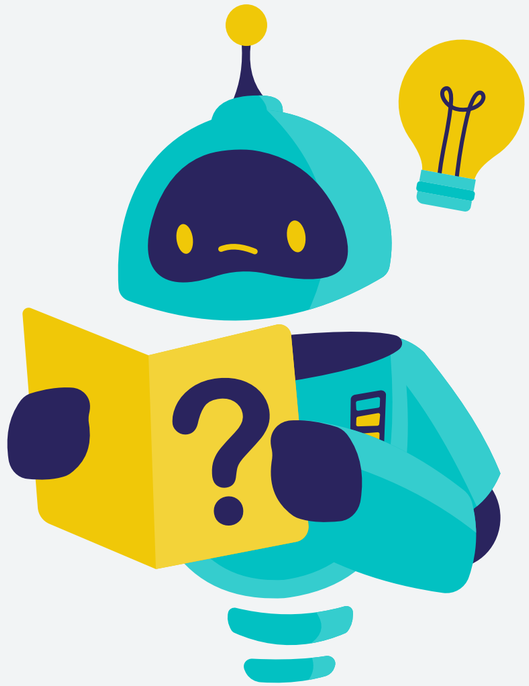
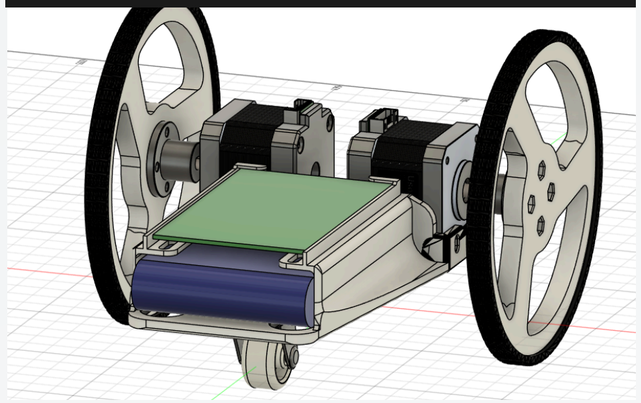
TECHNICAL
PROJECT

CONTENT



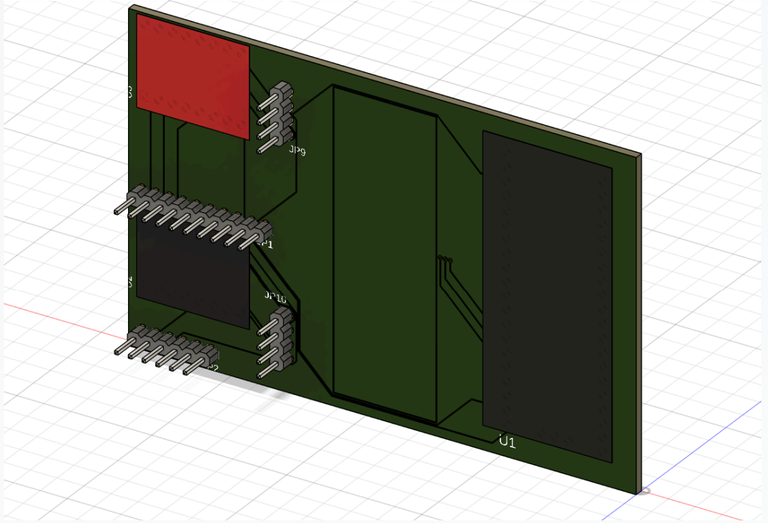
HARDWARE

CAD MODEL



- Designed with Autodesk Fusion 360
- 3D printed and assembled by us

PCB



Components on the PCB

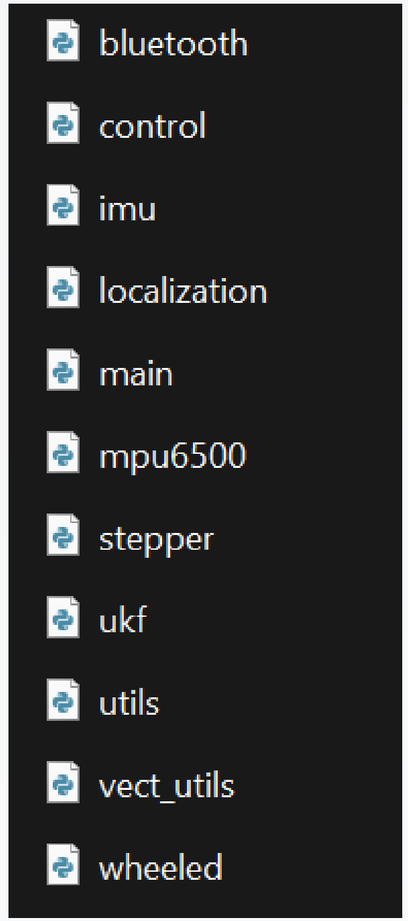
- Rasberry pi Pico
- HC 05 Bluetooth Module
- GY-6500 IMU Sensor
- Stepper Motors
- DRV8825 Stepper drivers

ROBOT SOFTWARE

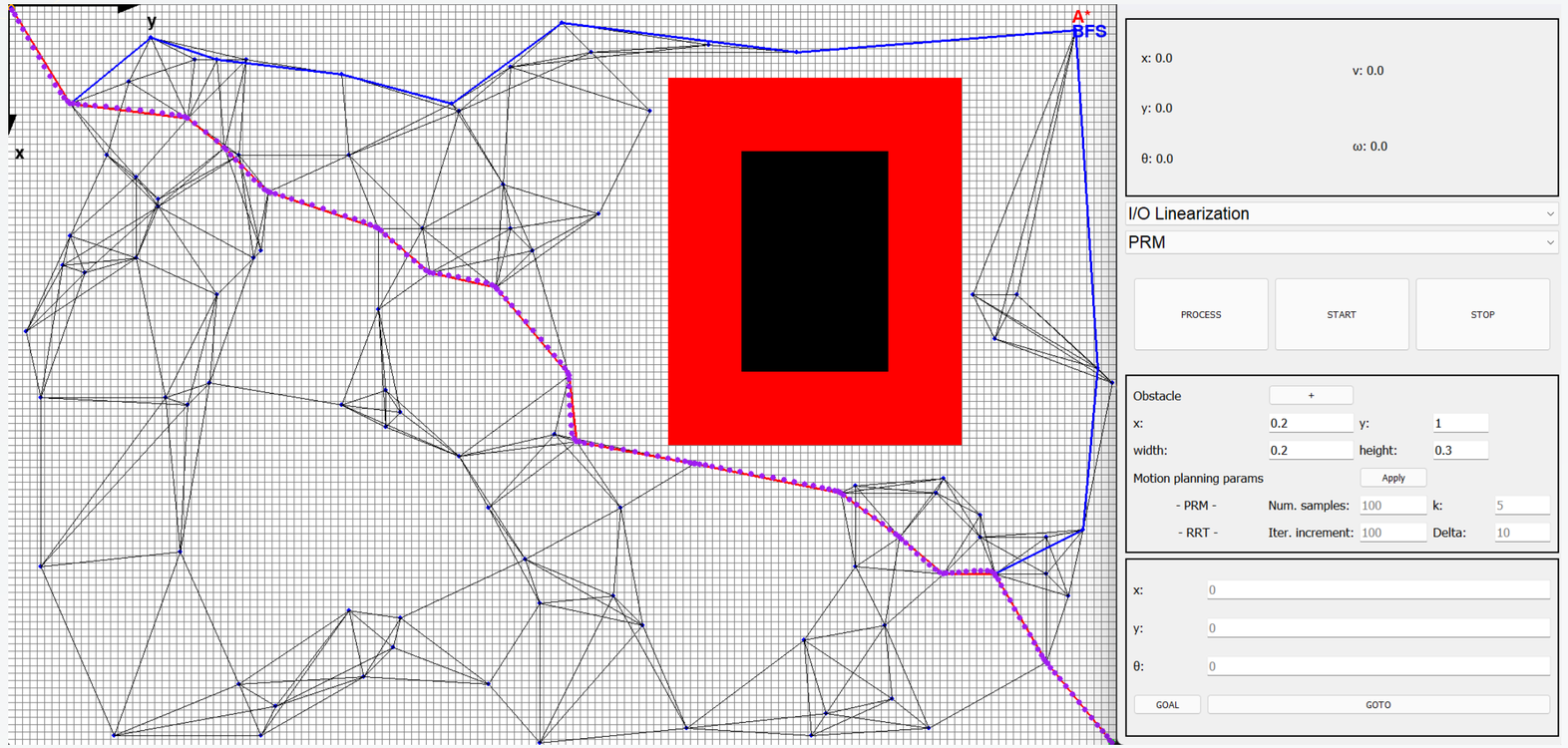
- MicroPython
- OOP (object oriented programming)
- Interruptions no busy wait

- Control: I/O Linearization, Posture Regulation
- IMU: Calibration and reading
- Localization: Euler approximation, Second-order Runge-Kutta
- Wheeled: IK solver

LOW LEVEL CONTROL



GCS SOFTWARE



- Python
- OOP (object oriented programming)
- Show the gridmap
- Handle the communication with DRIFTY (controller, goto, start, stop...)
- Contain Motion planning and trajectory planning modules

ESTIMATORS

EULERO

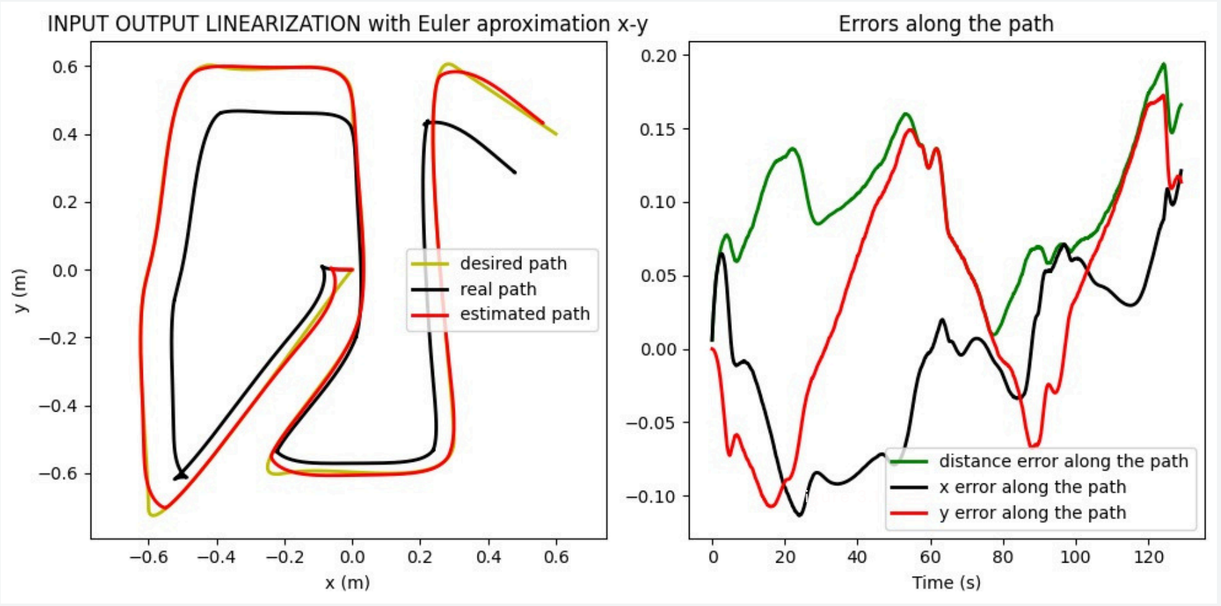
RUNGE KUTTA
SECOND ORDER

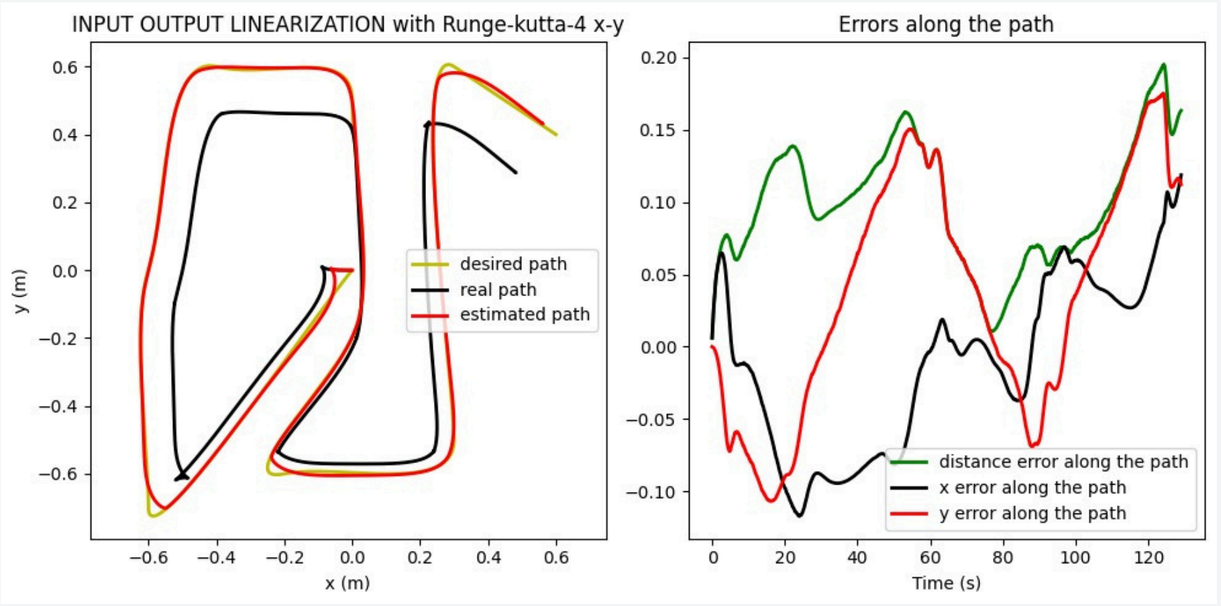
RUNGE KUTTA
FOURTH ORDEN

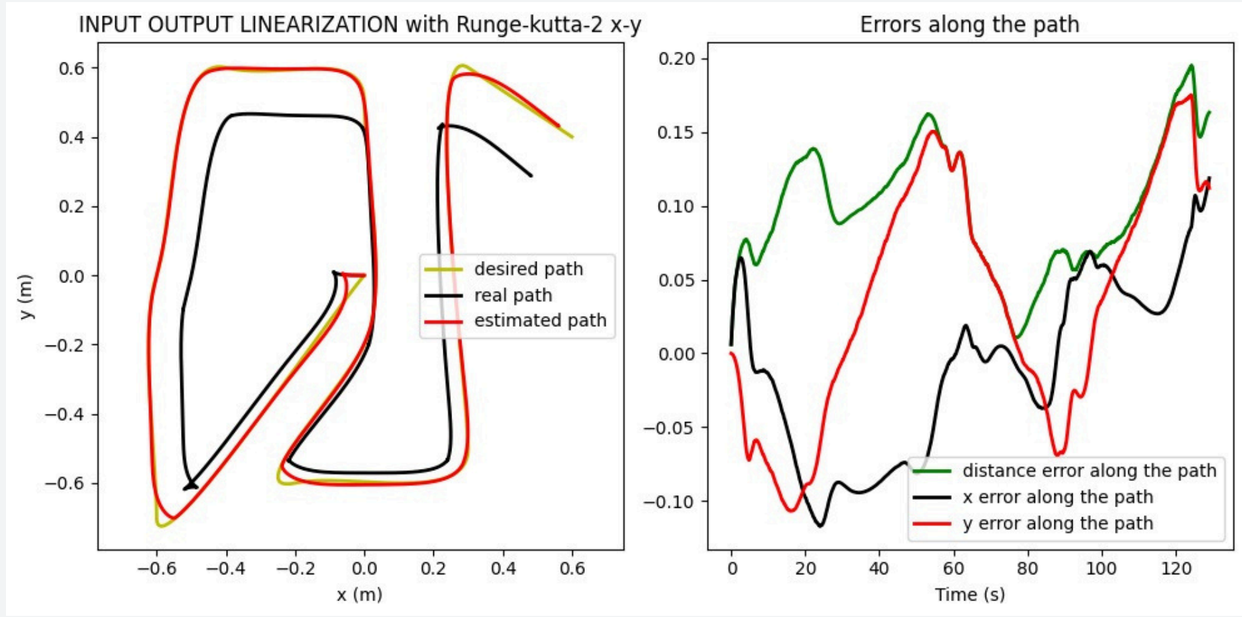
KALMAN FILTER

UNSCENDED
KALMAN FILTER

SIMULATIONS



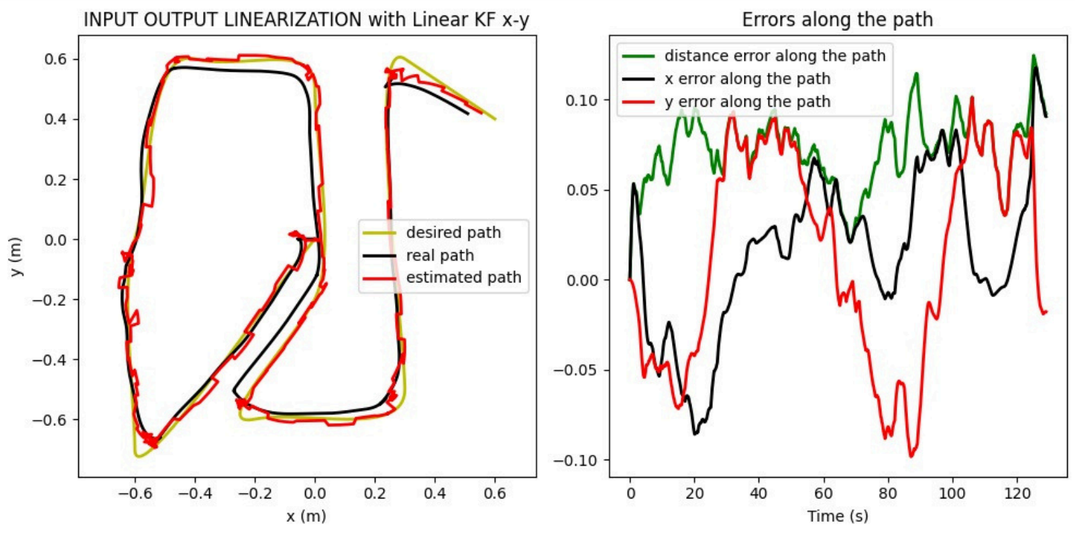


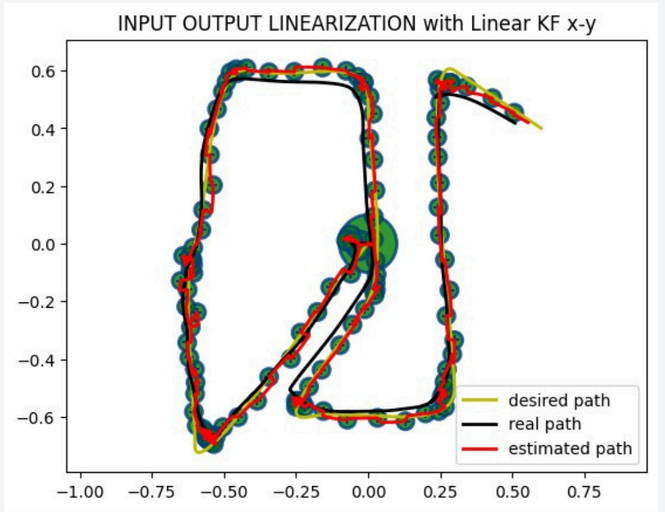


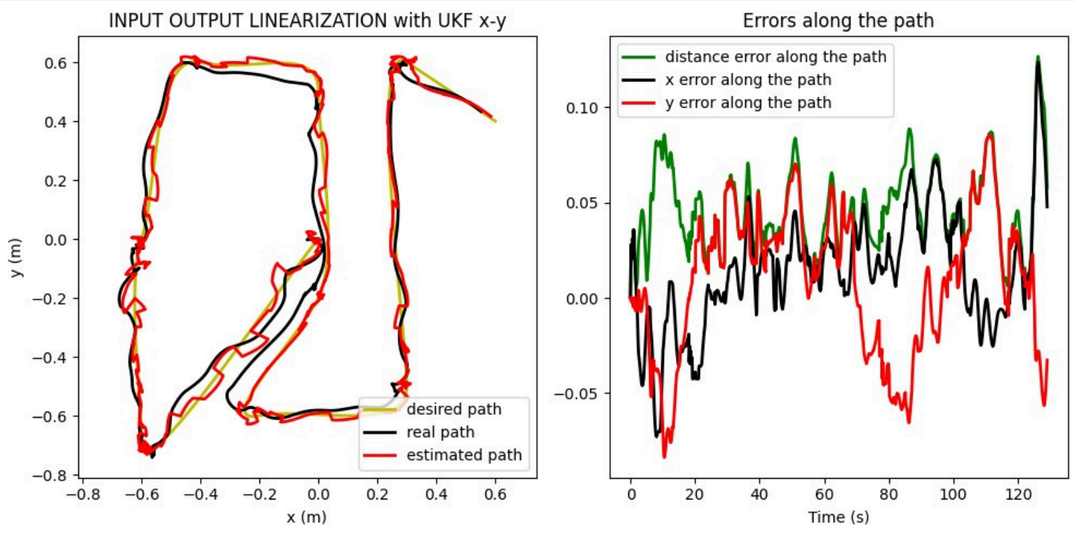
Maximum difference between :

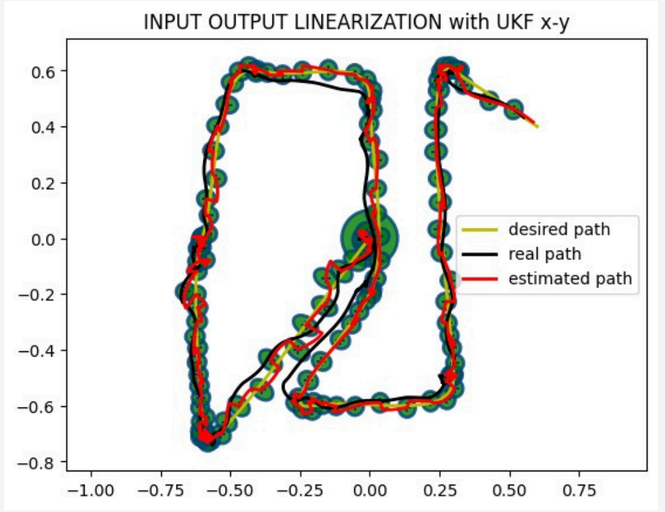
- RK-2 and Euler 0.00353
- RK-4 and Euler: 0.00353
- RK-4 and RK-2: 1.60478 e-05

SIMULATIONS

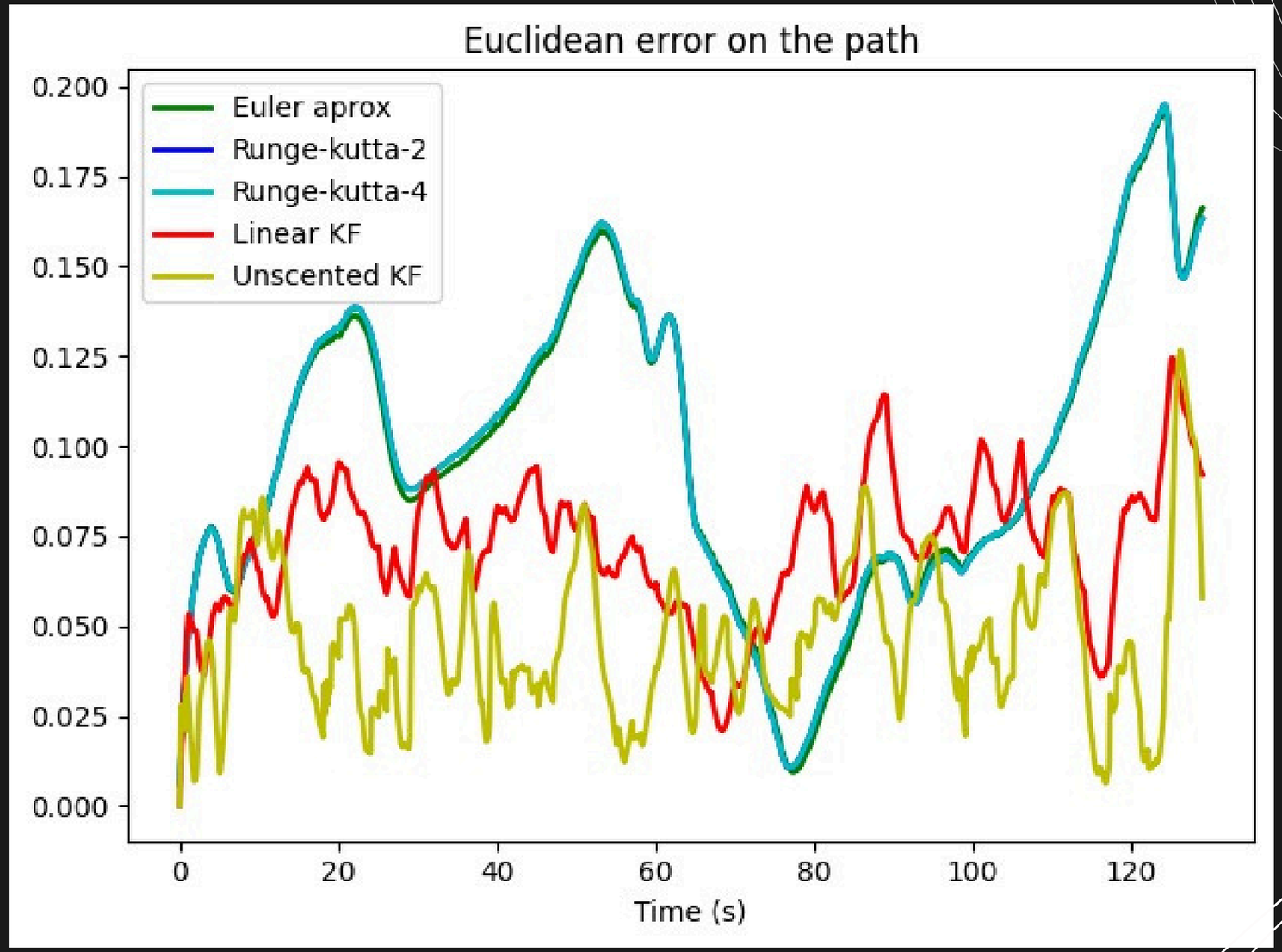




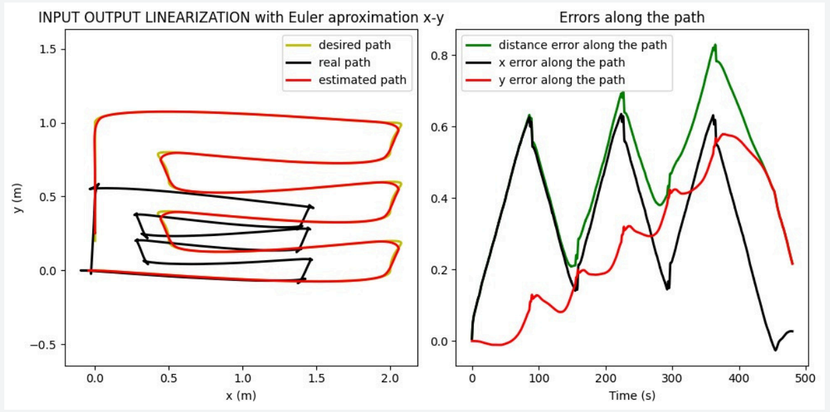


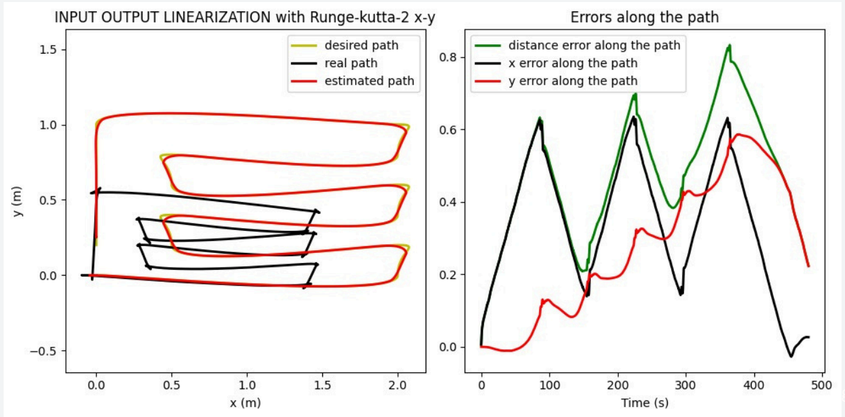


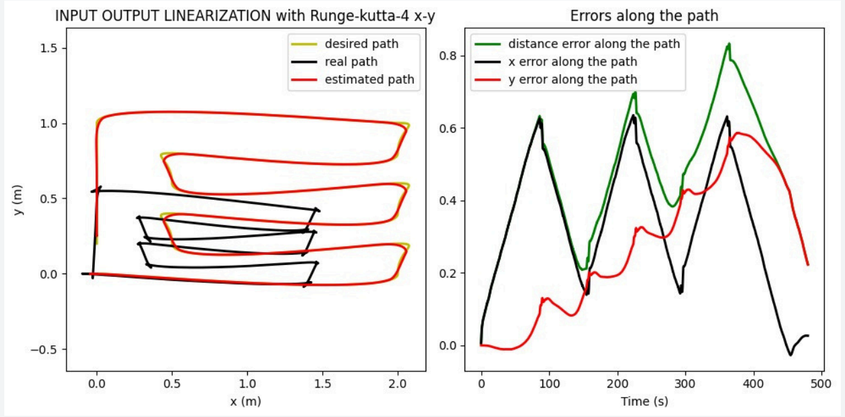
SIMULATIONS

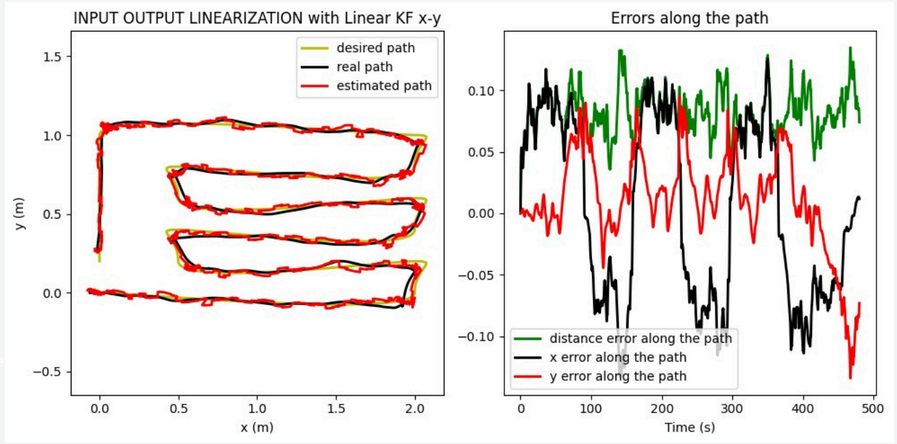


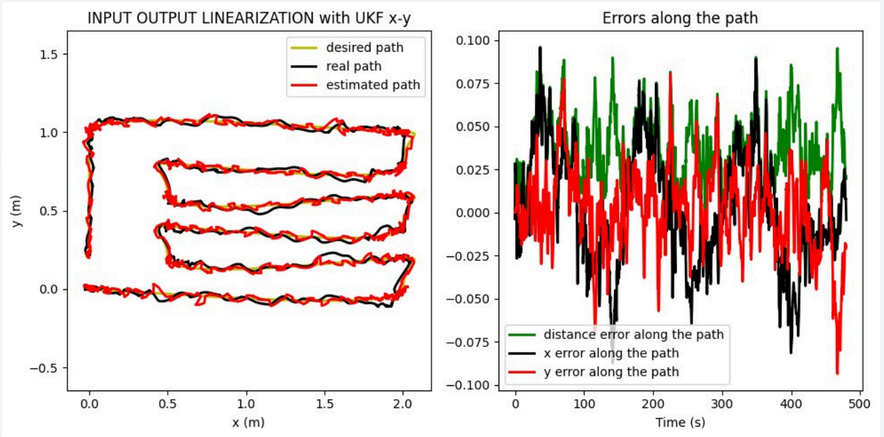
SIMULATIONS

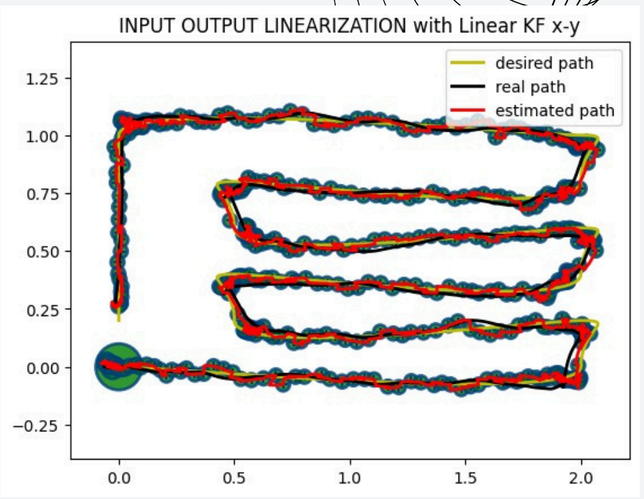


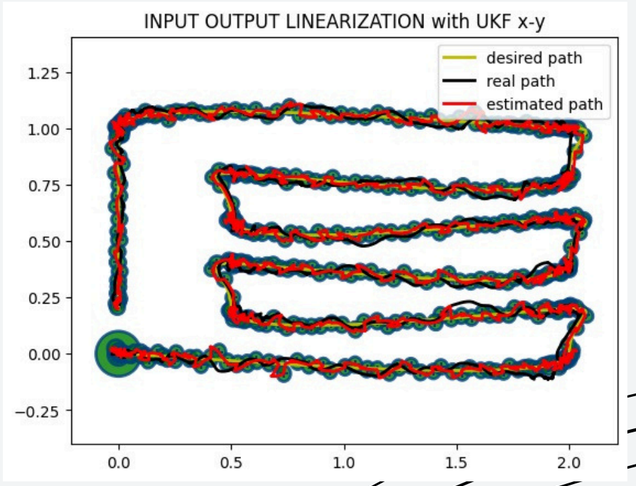




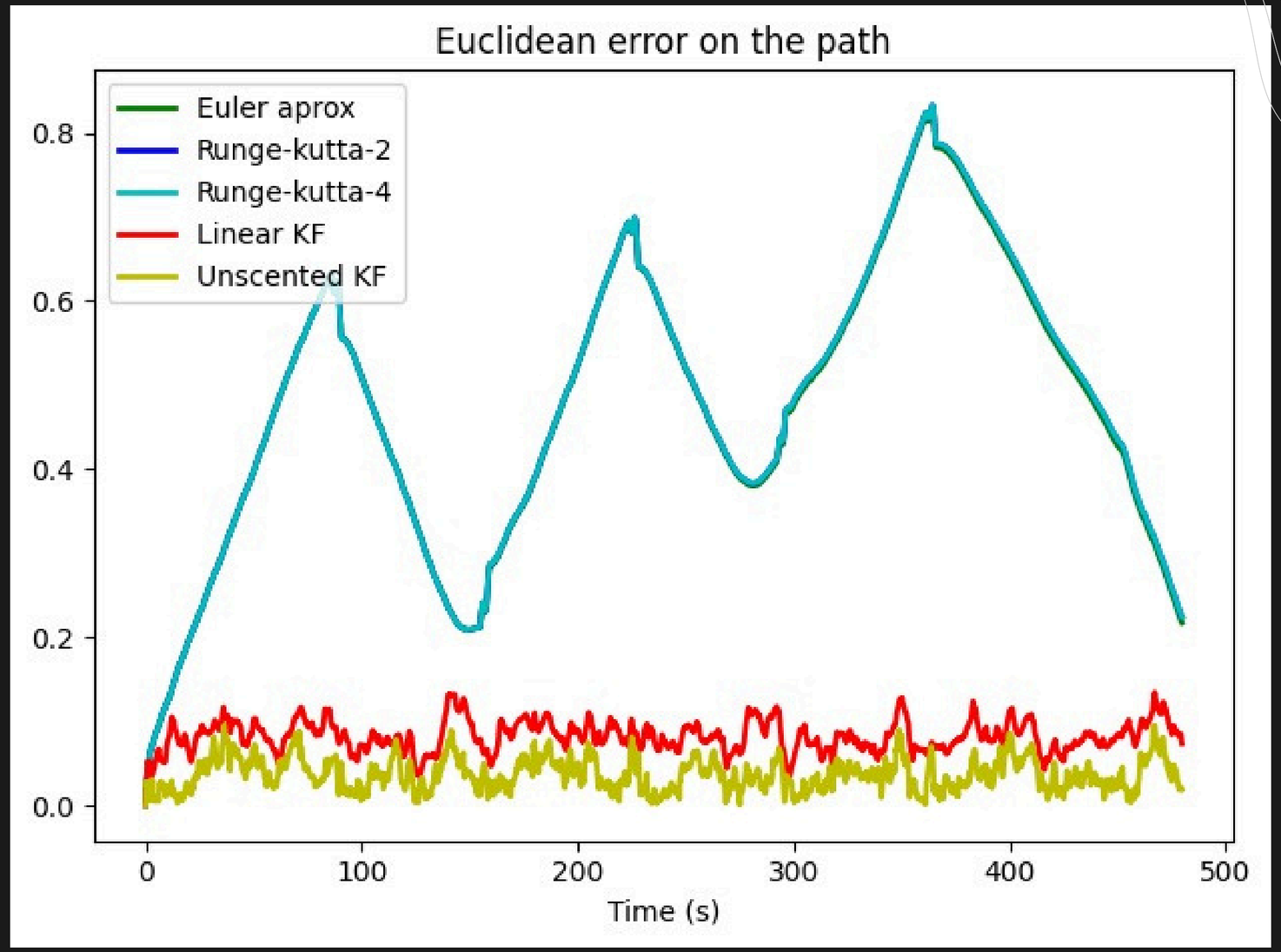








SIMULATIONS



CONTROL STRATEGIES

I/O Linearization

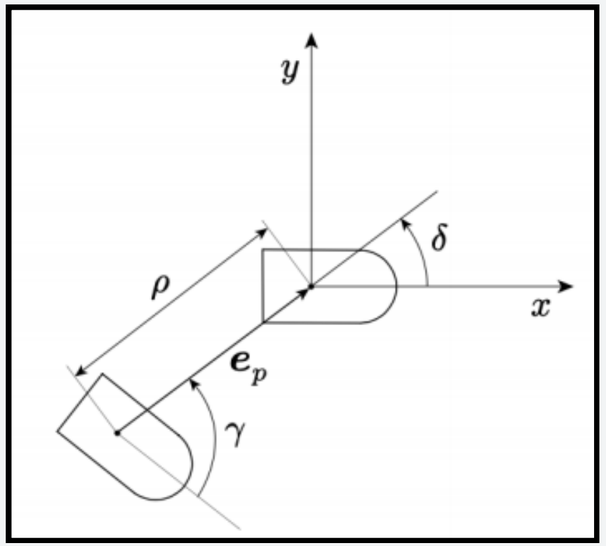
- Linearization of the system through feedback.
- A point B at a distance |b| from the center of the wheel is choosen along the sagittal axis
- Orientation uncontrolled
- Outputs:

$$y_1 = x + b\cos\theta$$
$$y_2 = y + b\sin\theta$$

- We can design

$$u_1 = \dot{y}_{1,d} + k_1(y_{1,d} - y_1)$$
$$u_2 = \dot{y}_{2,d} + k_2(y_{2,d} - y_2)$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = T(\theta)^{-1} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$



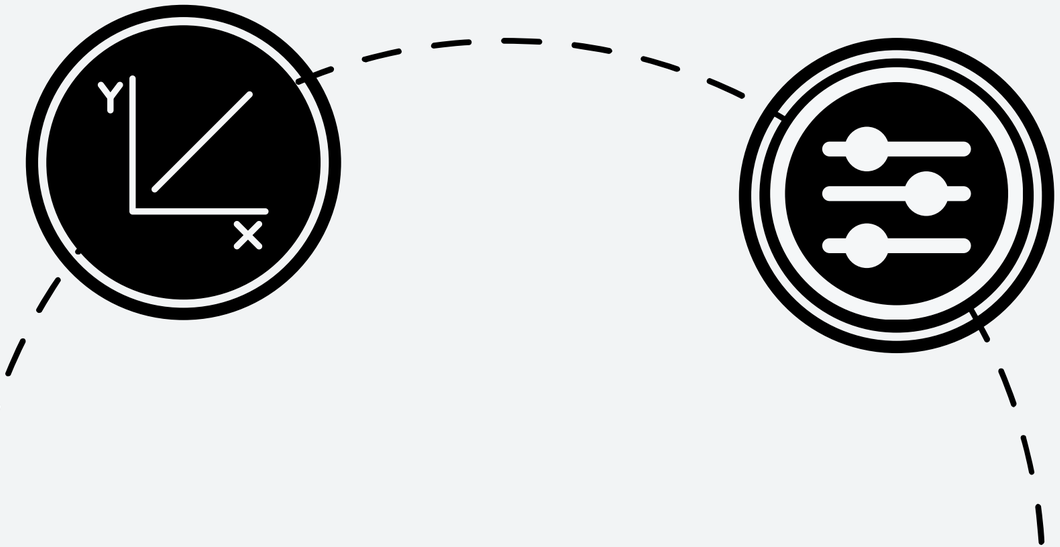
Posture Regulation

- We use $q_d = [0 \quad 0 \quad 0]^T$
- Expressing the problem in polar coordinates

$$\rho = \sqrt{x^2 + y^2}$$
$$\gamma = \operatorname{atan2}(y, x) - \theta + \pi$$
$$\delta = \gamma + \theta$$

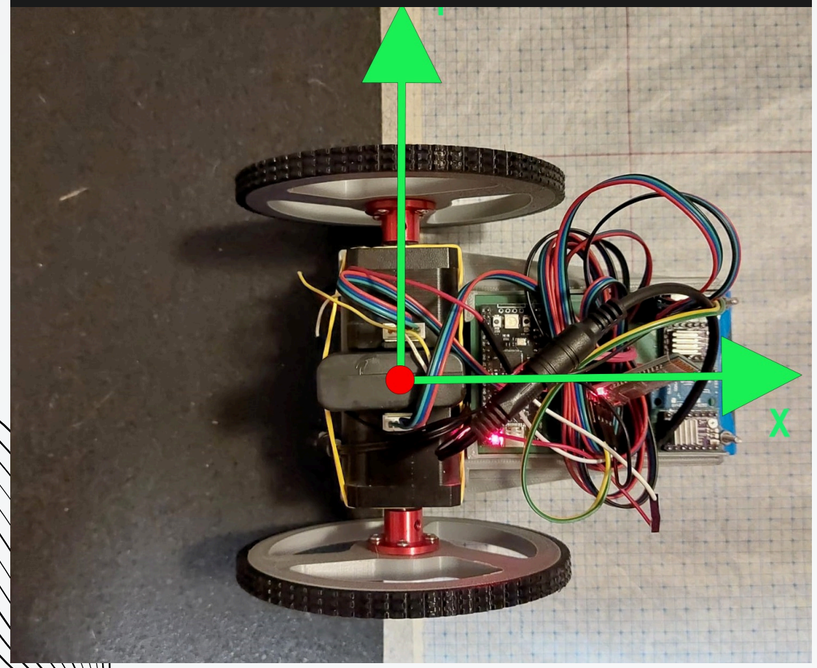
- We can design the following feedback controller

$$v = k_1 \rho \cos\gamma$$
$$\omega = k_2\gamma + k_1 \sin\gamma\cos\gamma\left(1 + k_3\frac{\delta}{\gamma}\right)$$

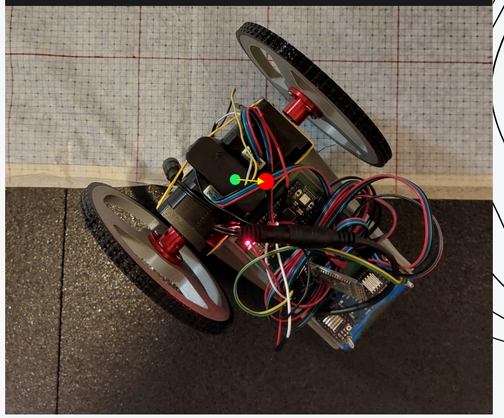


I/O LINEARIZATION

Start Position

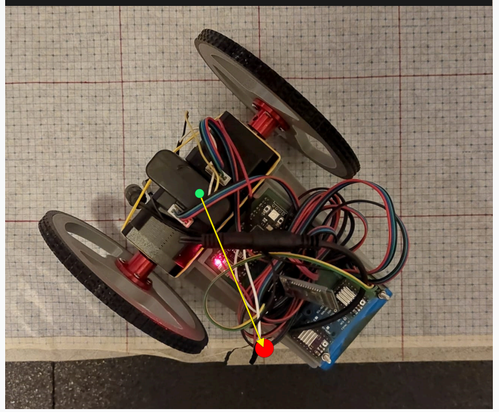


End Position



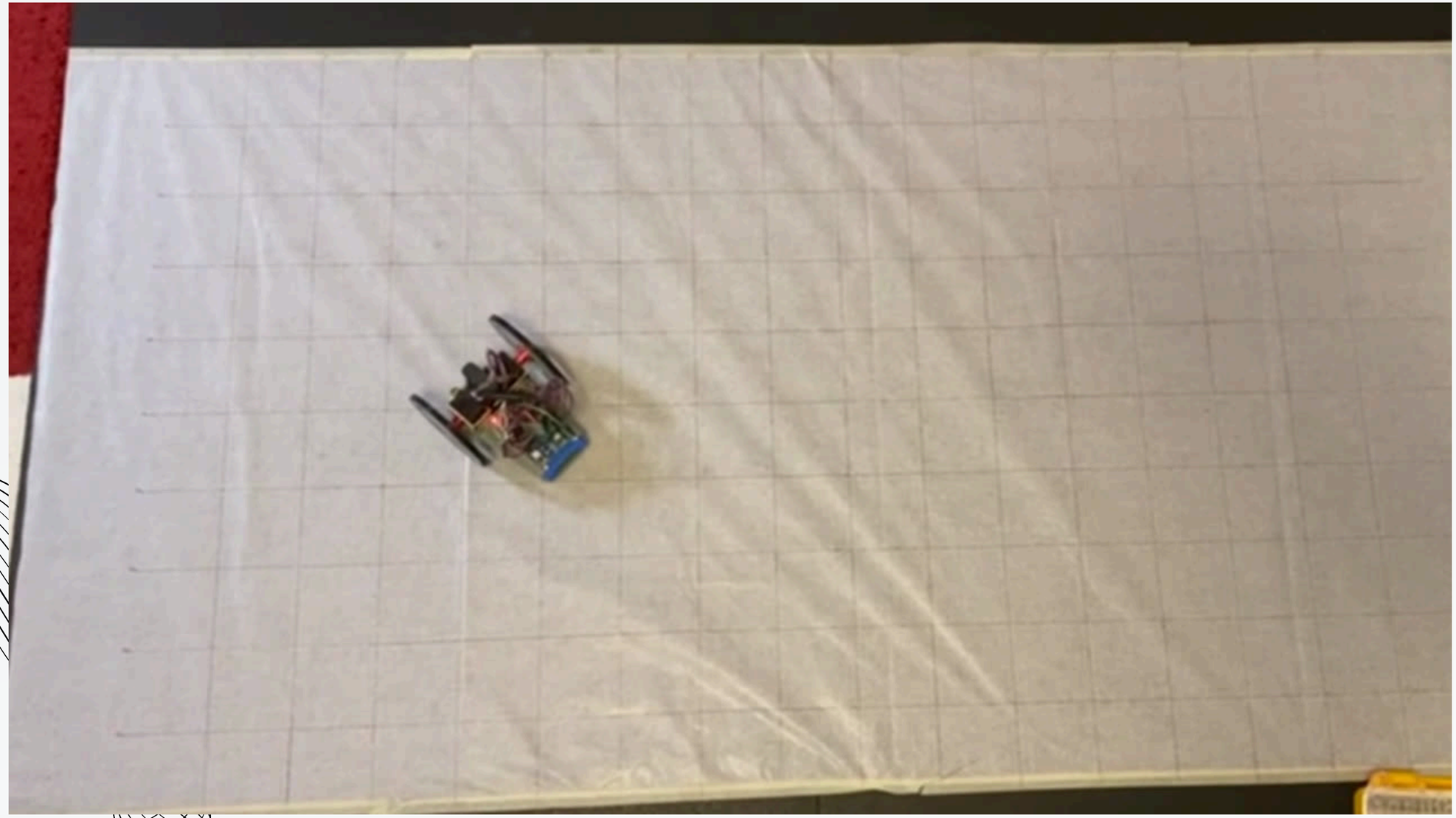
K1=0.01 K2=0.01

End Position

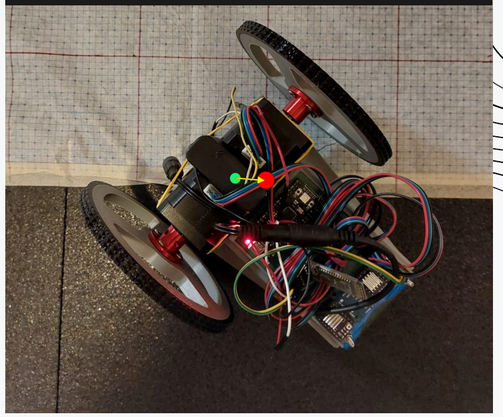


K1=0.1 K2=0.1

I/O LINEARIZATION



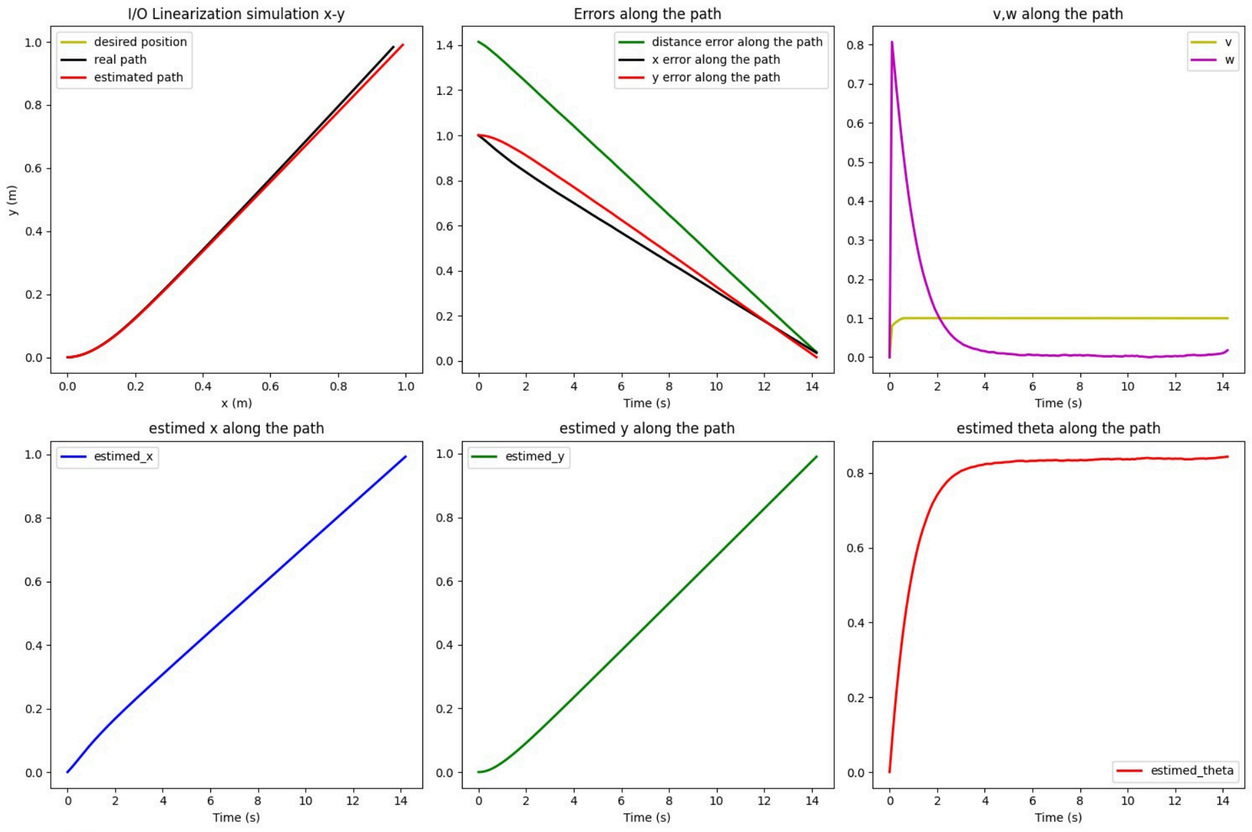
End Position



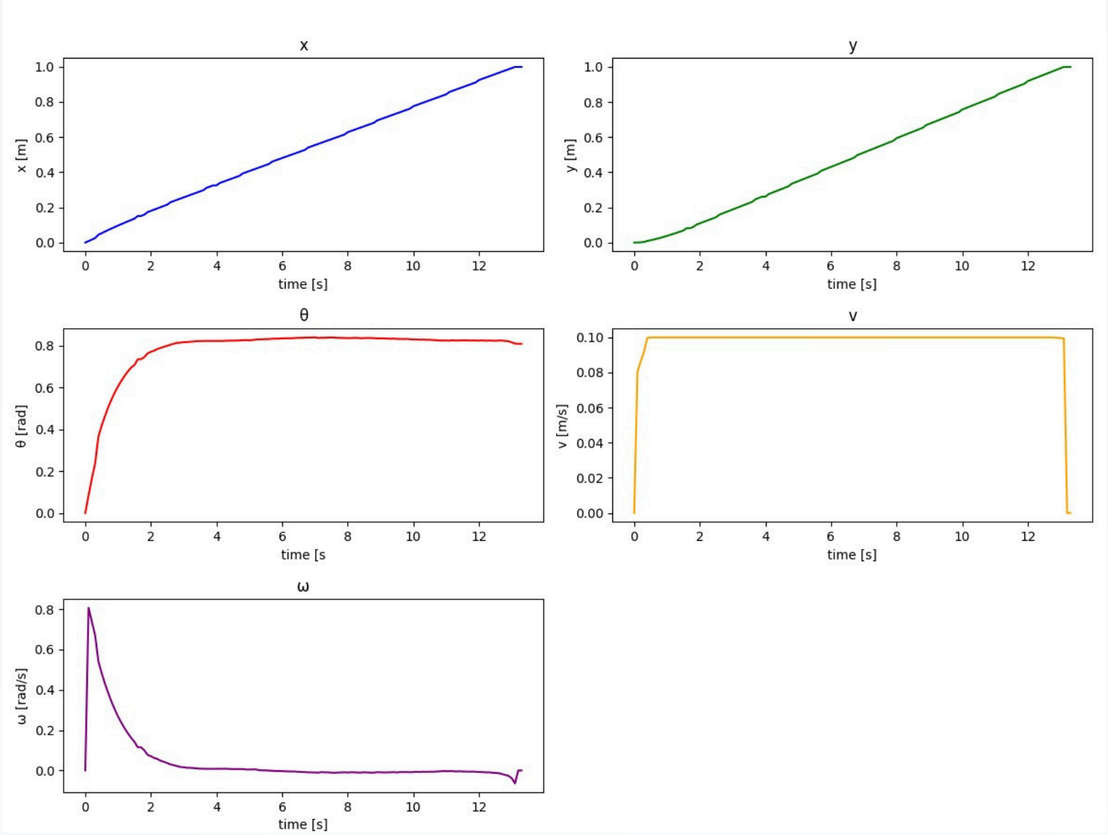
K1=0.01 K2=0.01

I/O LINEARIZATION

K1=0.01 K2=0.01



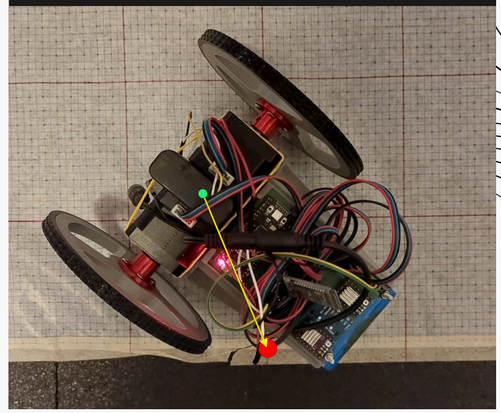
SIMULATION



TEST

I/O LINEARIZATION

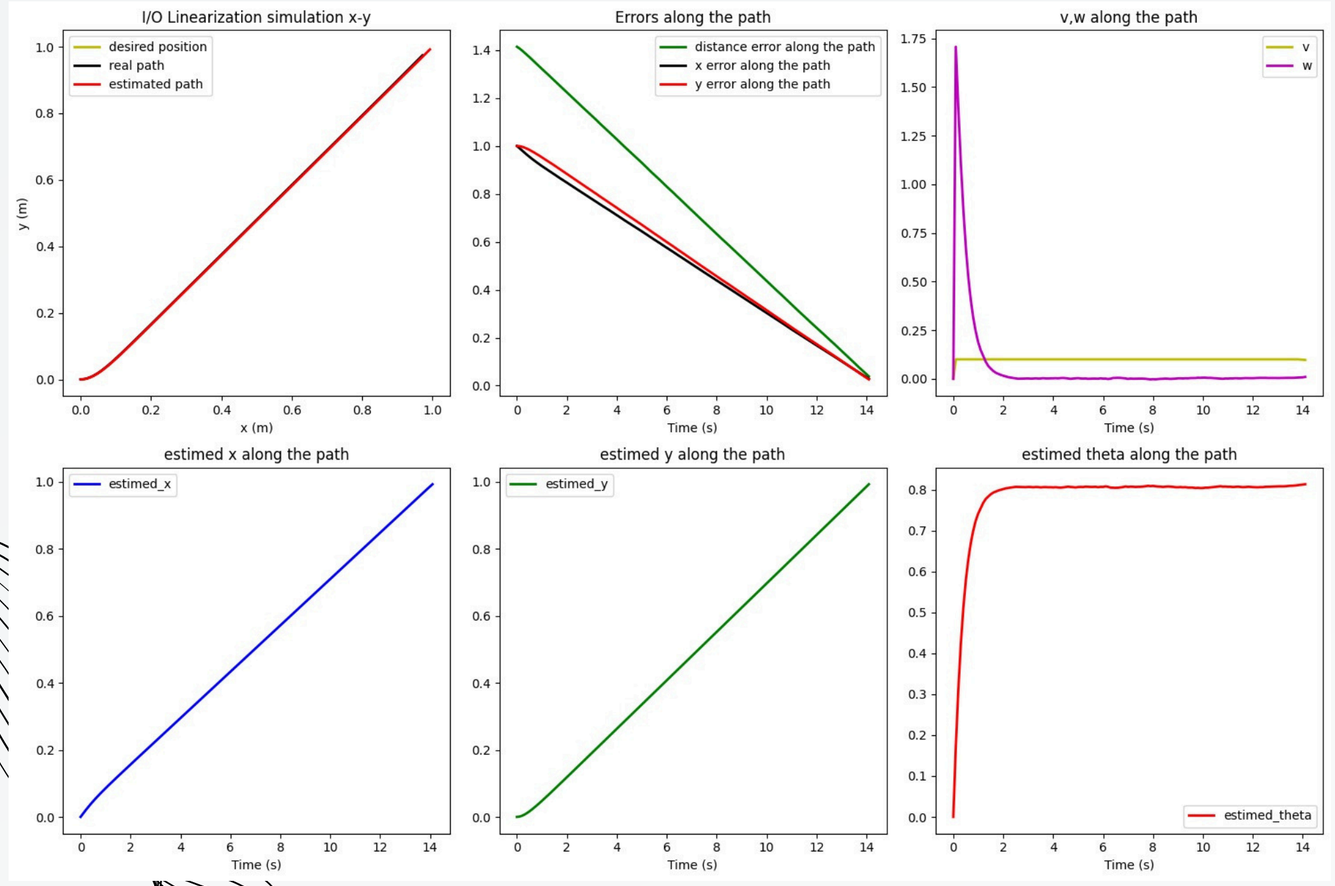
End Position

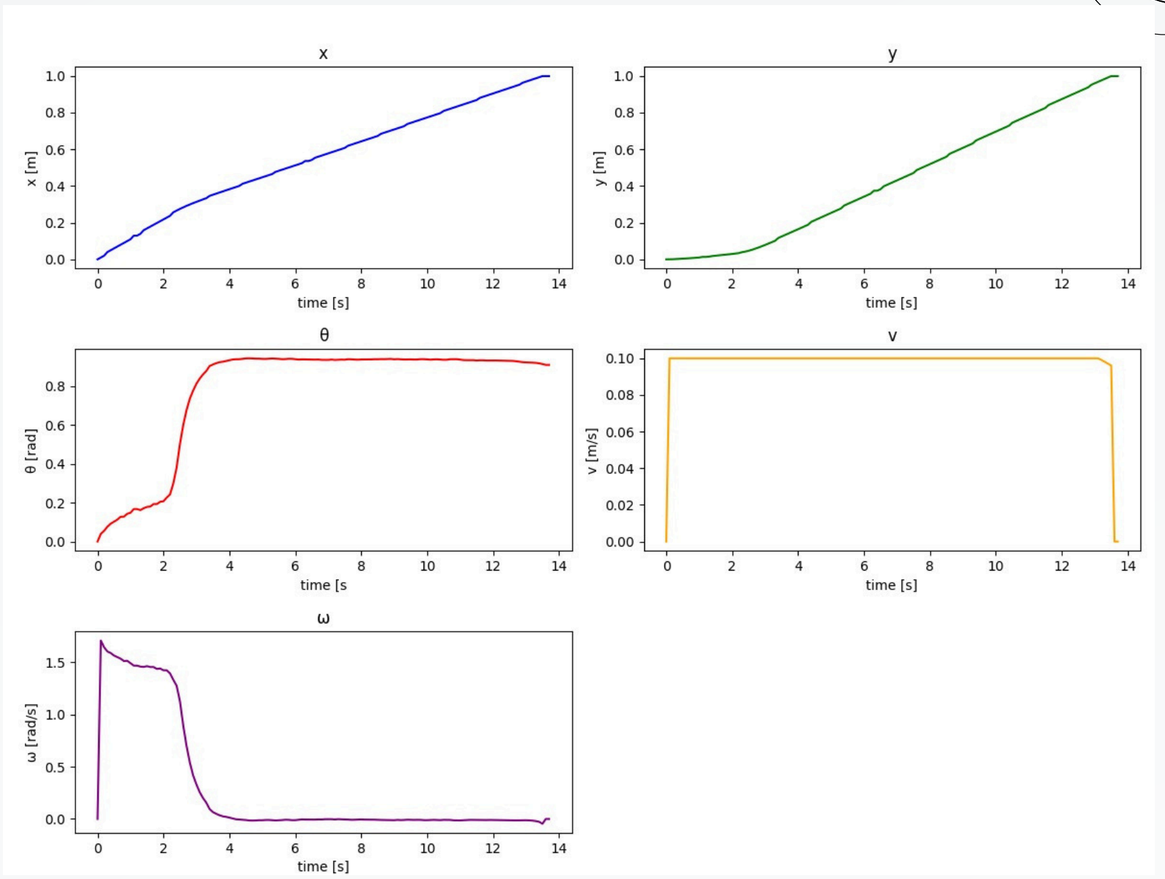


K1=0.1 K2=0.1

I/O LINEARIZATION

K1=0.1 K2=0.1



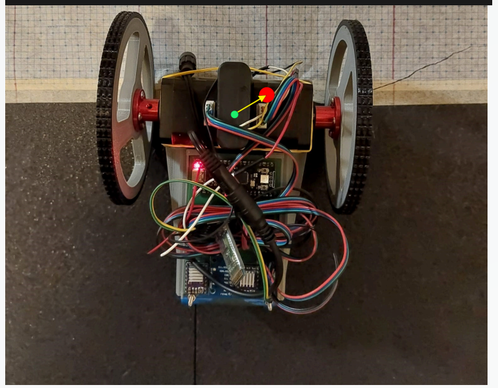


SIMULATION

TEST

POSTURE REGULATION

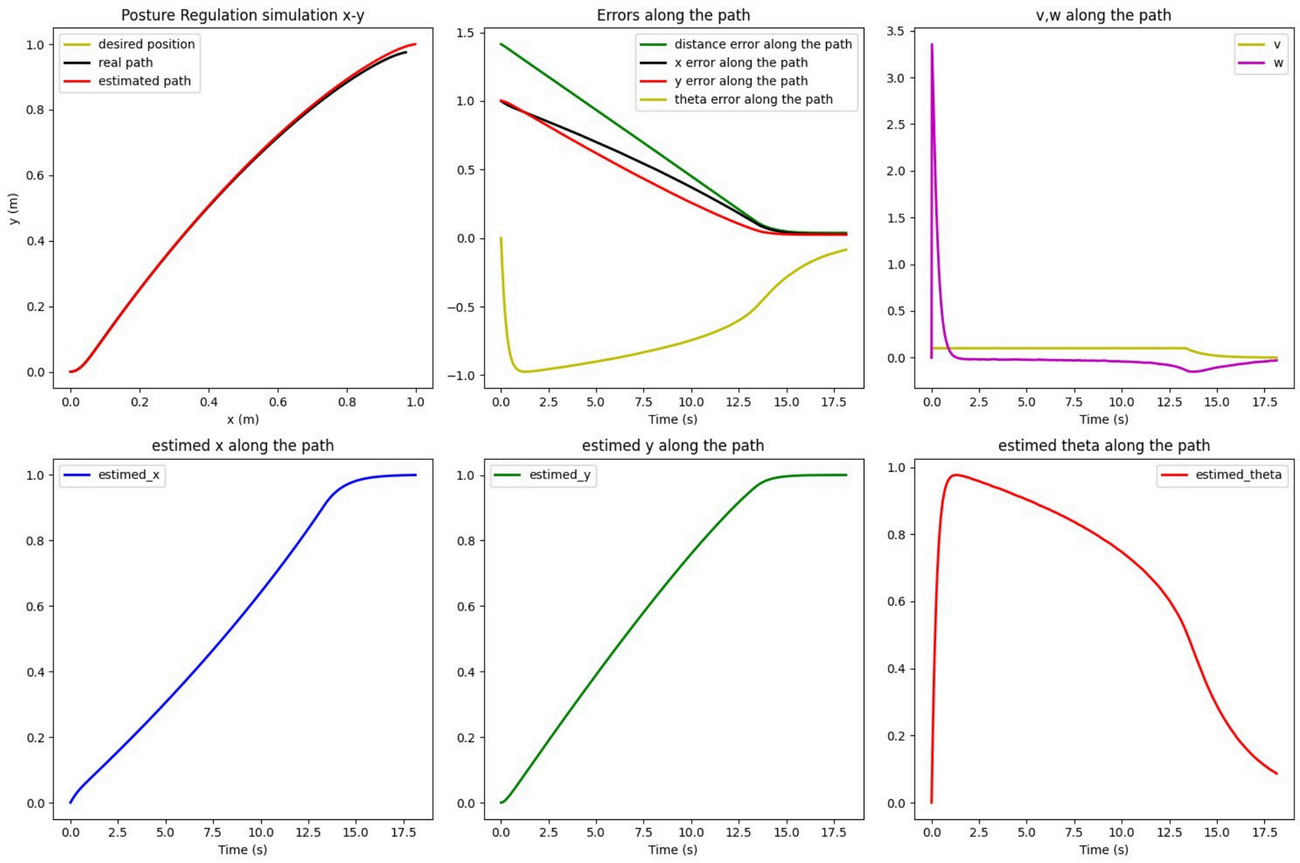
End Position
(0,0,0)



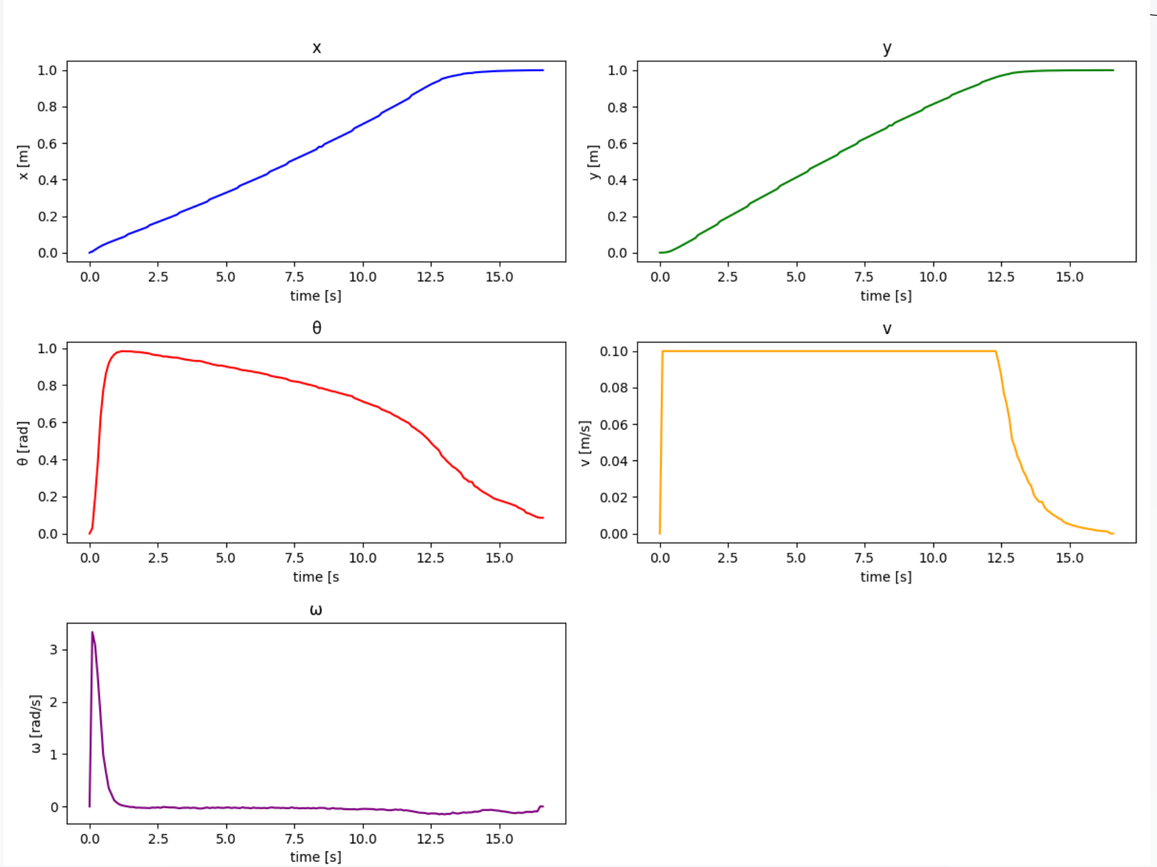
K1=1 K2=3 K3=1

POSTURE REGULATION

K1=1 K2=3 K3=1



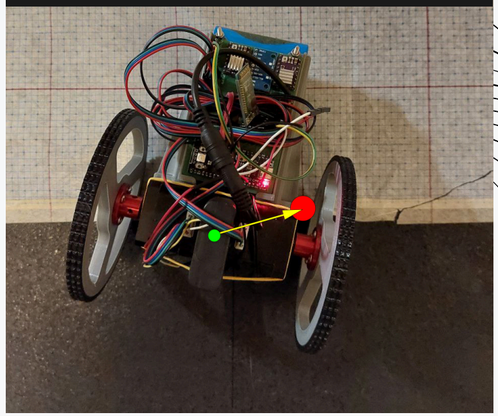
SIMULATION



TEST

POSTURE REGULATION

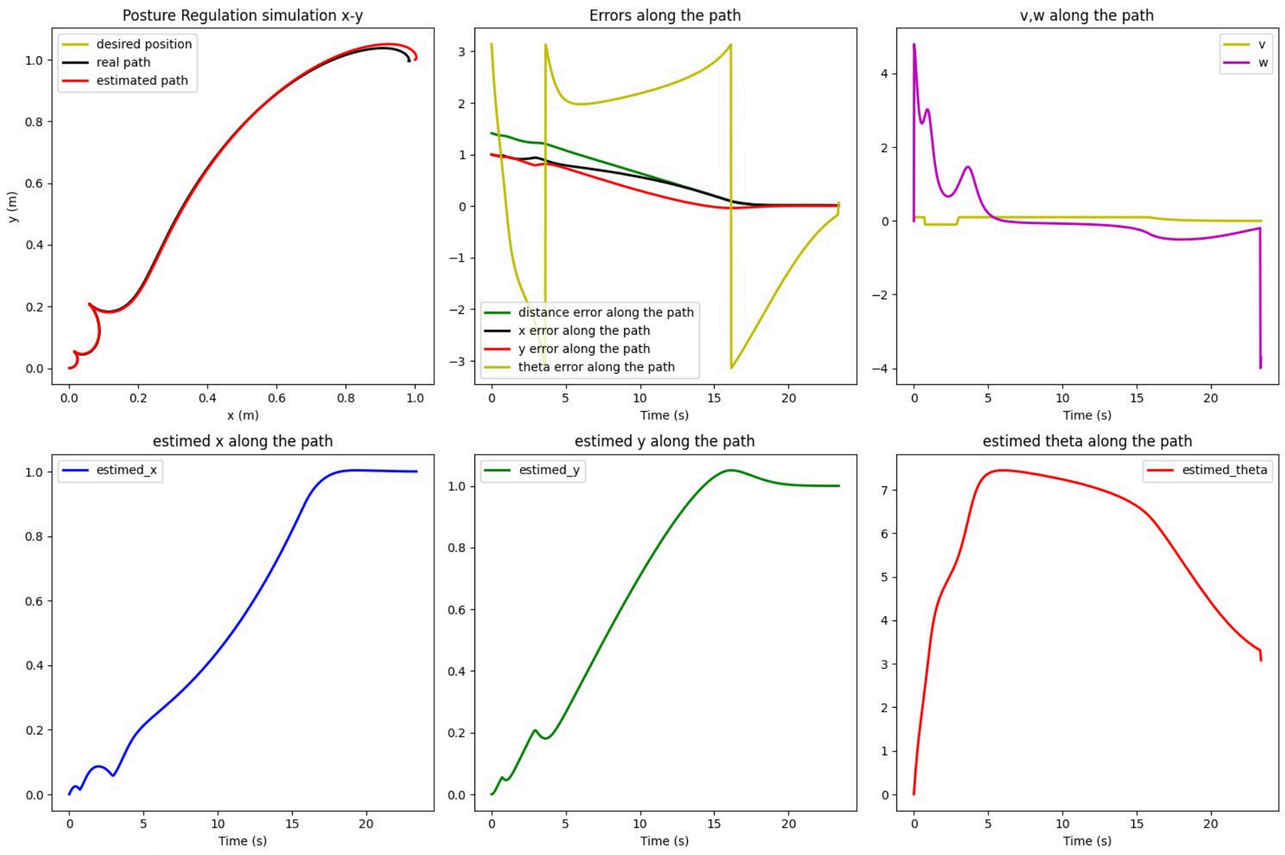
End Position
(0,0,pi)



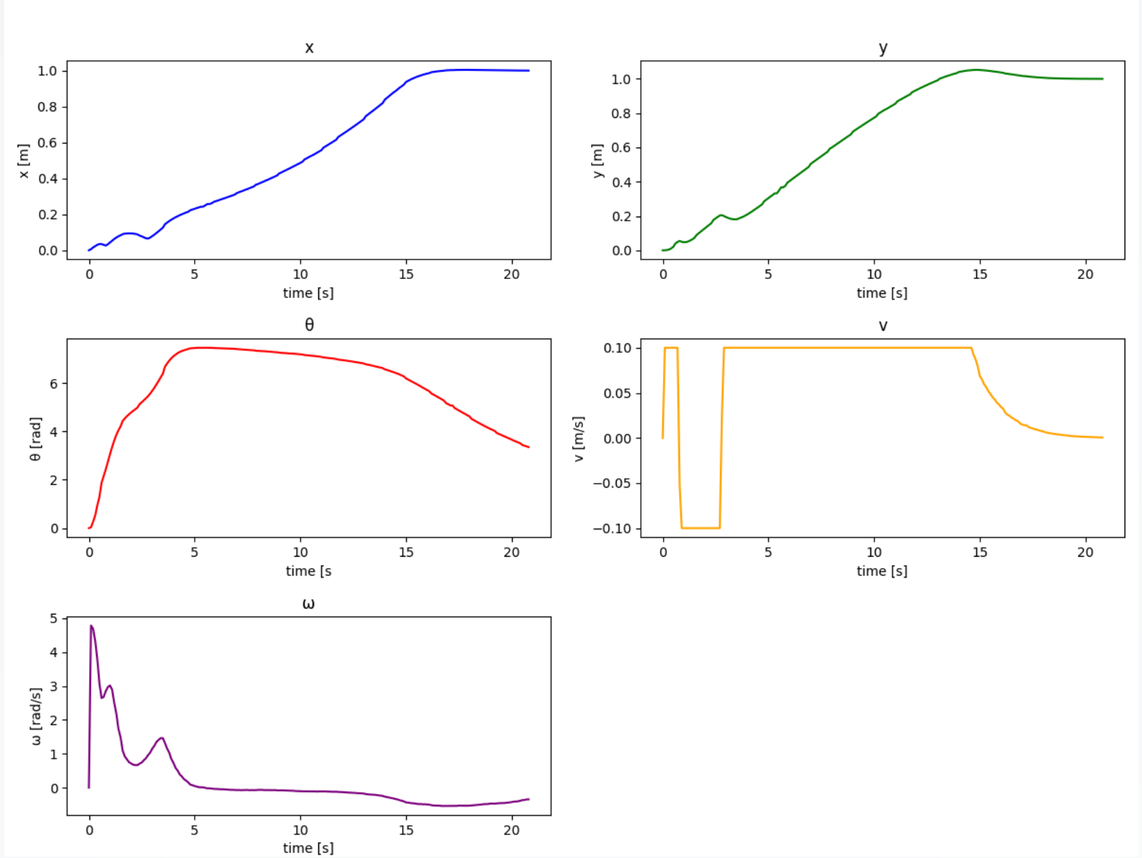
K1=1 K2=0.6 K3=0.2

POSTURE REGULATION

K1=1 K2=0.6 K3=0.2 WITH FINAL CONFIGURATIO (0,0,PI)



SIMULATION

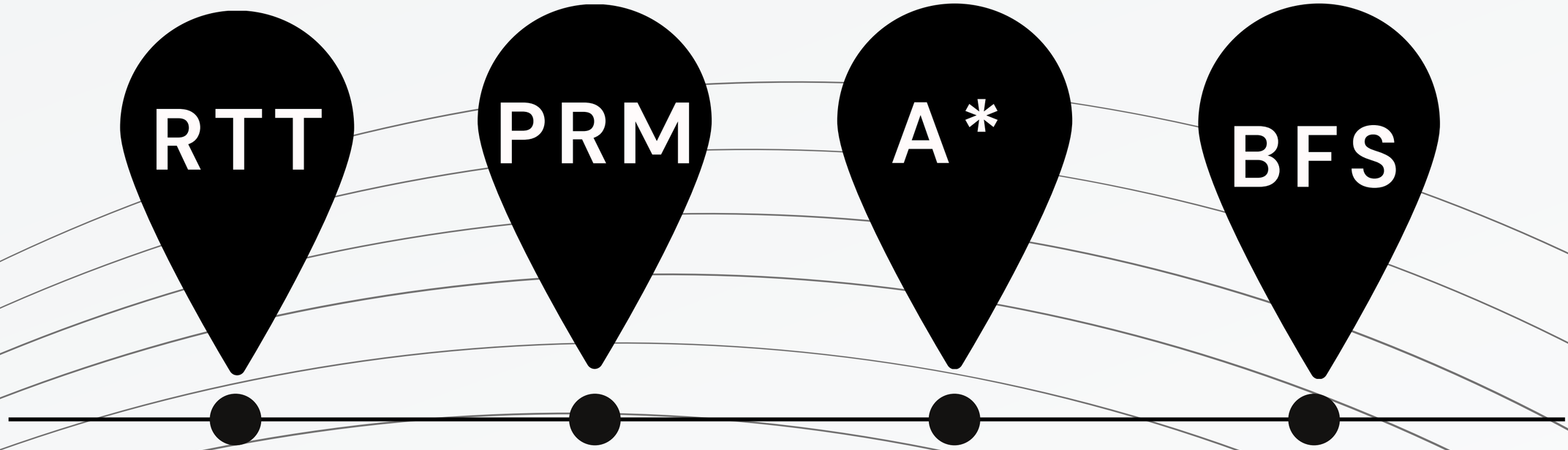


TEST

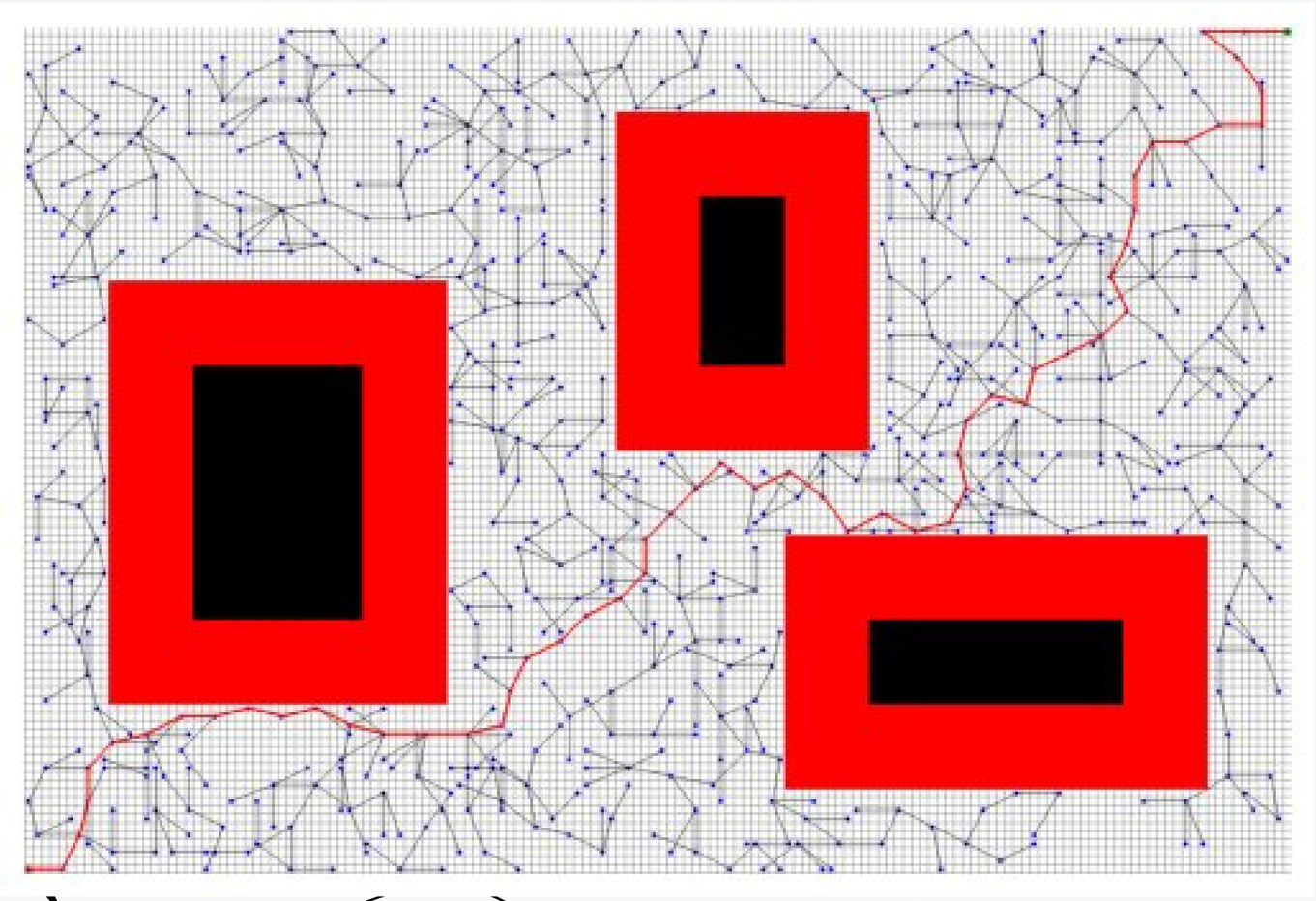
MOTION PLANNING

NODE GENERATION

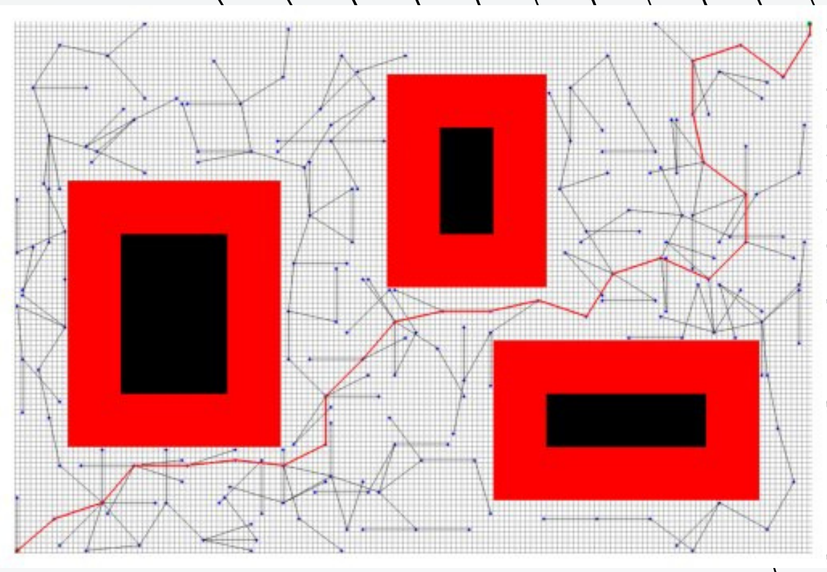
SEARCH ALGORITHMS



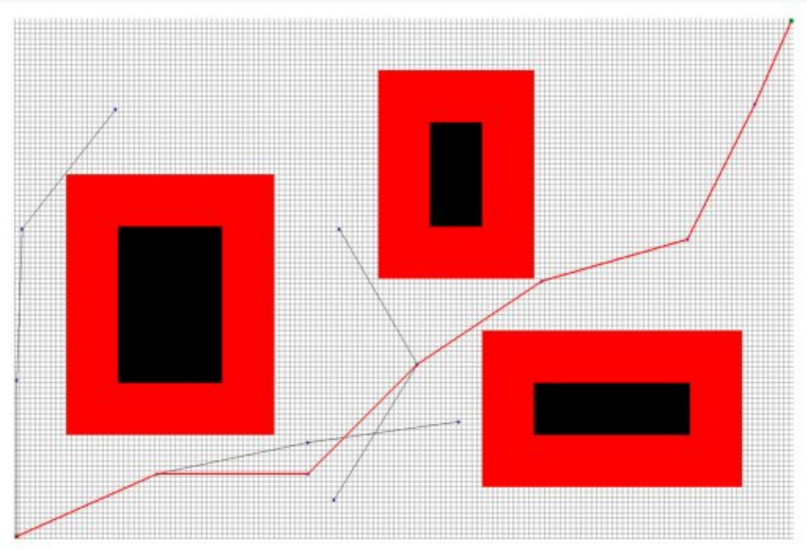
RRT



DELTA=5

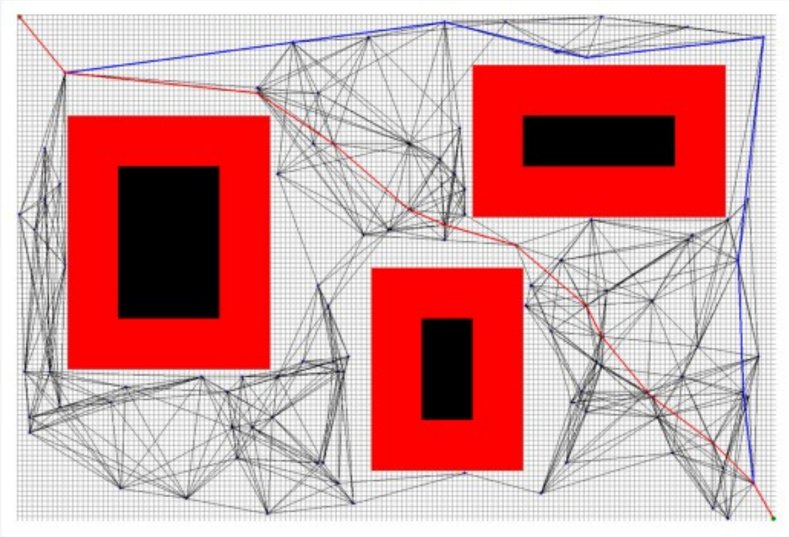


DELTA=10



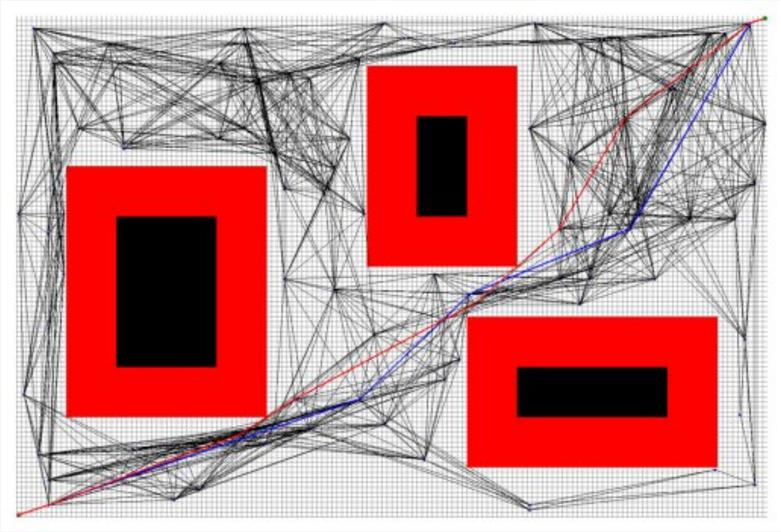
DELTA=30

PRM

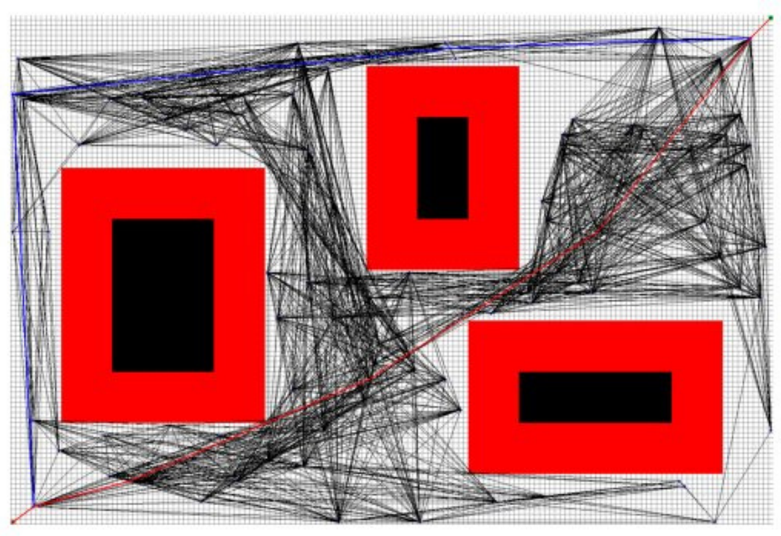


K=5

- A* algorithm: lenght 1876 with 14 points (RED)
- BFS algorithm: lenght 2504 with 10 points (BLUE)

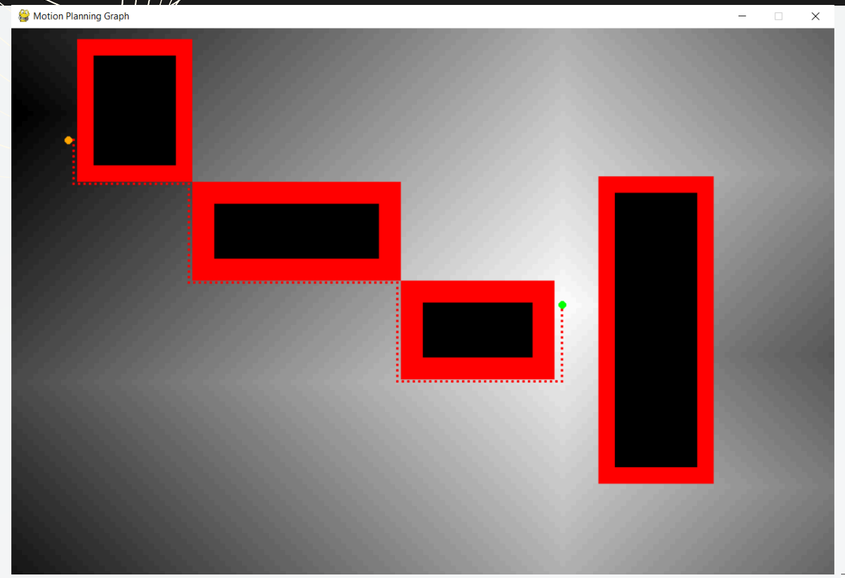


K=20

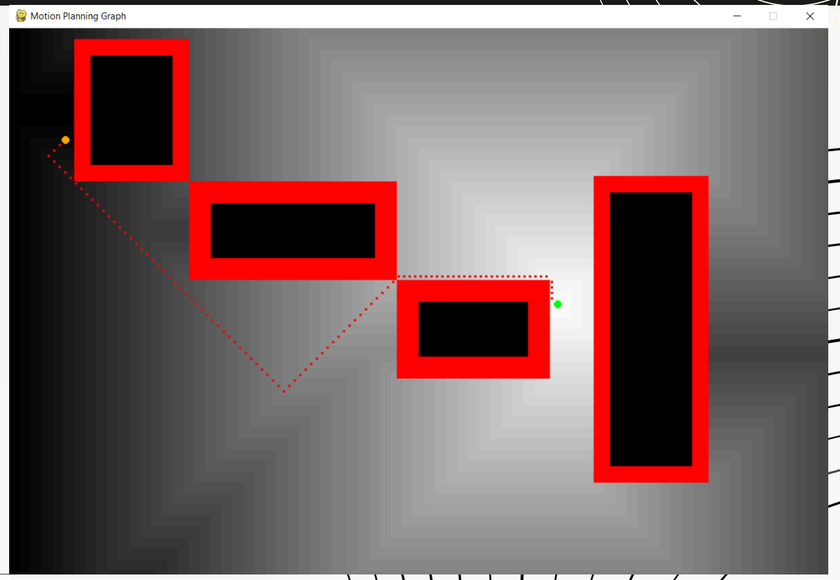


K=50

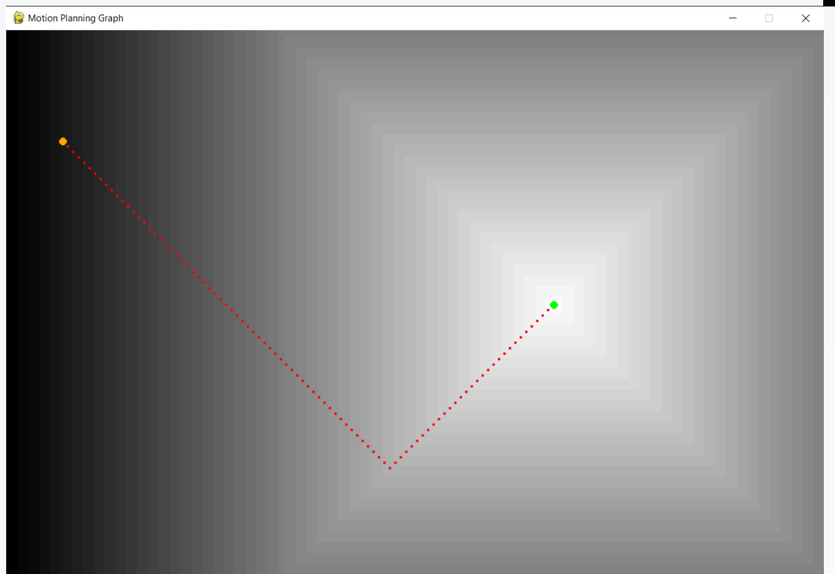
NAVIGATION FUNCTION



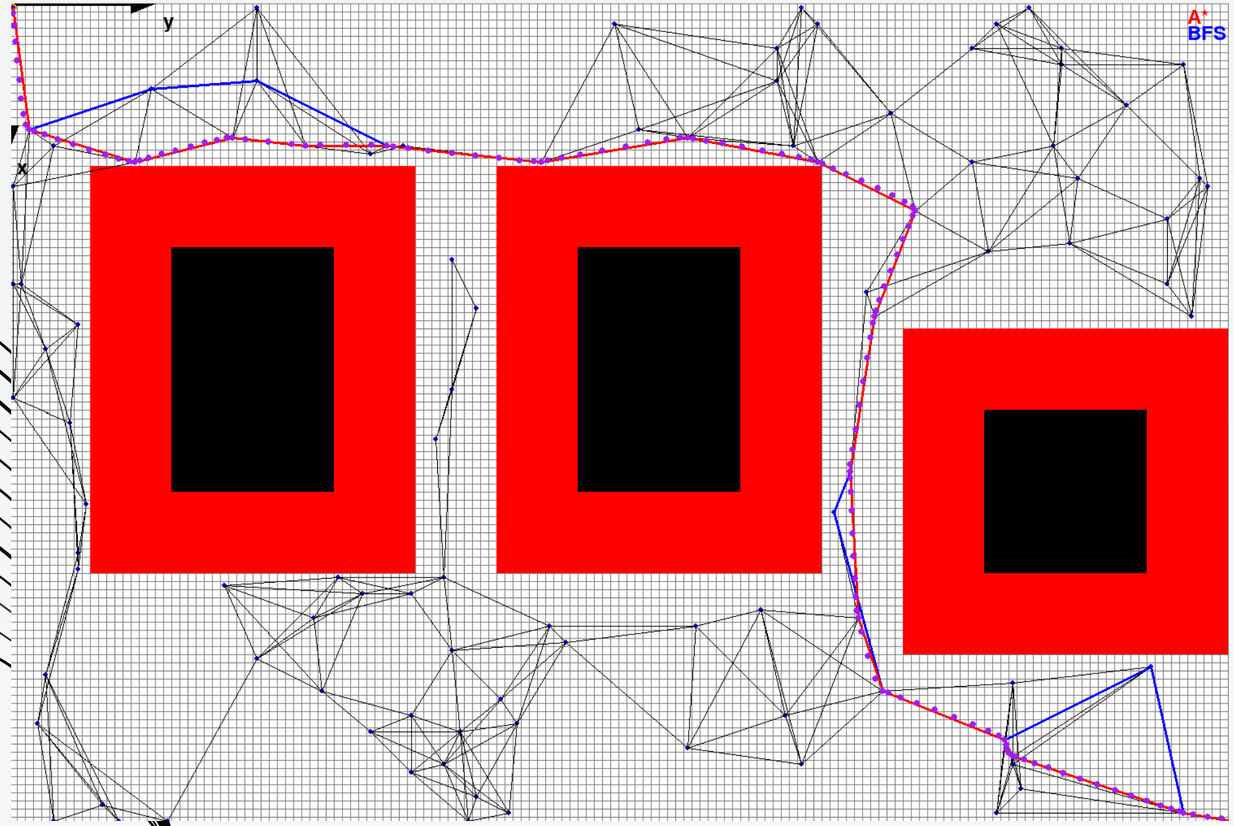
WITH OBSTACLE

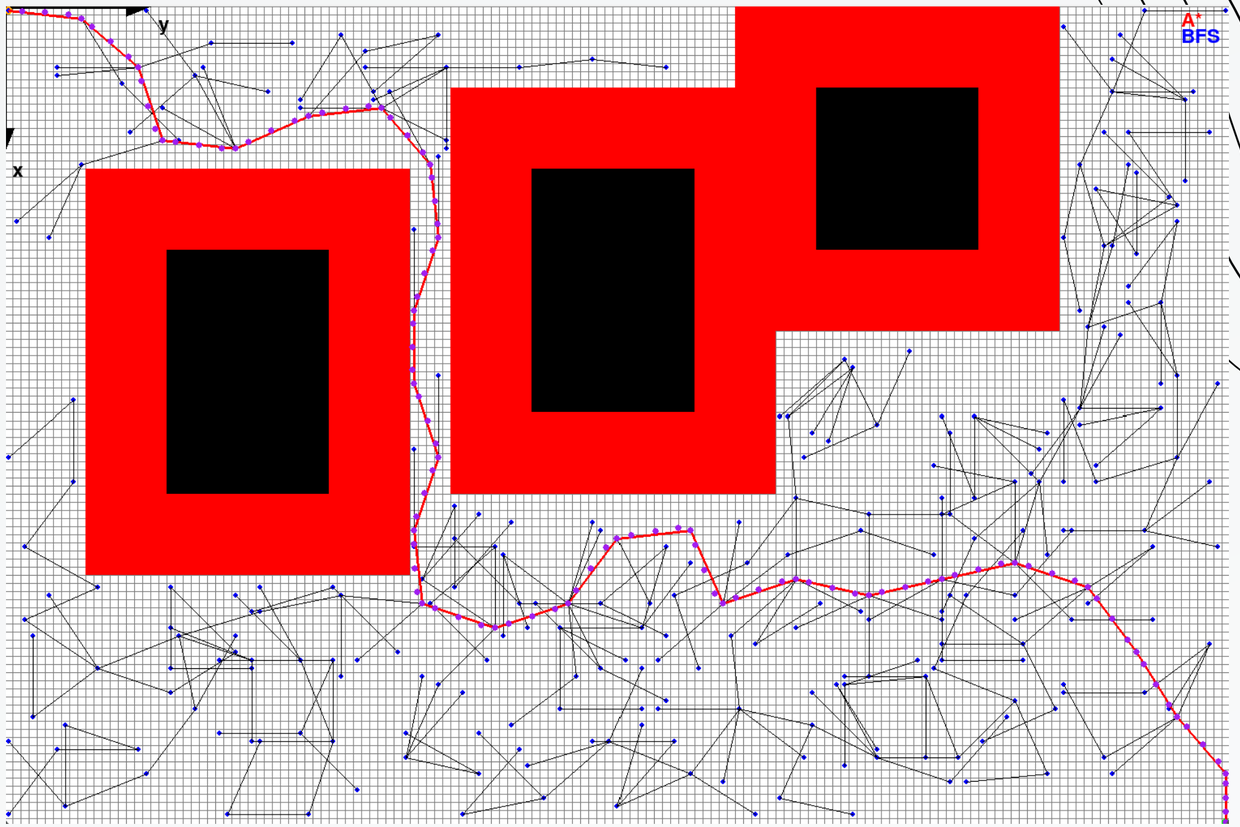


FREE SPACE



PRM VS RRT





TRAJECTORY PLANNING

- Input: A* Path generated by Motion Planning
- Cartesian polynomials trajectories

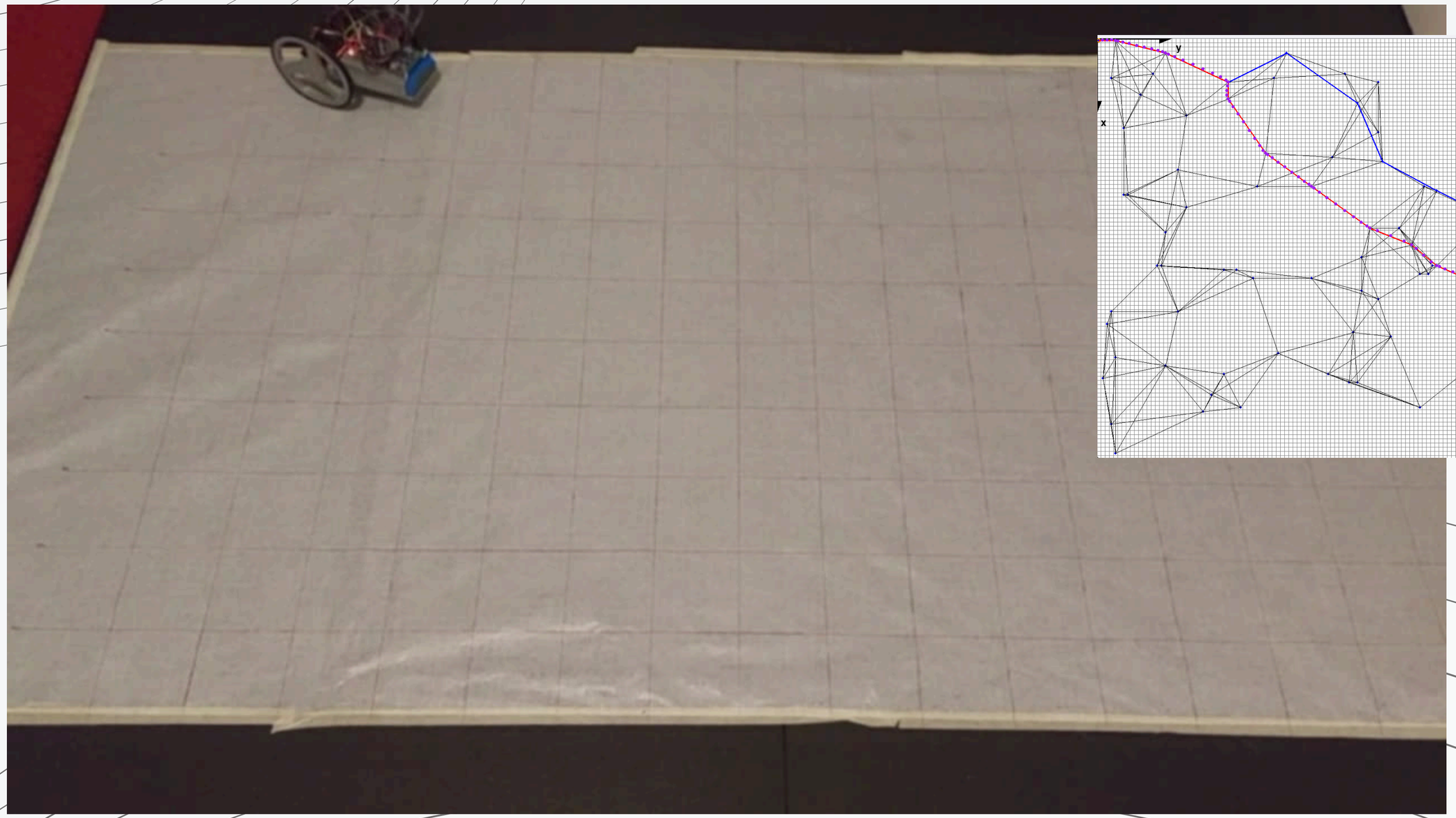
$$x(t) = s(t)^3 x_f - (s(t)-1)^3 x_i + \alpha_x s(t)^2 (s(t)-1) + \beta_x s(t)(s(t)-1)^2$$

$$y(t) = s(t)^3 y_f - (s(t)-1)^3 y_i + \alpha_y s(t)^2 (s(t)-1) + \beta_y s(t)(s(t)-1)^2$$

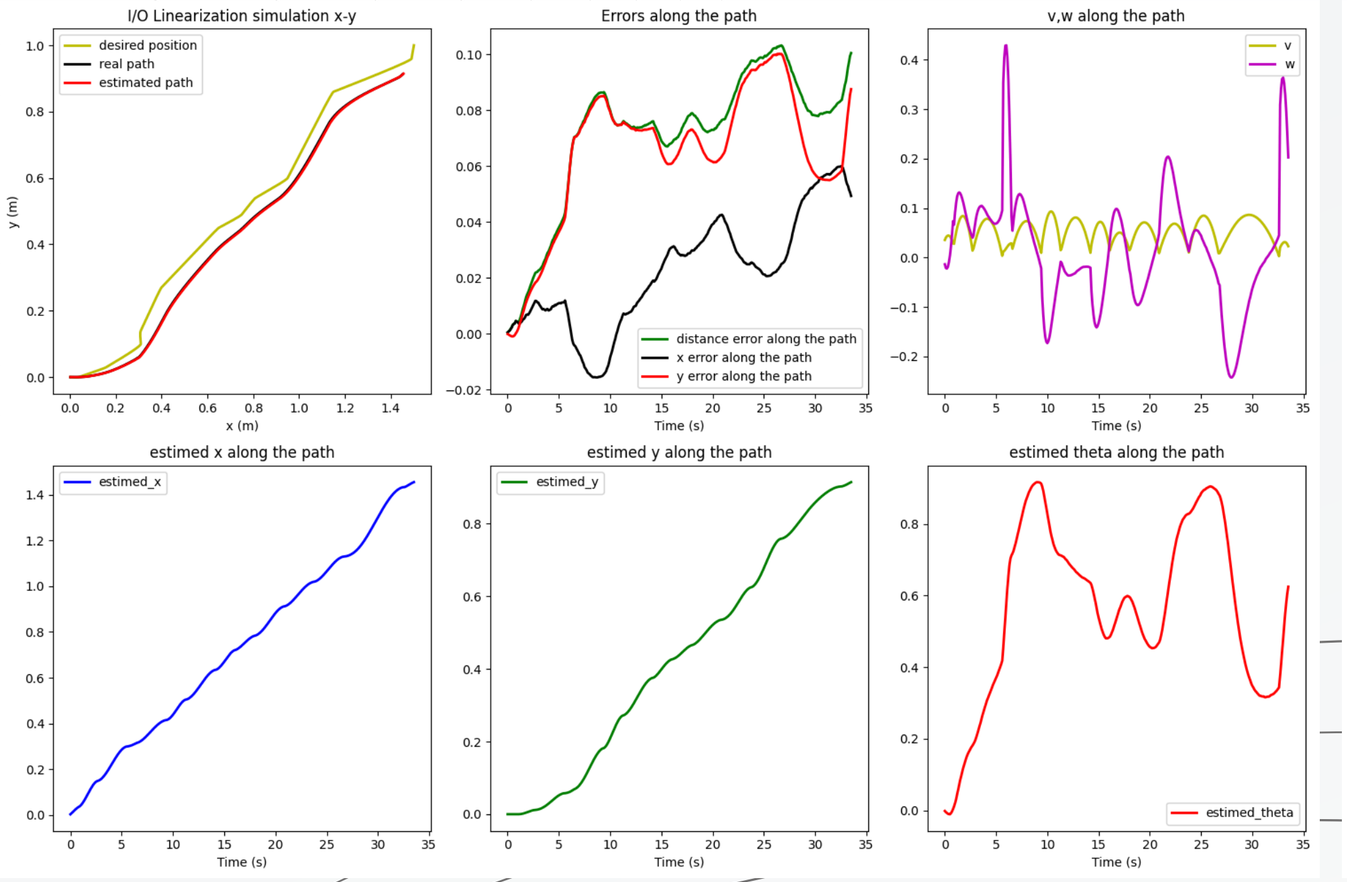
$$\alpha_x = k\cos\theta_f - 3x_f$$
$$\alpha_y = k\sin\theta_f - 3y_f$$
$$\beta_x = k\cos\theta_i + 3x_i$$
$$\beta_y = k\sin\theta_i + 3y_i$$

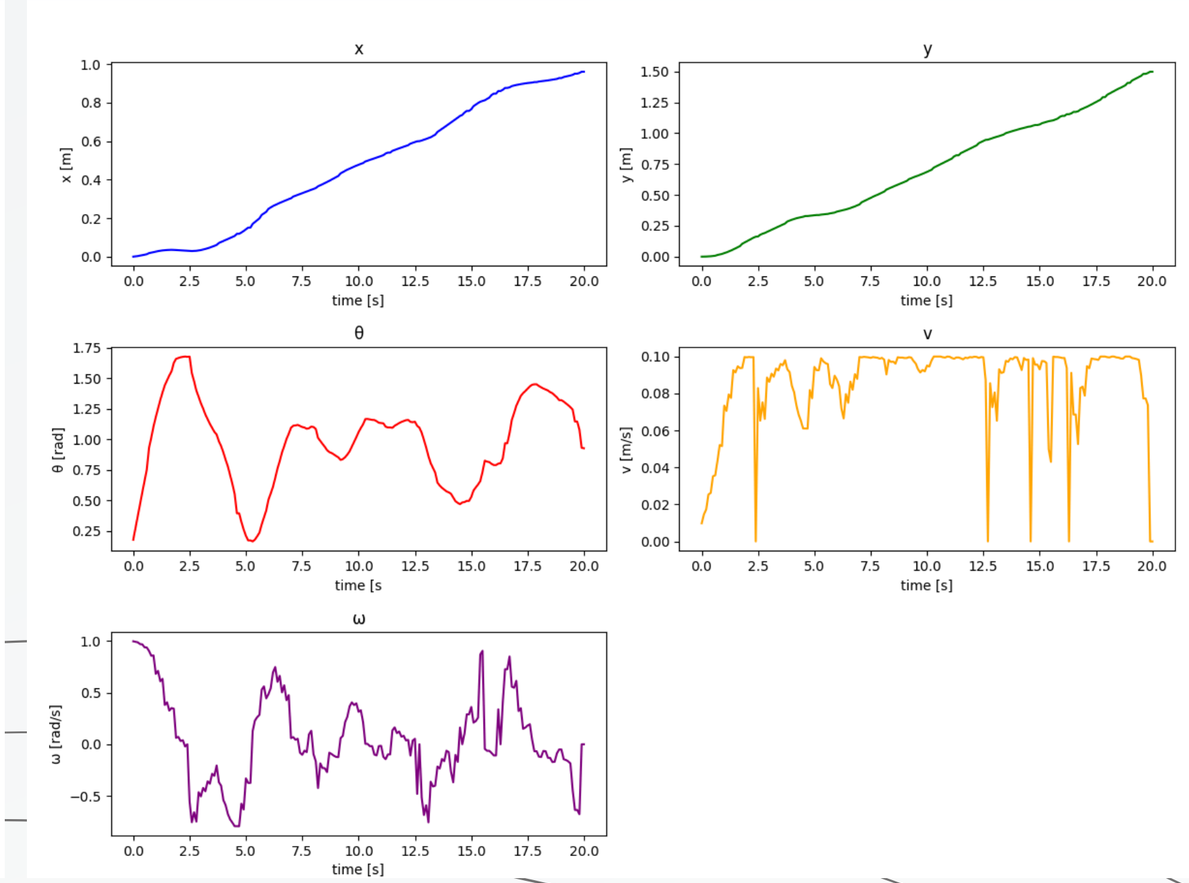
- Tuning k parameter
- Abscissa for each segment:
  - Linear profile (const. velocity)
  - Trapezoidal velocity profile
  - Cubic profile
- Trajectory scaling to respect motors saturation [linear, cubic]

TRAJECTORY PLANNING - LINEAR PROFILE



TRAJECTORY PLANNING - LINEAR PROFILE

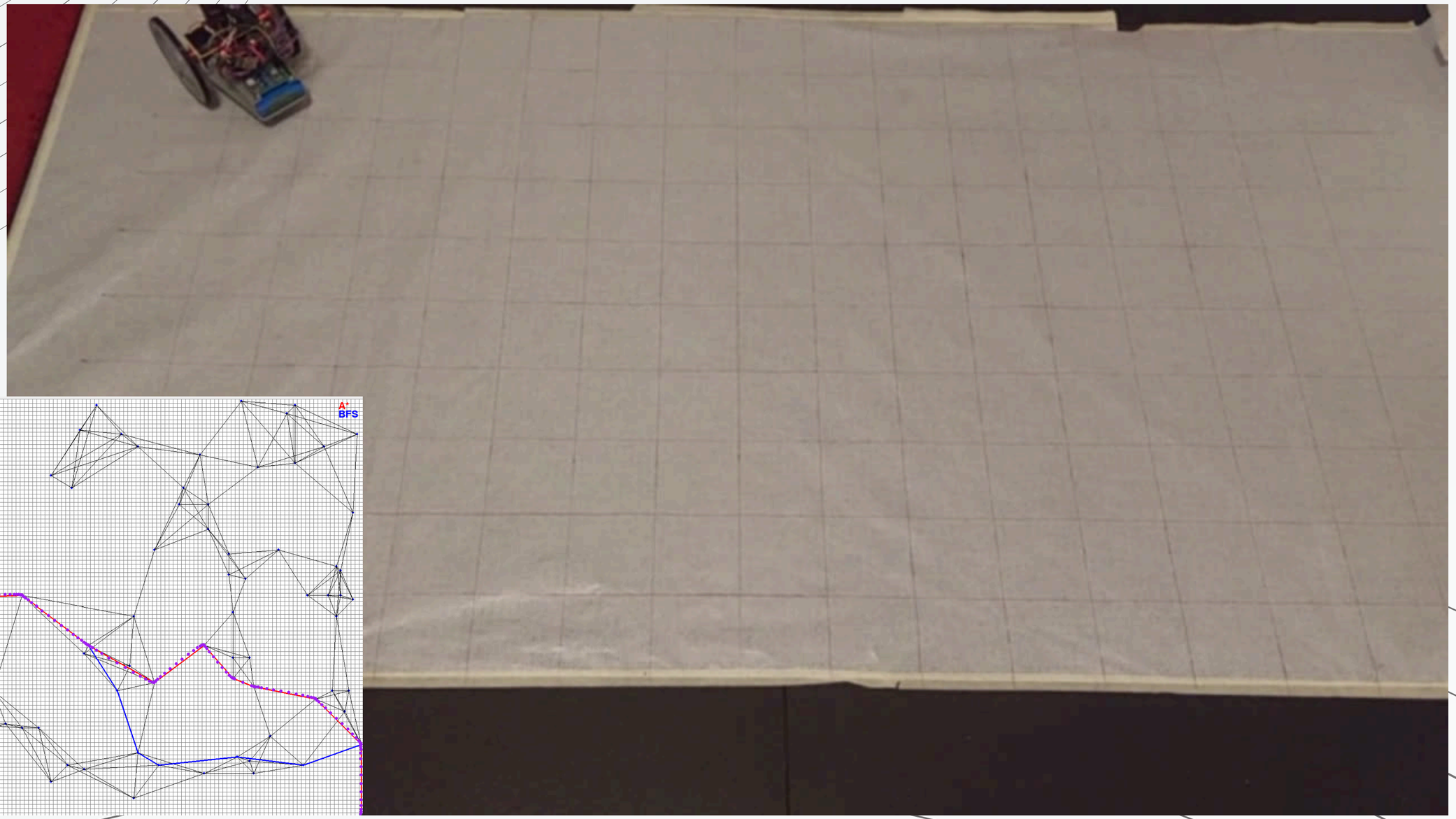




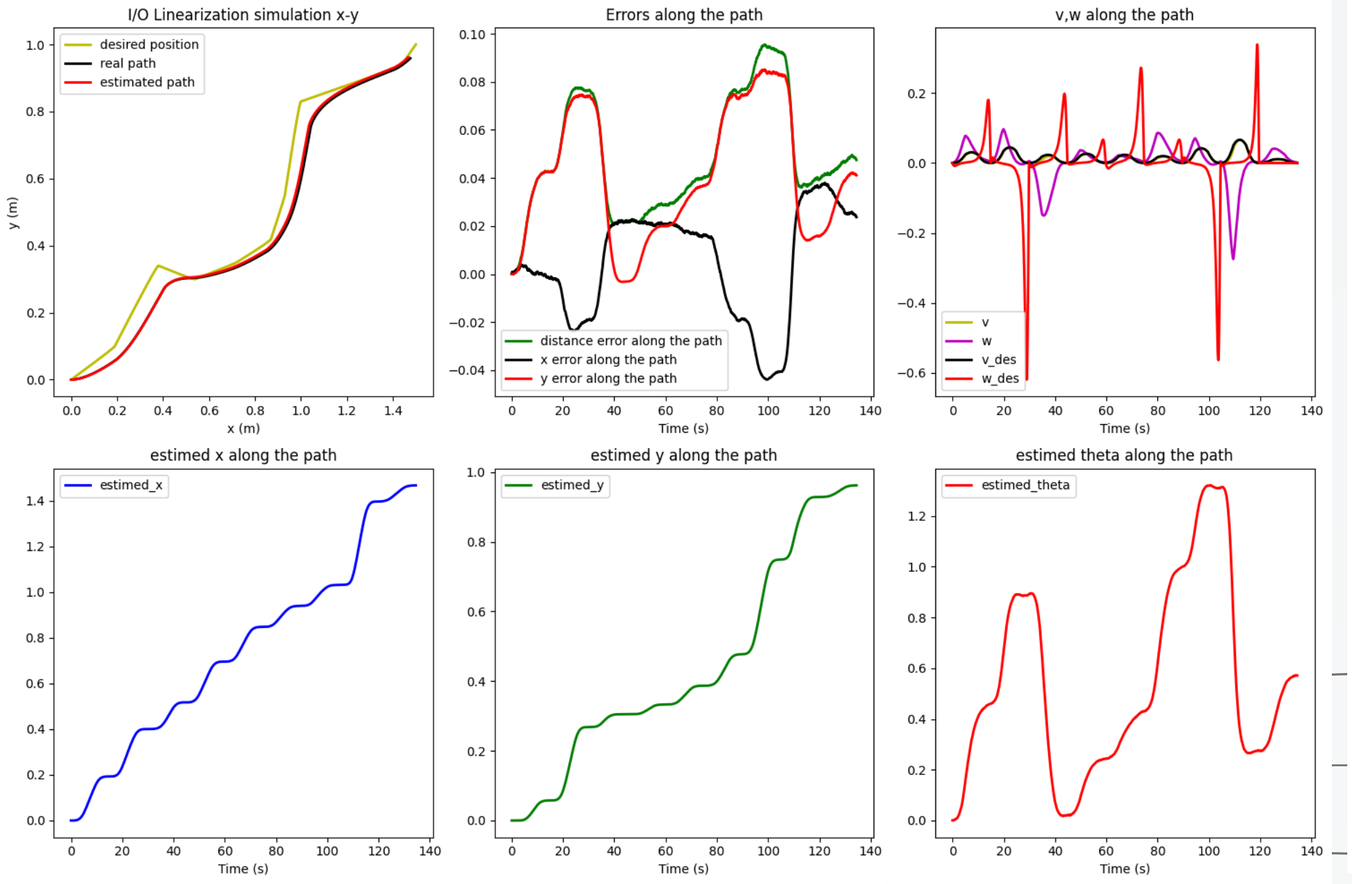
SIMULATION

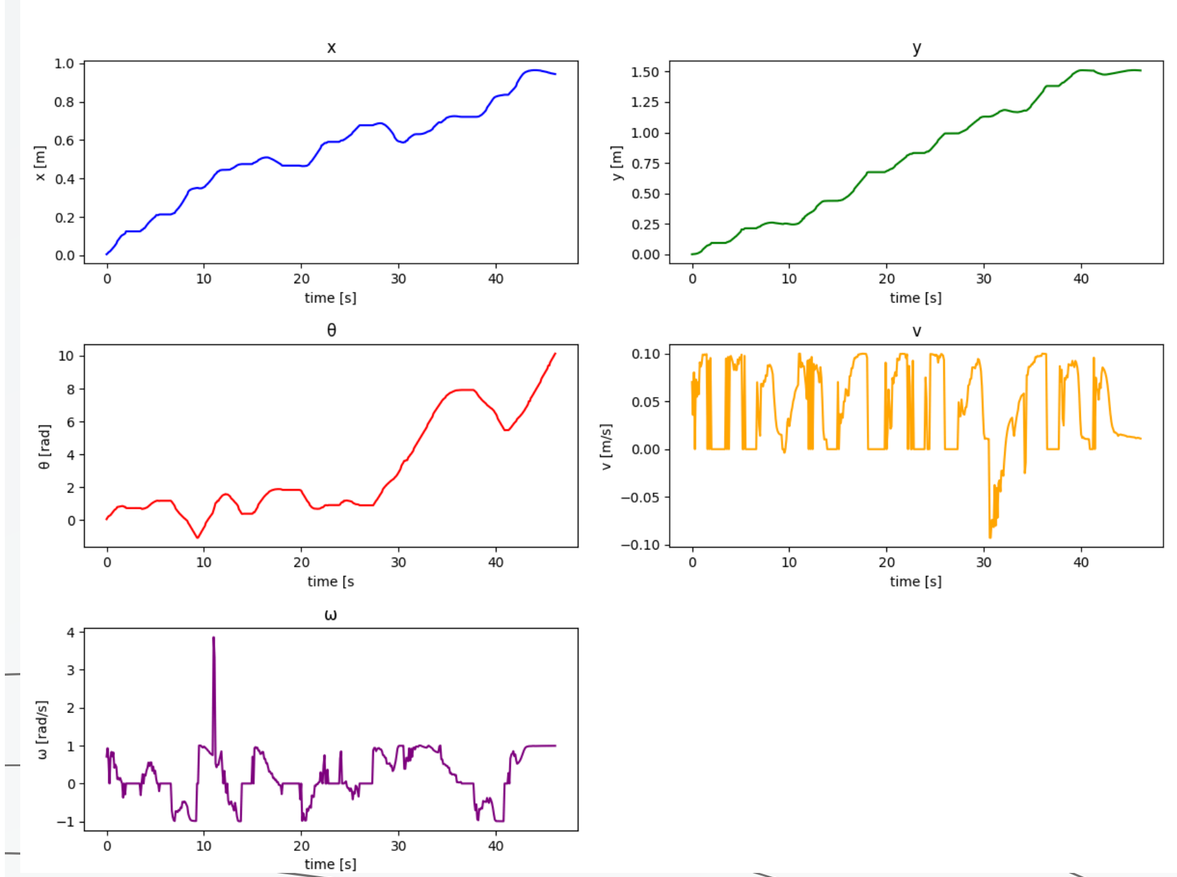
TEST

TRAJECTORY PLANNING - TRAPEZOIDAL PROFILE



TRAJECTORY PLANNING - TRAPEZOIDAL PROFILE

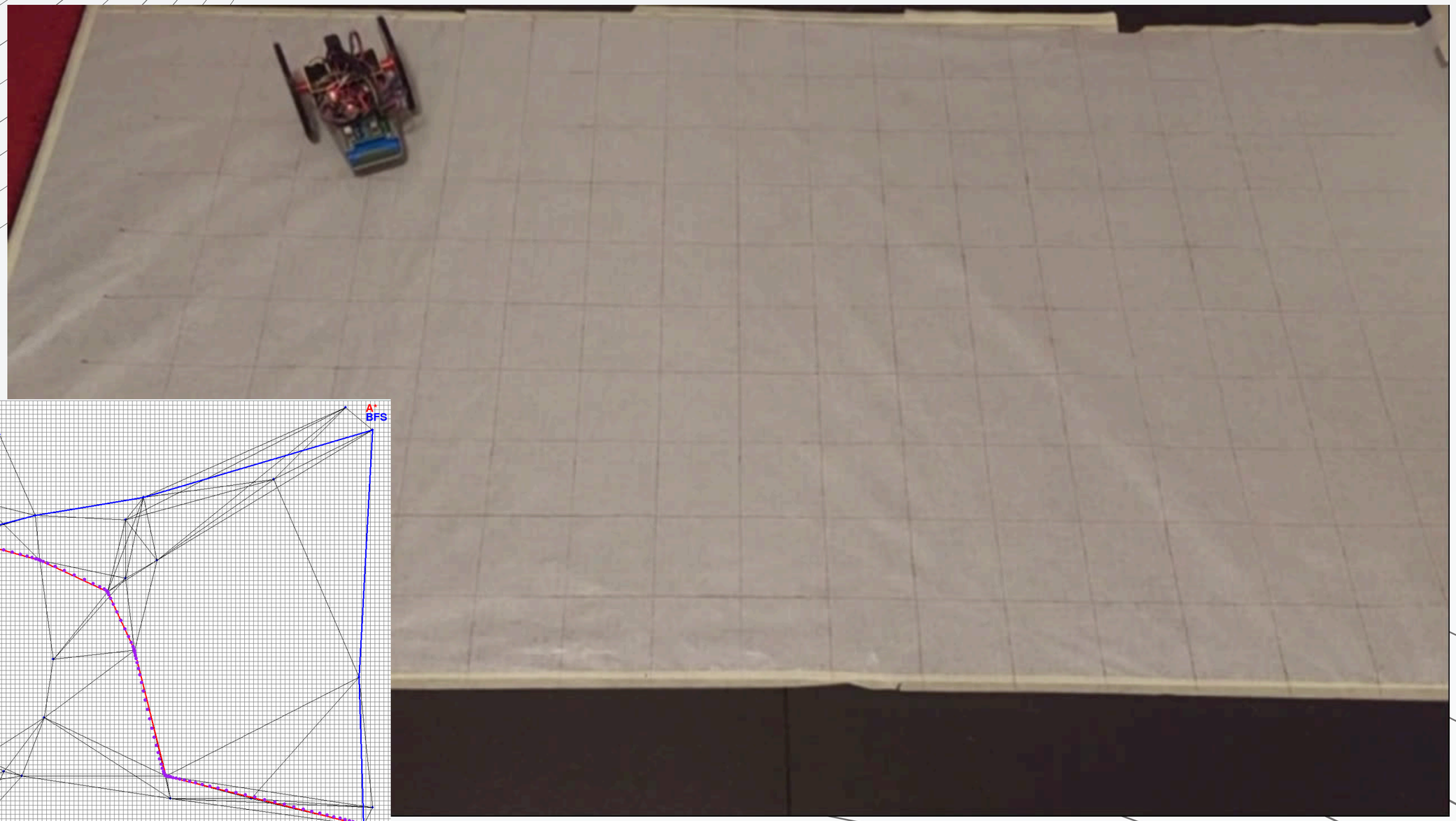




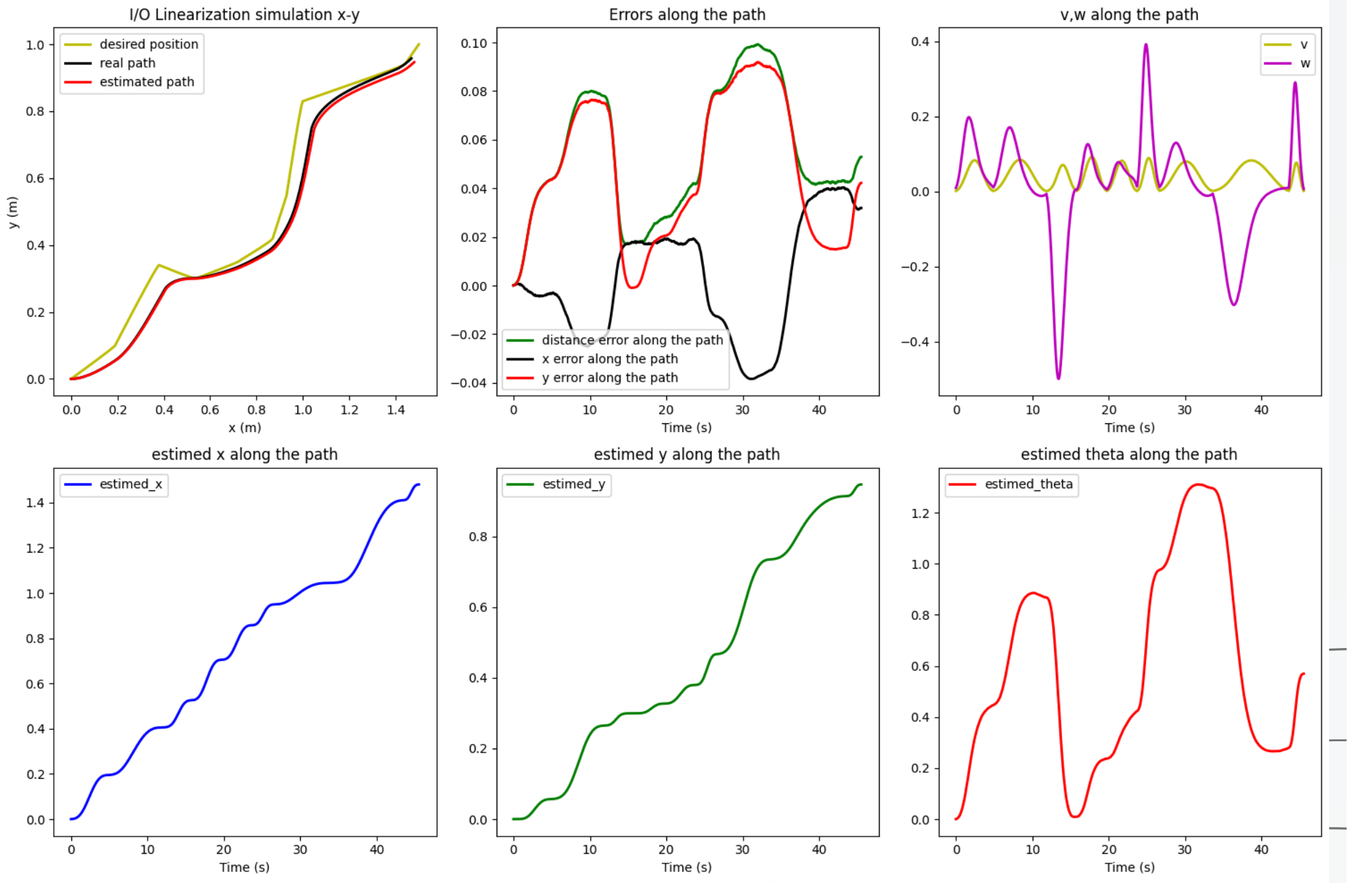
SIMULATION

TEST

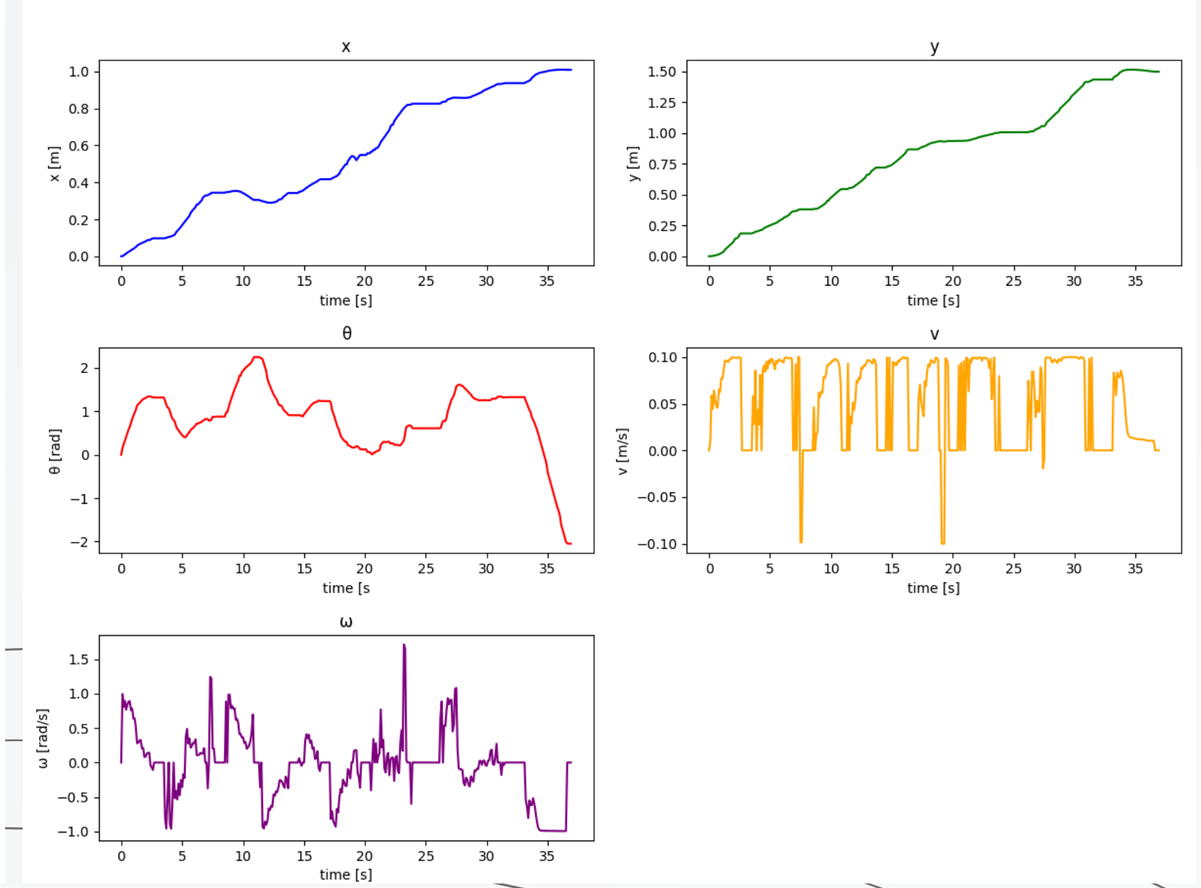
TRAJECTORY PLANNING - CUBIC PROFILE



TRAJECTORY PLANNING - CUBIC PROFILE

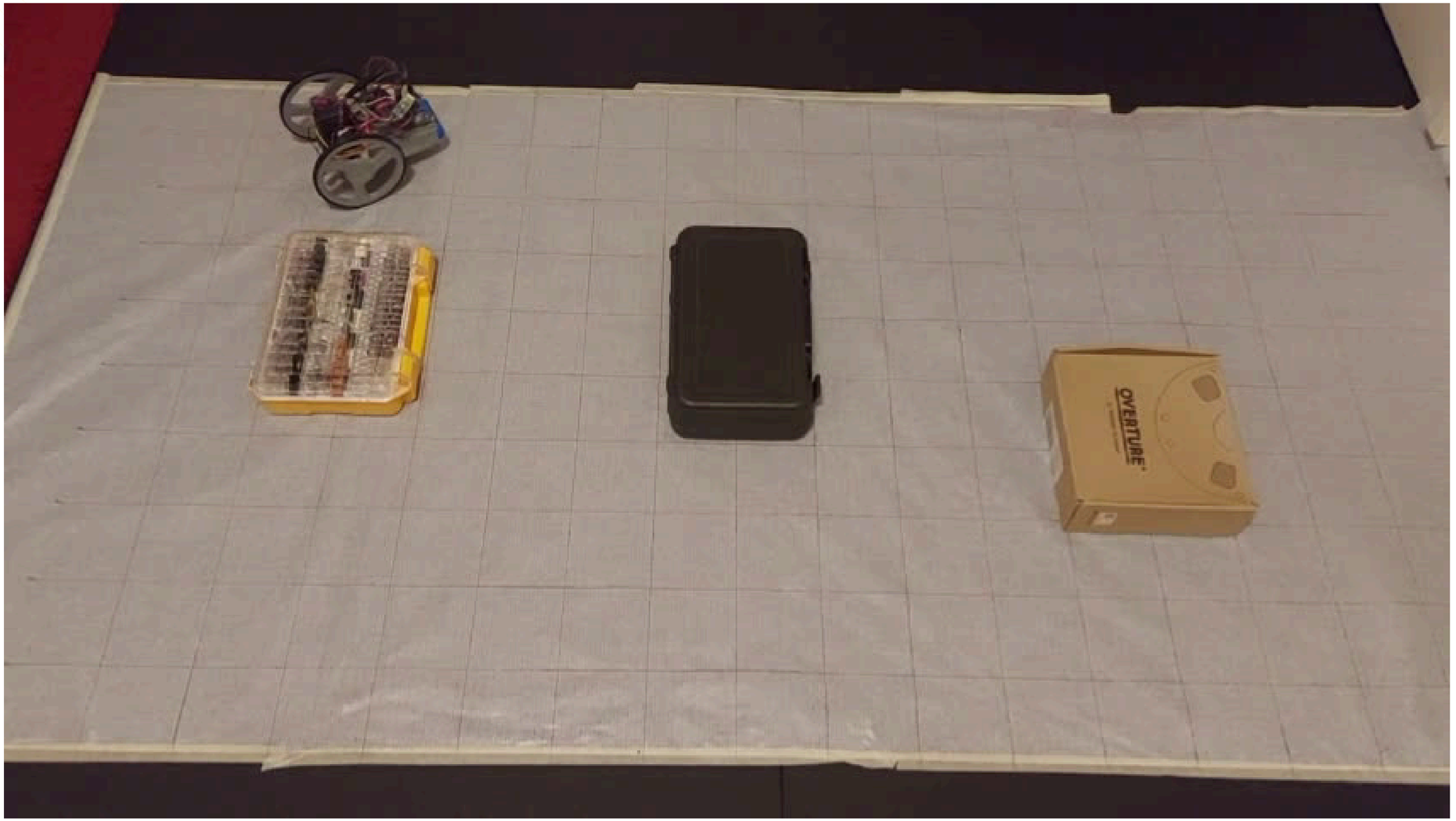


SIMULATION

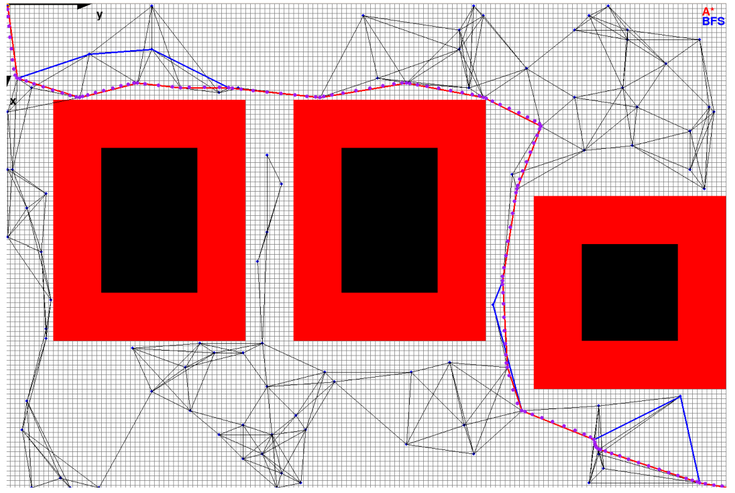


TEST

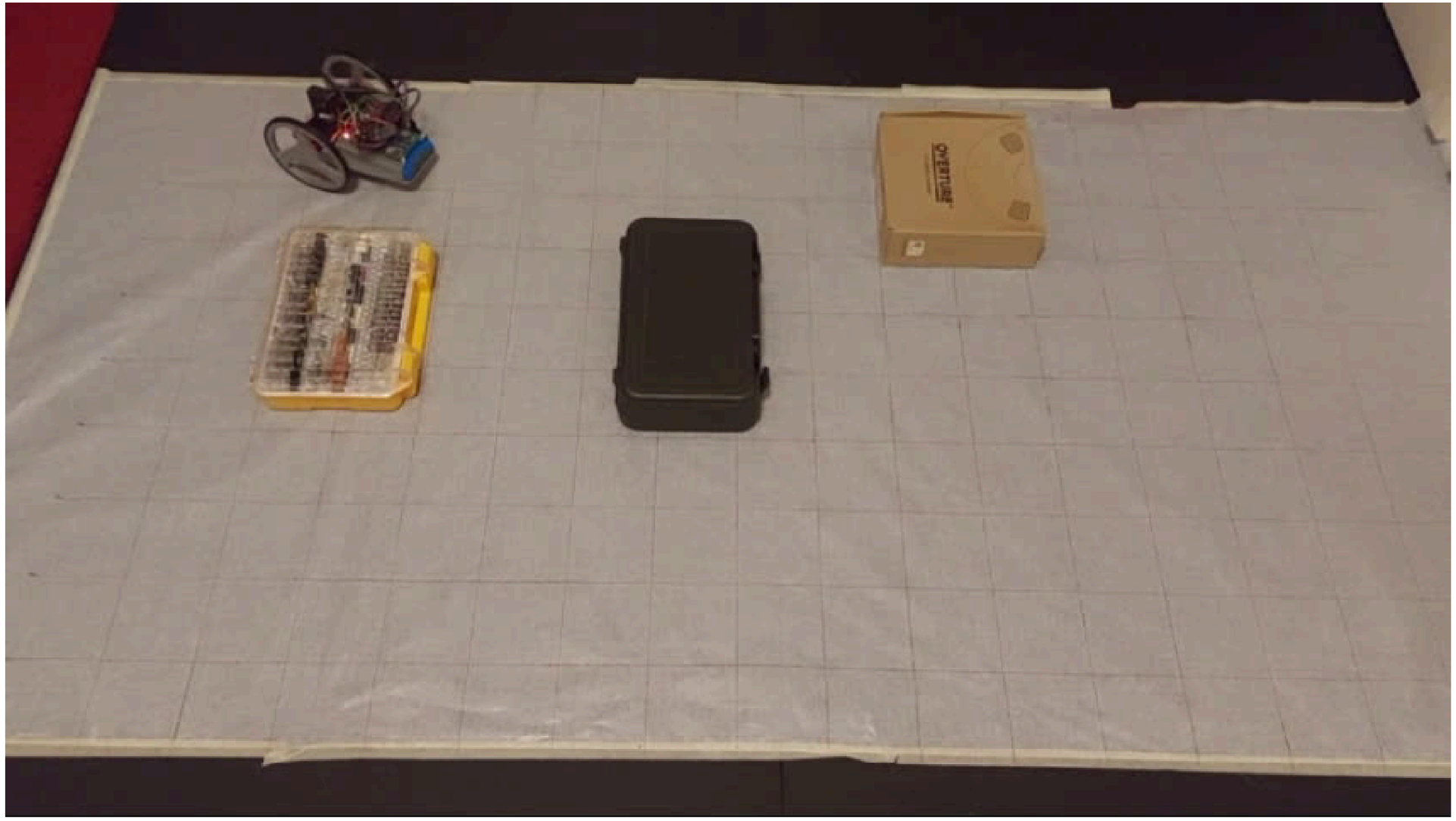
OBSTACLE AVOIDANCE



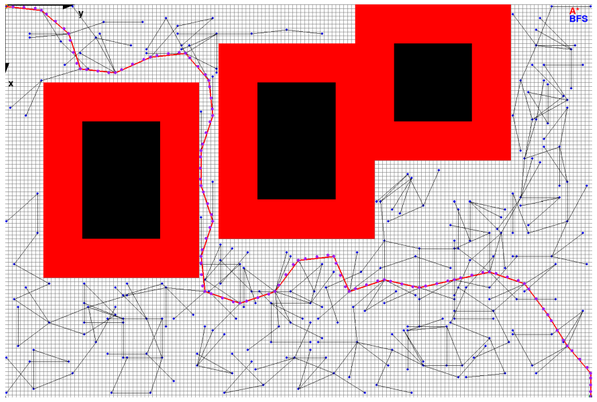
- Nodes: PRM, RRT
- Trajectory: Cartesian polynomial with linear profile



OBSTACLE AVOIDANCE



- Narrow corridor: Difficult to generate feasible trajectories with PRM



FUTURE DEVELOPMENTS

- UKF onboard with camera or LIDAR
- Spline
- Car body
- Add features GCS (parameters Trajectory planning, live visualization)
- DC motors with incremental encoders