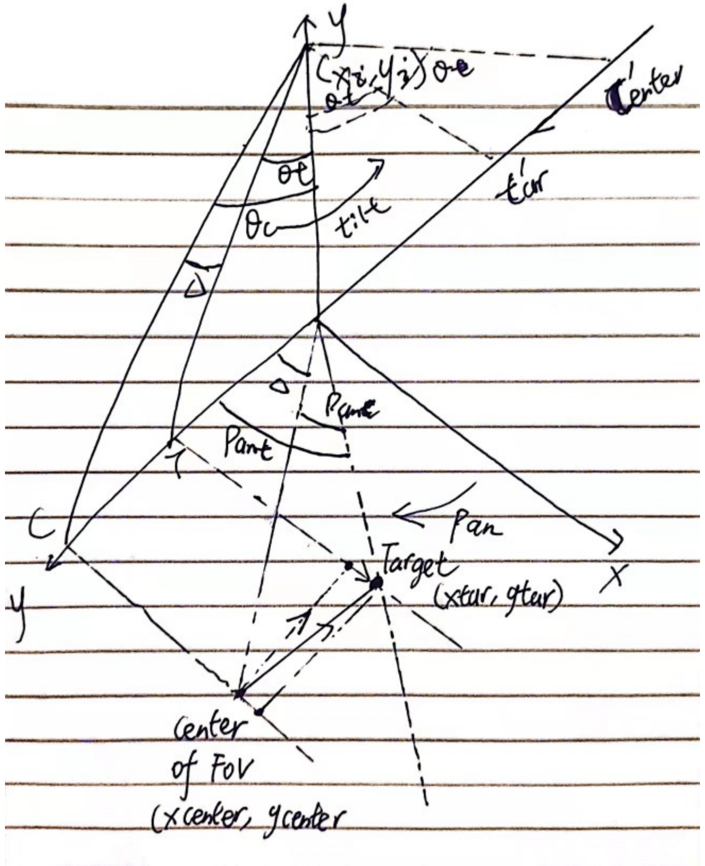
$$\boxed{\begin{aligned} v_r &= v_c + \frac{w_c l}{2} \\ v_l &= v_c - \frac{w_c l}{2} \end{aligned}}$$

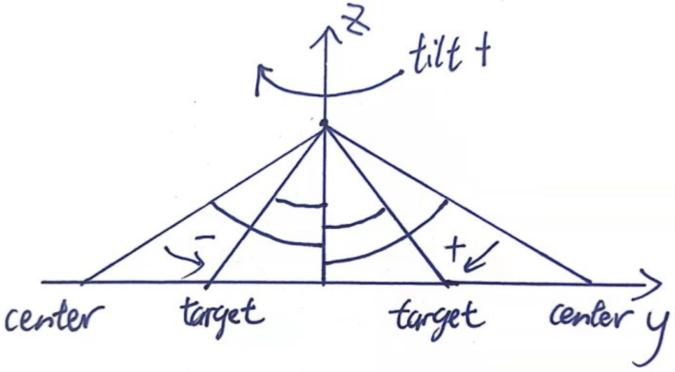
# 2. Pinhole Camera Model

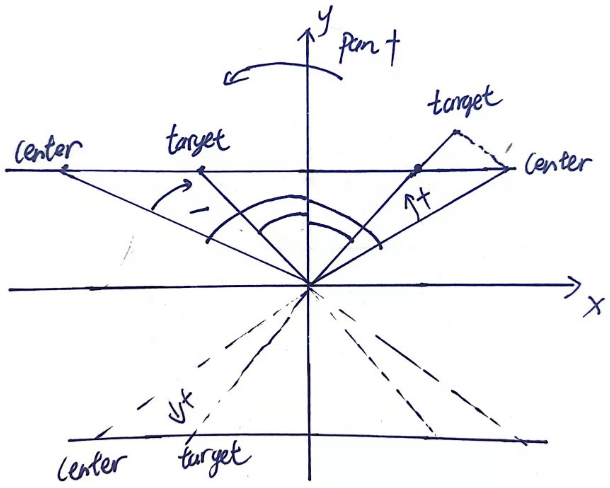
- $(\mathbf{q}_1)_V = \begin{bmatrix} \frac{a}{2} \\ \frac{b}{2} \end{bmatrix}, (\mathbf{q}_2)_V = \begin{bmatrix} \frac{a}{2} \\ -\frac{b}{2} \end{bmatrix}, (\mathbf{q}_3)_V = \begin{bmatrix} -\frac{a}{2} \\ -\frac{b}{2} \end{bmatrix}, (\mathbf{q}_4)_V = \begin{bmatrix} -\frac{a}{2} \\ \frac{b}{2} \end{bmatrix}, (1)$
- $(\mathbf{q}_\ell)_B = [(\mathbf{q}_\ell)_V^T \quad \lambda]^T, \ell = 1, \dots, 4 (2)$
- $\mathbf{H}_\psi \triangleq \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{H}_\phi \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} (3)$
- $$\begin{aligned} (\mathbf{q}_\ell)_I &= \left(\mathbf{H}_\phi^T \mathbf{H}_\psi^T\right)^{-1} (\mathbf{q}_\ell)_B \\ &= \left[\left(\mathbf{H}_\psi \mathbf{H}_\phi\right)^{-1}\right]^T (\mathbf{q}_\ell)_B \qquad (4) \\ &= \mathbf{H}_\psi \mathbf{H}_\phi (\mathbf{q}_\ell)_B, \ell = 1, \dots, 4 \end{aligned}$$
- $(\mathbf{q}'_\ell)_I = \rho_\ell (\mathbf{q}_\ell)_I, \ell = 1, \dots, 4 (5)$
- $\rho_{1,4} = \frac{-\mathbf{x}_{(3)}}{\left(\frac{b}{2}\sin\phi + \lambda\cos\phi\right)} \rho_{2,3} = \frac{-\mathbf{x}_{(3)}}{\left(-\frac{b}{2}\sin\phi + \lambda\cos\phi\right)} (6)$

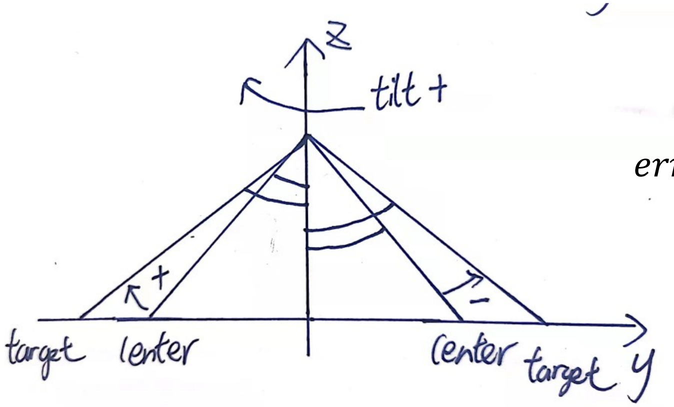
$$\begin{aligned} \xi_1 &= \boldsymbol{X} + \mathbf{q}'_1 \\ \xi_2 &= \boldsymbol{X} + \mathbf{q}'_2 \\ \xi_3 &= \boldsymbol{X} + \mathbf{q}'_3 \\ \xi_4 &= \boldsymbol{X} + \mathbf{q}'_4 \end{aligned}$$

# 3. Distance /Heading angle Control and Tracking




$$errorpan = atan\left(abs\left(\frac{xcenter}{ycenter}\right)\right) - atan\left(abs\left(\frac{xtar}{ytar}\right)\right);$$

$$errortilt = atan\left(\frac{lenctilt}{z_i}\right) - atan\left(\frac{lenttilt}{z_i}\right);$$

- $theta_{goal} = \text{atan}(\frac{y_{goal} - ytrac}{x_{goal} - xtrac}));$

- $theta_{error} = theta - theta_{goal};$

- $error_{sum} = error_{sum} + theta_{error}$

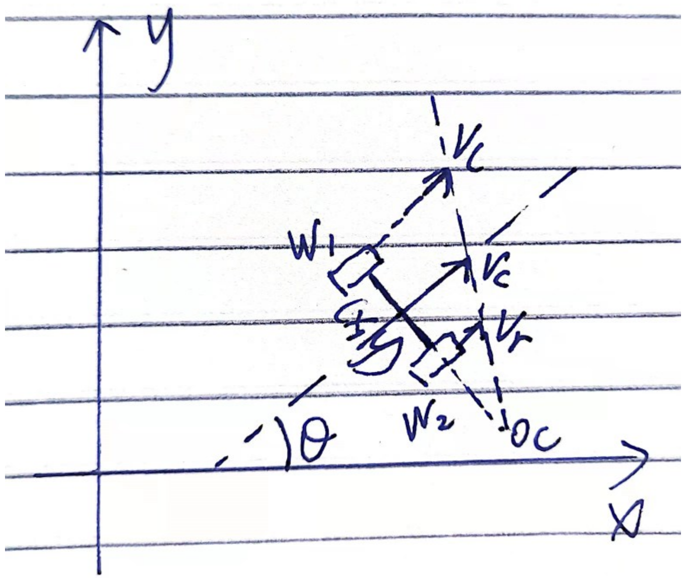
## Control:

- $w = -k_p * (theta_{error}) + k_i * error_{sum} + k_d * (theta_{error} - pre_{error});$
- $dist = \left(\left(xtrac - x_{goal}\right)^2 + \left(ytrac - y_{goal}\right)^2\right)^{\frac{1}{2}};$
- $v = k2 * dist;$

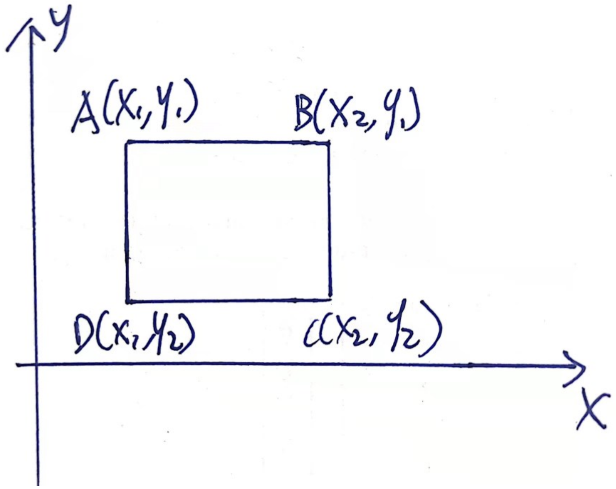
$$v_r = v + w * \frac{L}{2} \quad v_l = v - w * \frac{L}{2}$$

## Dynamic model:

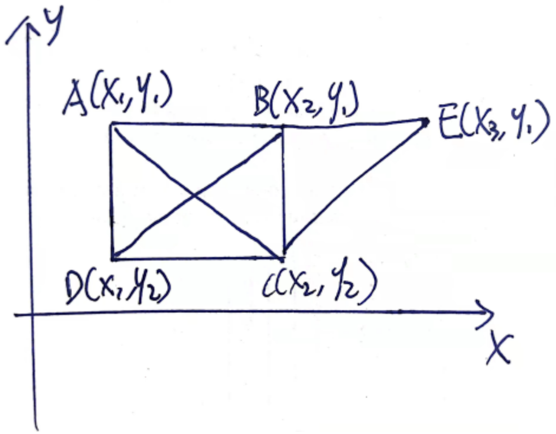
- $v = (vl + vr)/2$
- $w = (vr - vl)/L$
- $xtrac = xtrac + v * cos(theta) * T$
- $ytrac = ytrac + v * sin(theta) * T$
- $theta = theta + w * T$



# 4. A*-Based Heuristic Search Algorithm for path planning




$$F(n) = G(n) + H(n)$$




$$H(n) = \bar{AB} + \bar{BC} = |x_1 - x_2| + |y_1 - y_2|$$

$$G(n) = \bar{AB} = |x_1 - x_2|;$$

$$H(n) = \bar{BE} = \sqrt{(x_2 - x_3)^2 + (y_1 - y_2)^2}$$

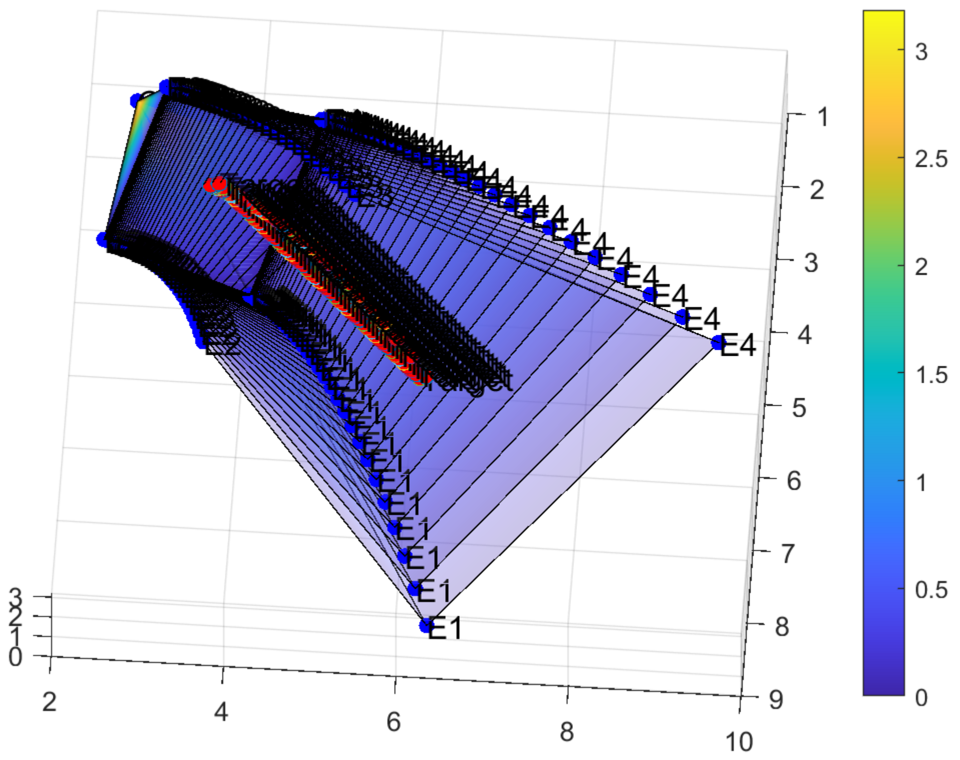
$$F(n) = G(n) + H(n) = \overline{AB} + \overline{BE}$$

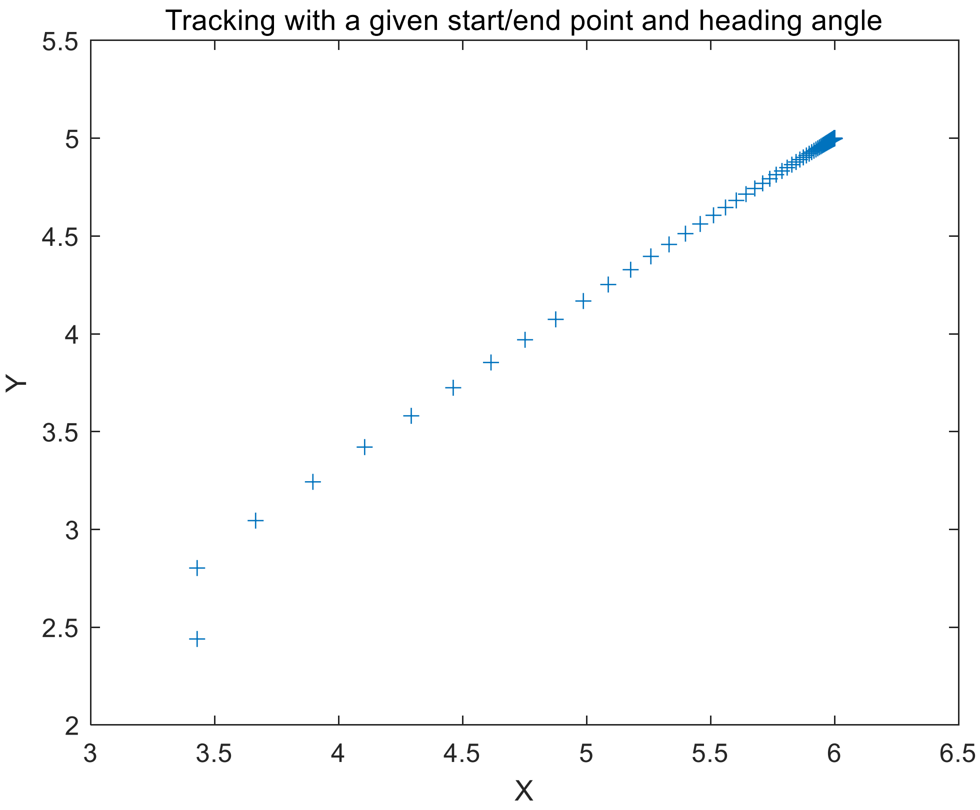
$$= |x_1 - x_2| + \sqrt{(x_2 - x_3)^2 + (y_1 - y_2)^2}$$

The search process of the A-star algorithm is implemented in the code as follows.

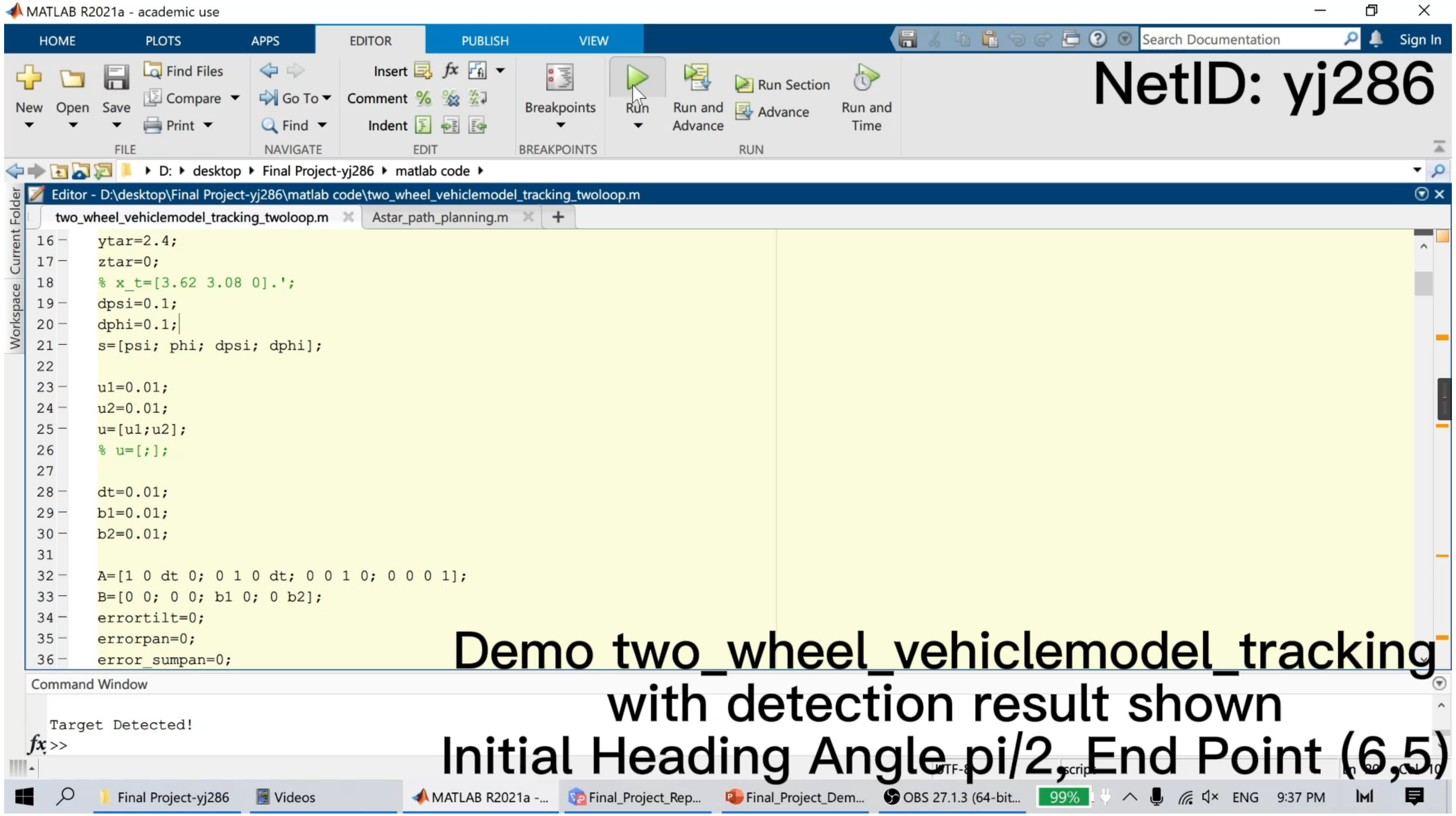
- 1’ Calculate the costs of all the blocks that are immediately next to the present one.
- 2’ Add them to valid list
- 3’ Find block with minimum total cost from valid list, make it parent for next iteration and make it unavailable for next minimum total cost comparison
- 4’ Put that in in_valid list so it won’t be visited again
- 5’ Keep doing this until the block with minimum cost is not target block.

# *Simulation Results of two_wheel_vehiclemodel_tracking*

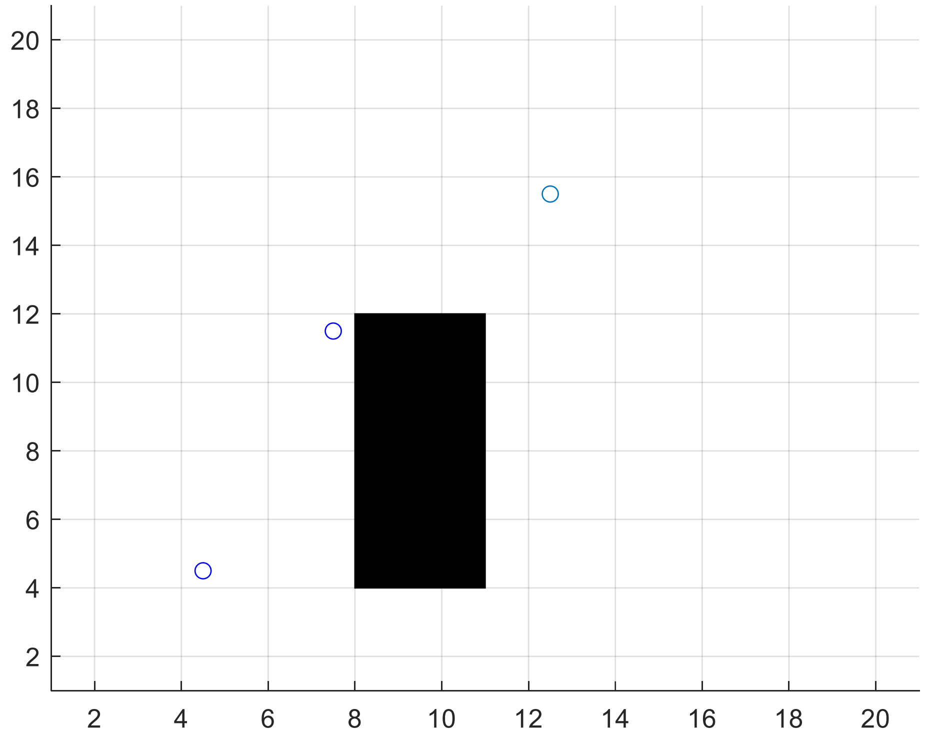


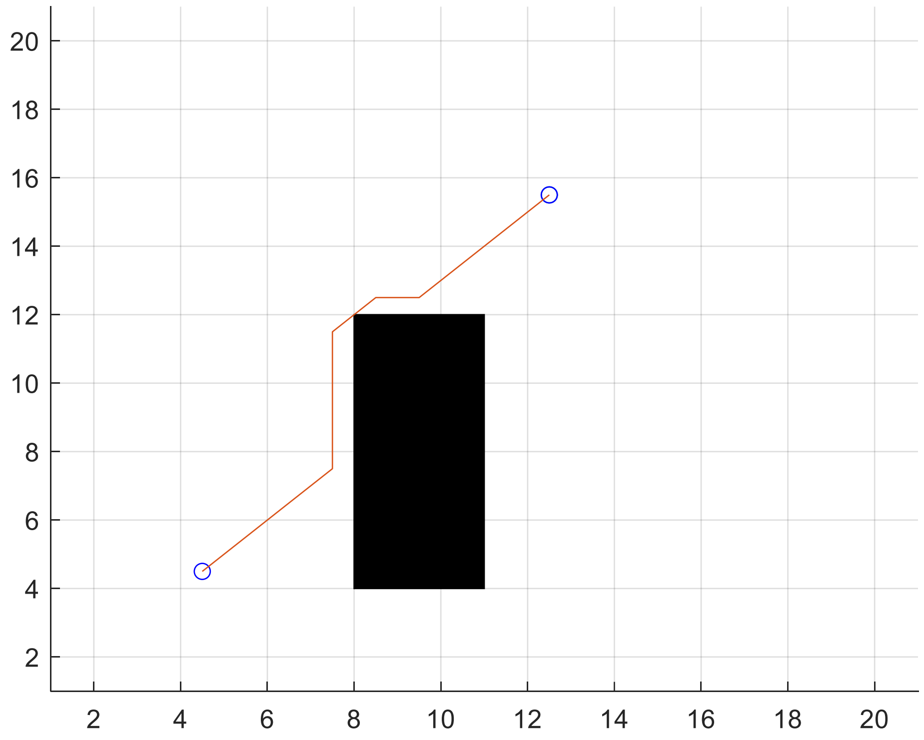



*Video also included in the folder if is not playable here*



# *Simulation Results of Astar_path_planning*



# *Video also included in the folder if is not playable here*