The Latest Result of CAN Self-Driving Car Project

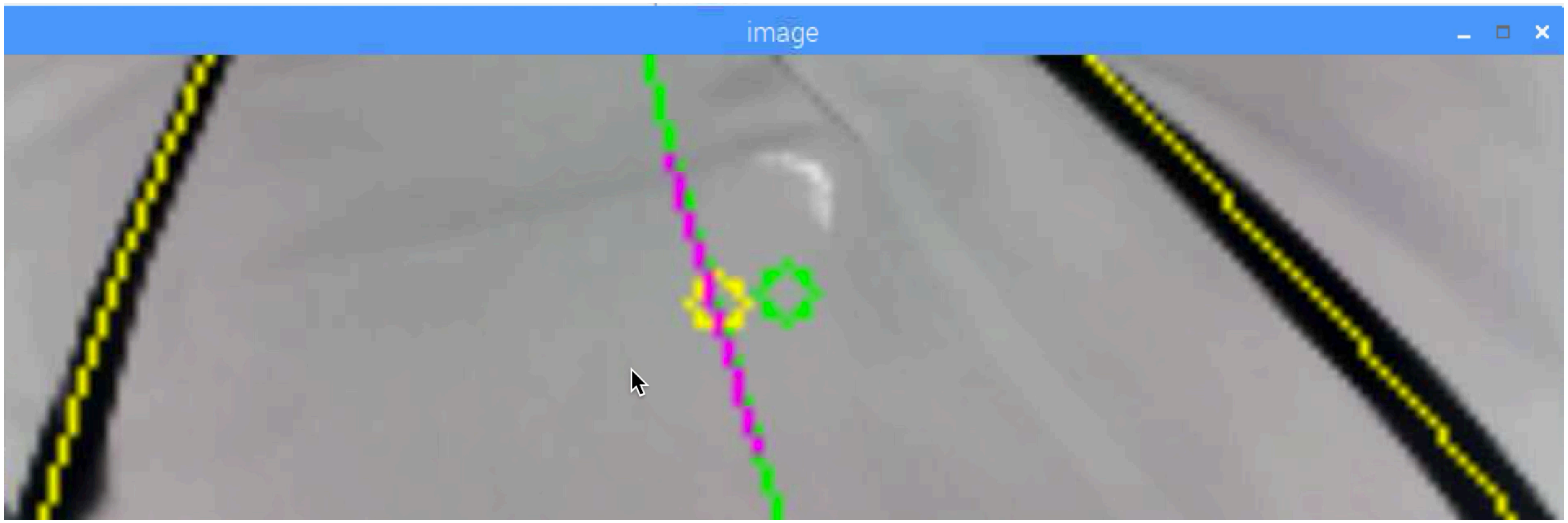
Mingyu Zhao
Control **A**nd **N**etwork Lab
Graduate Assistant
11/9/2018

Overview

- Project's Objective
- Processing flow
- Comparison
- Learning Process

I Objective

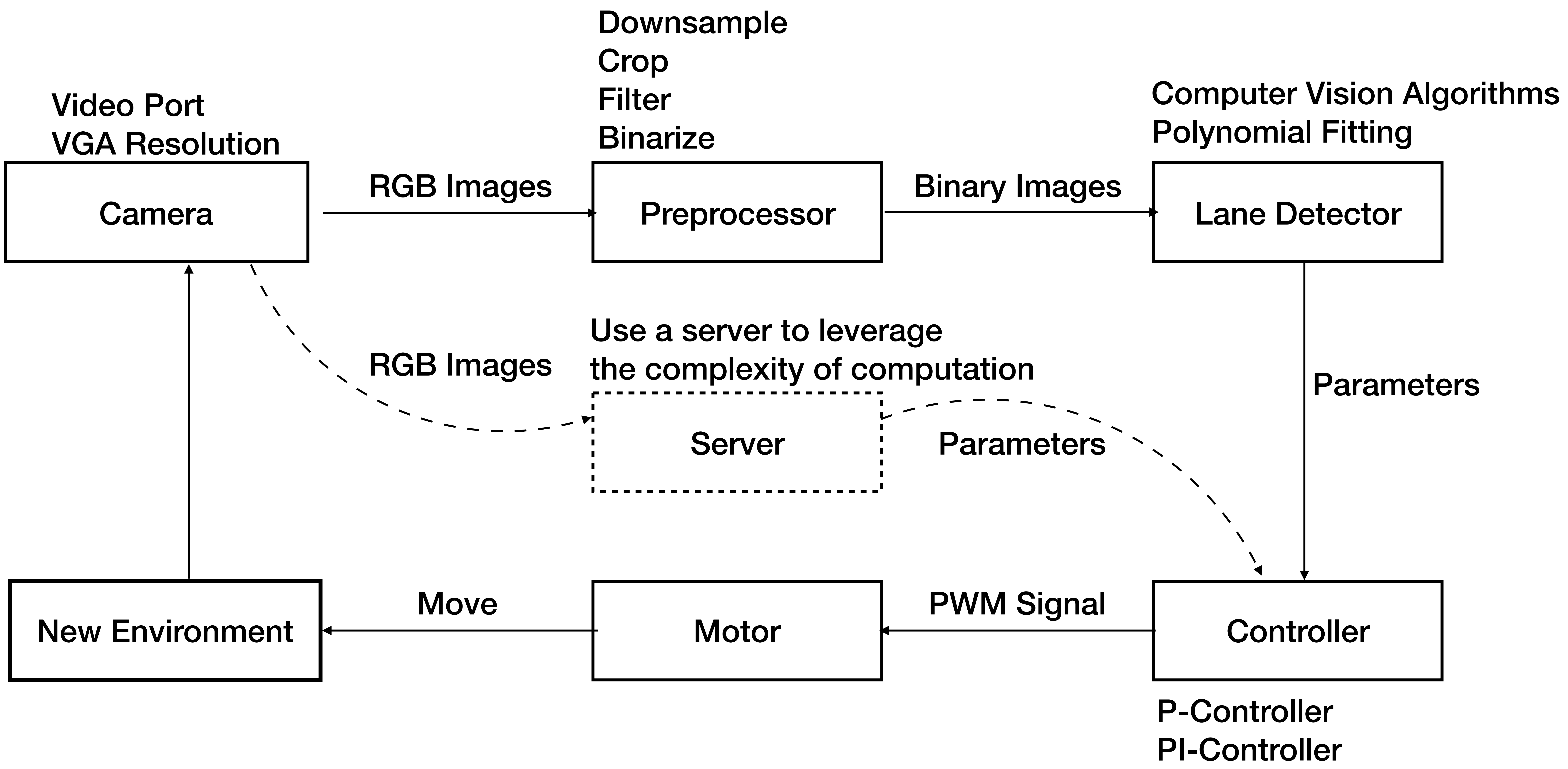
- Keep stay in the center of the lane.
- Minimize the distance between the guide line and its own position.



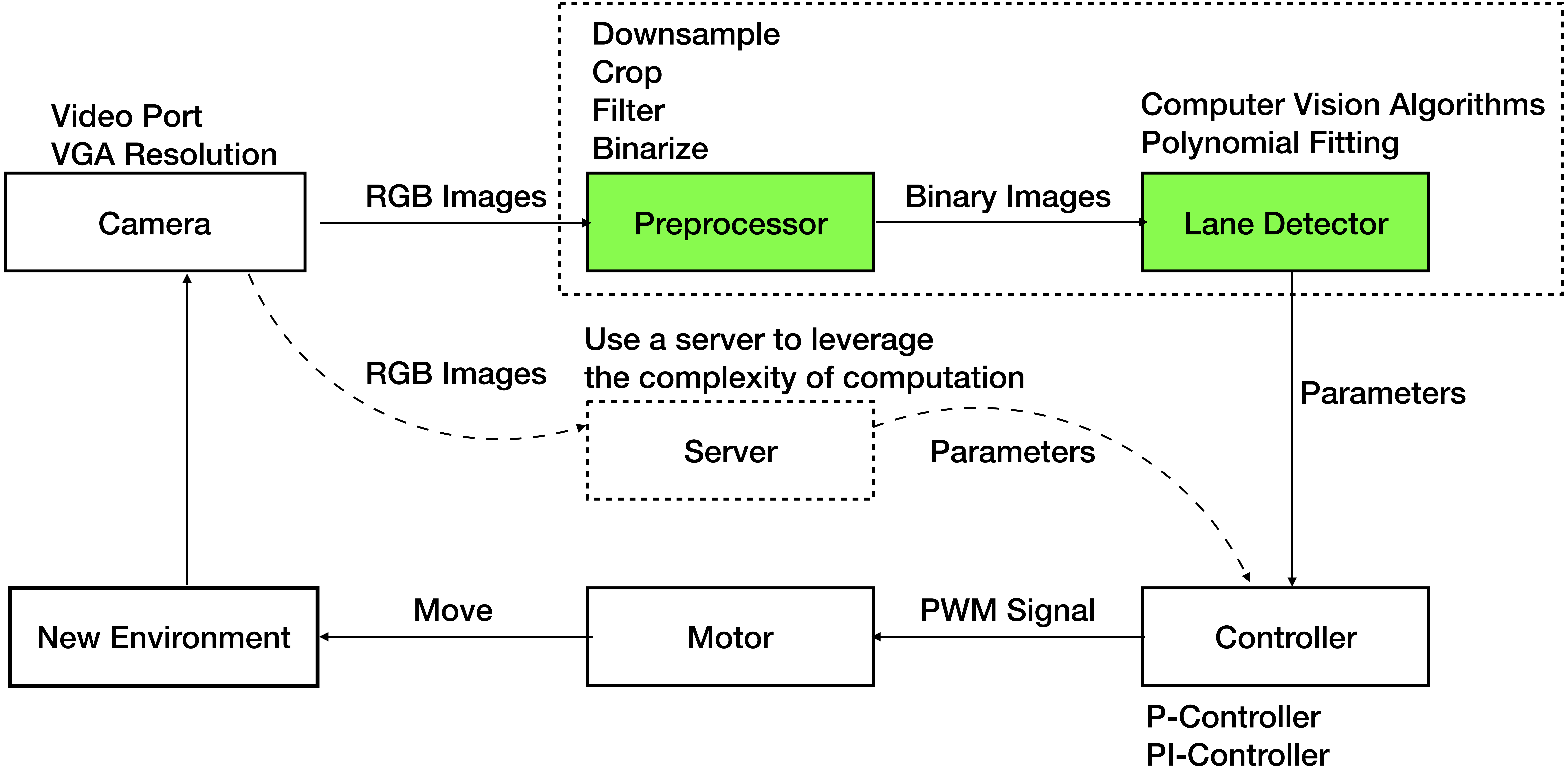
I Objective

- Environment:
  - Circular lane
  - Start with the same position and angle
    - D = 4.716 cm
    - *Theta = 0.4*
  - Image Processing Speed: 8~10 FPS
  - Offline mode (Run the code directly on the car)

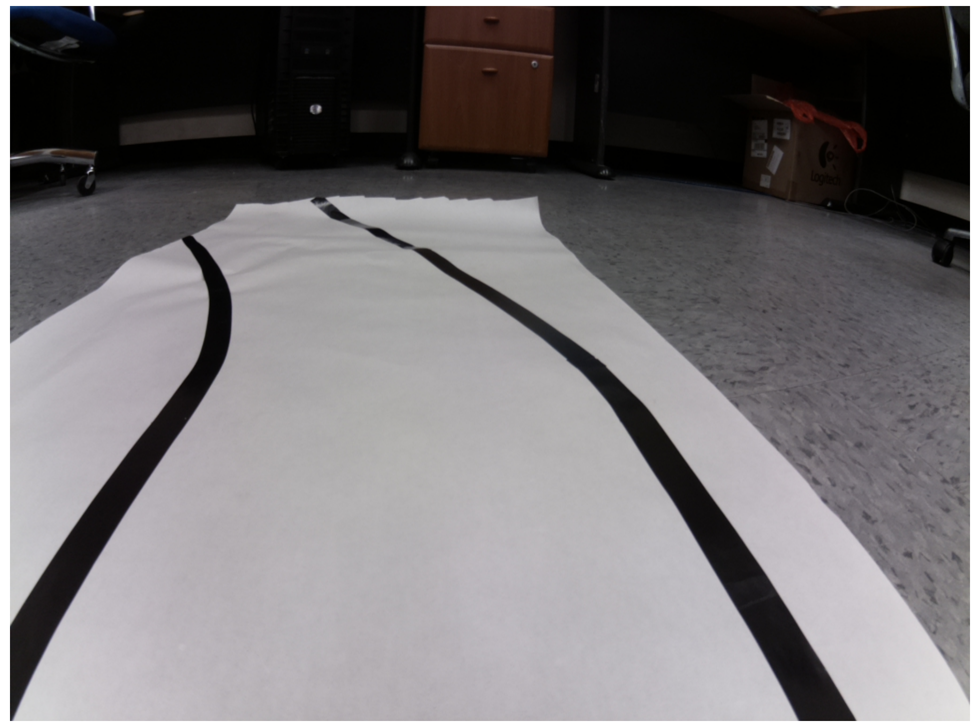
II Processing Flow



II Processing Flow



II Processing Flow



Original Image:
height:480
weight: 640
channel: RGB

Downsample: reduce the computation complexity
Crop: crop the part of lane we care about
Color Filter: only keep the black pixels
Binarize: transform into 0s and 1s

Preprocessed Image:
height: 48
width: 160
channel: binary

Camera

Downsample
Crop
Filter
Binarize

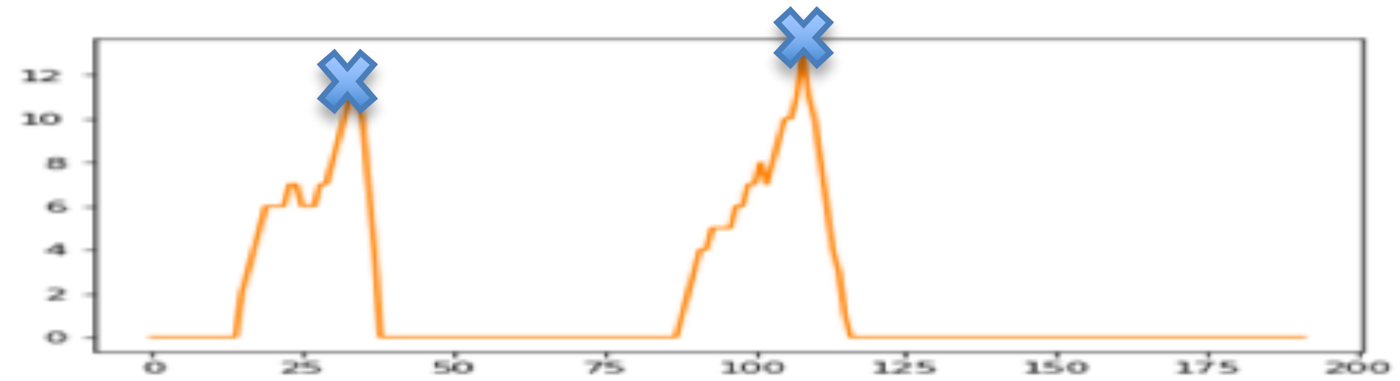
Preprocessor

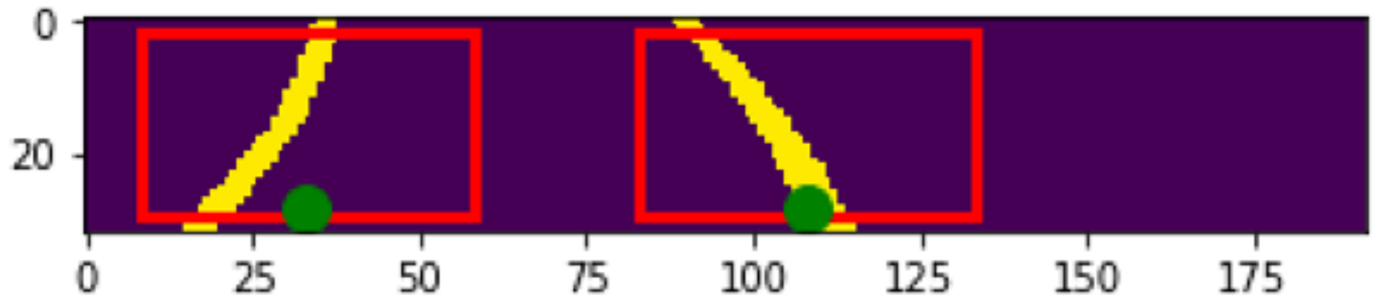
Polynomial Fitting

Lane Detector

II Processing Flow

	5	86	87	88	89	90	91	92	93	94	95
0		0	0	1	1	1	1	0	0	0	0
1		0	0	0	1	1	1	1	0	0	0
2		0	0	0	0	1	1	1	1	0	0
3		0	0	0	0	0	1	1	1	1	0





Preprocessed Image:
height: 48
width: 160
channel: binary

Sum up over the height's axis

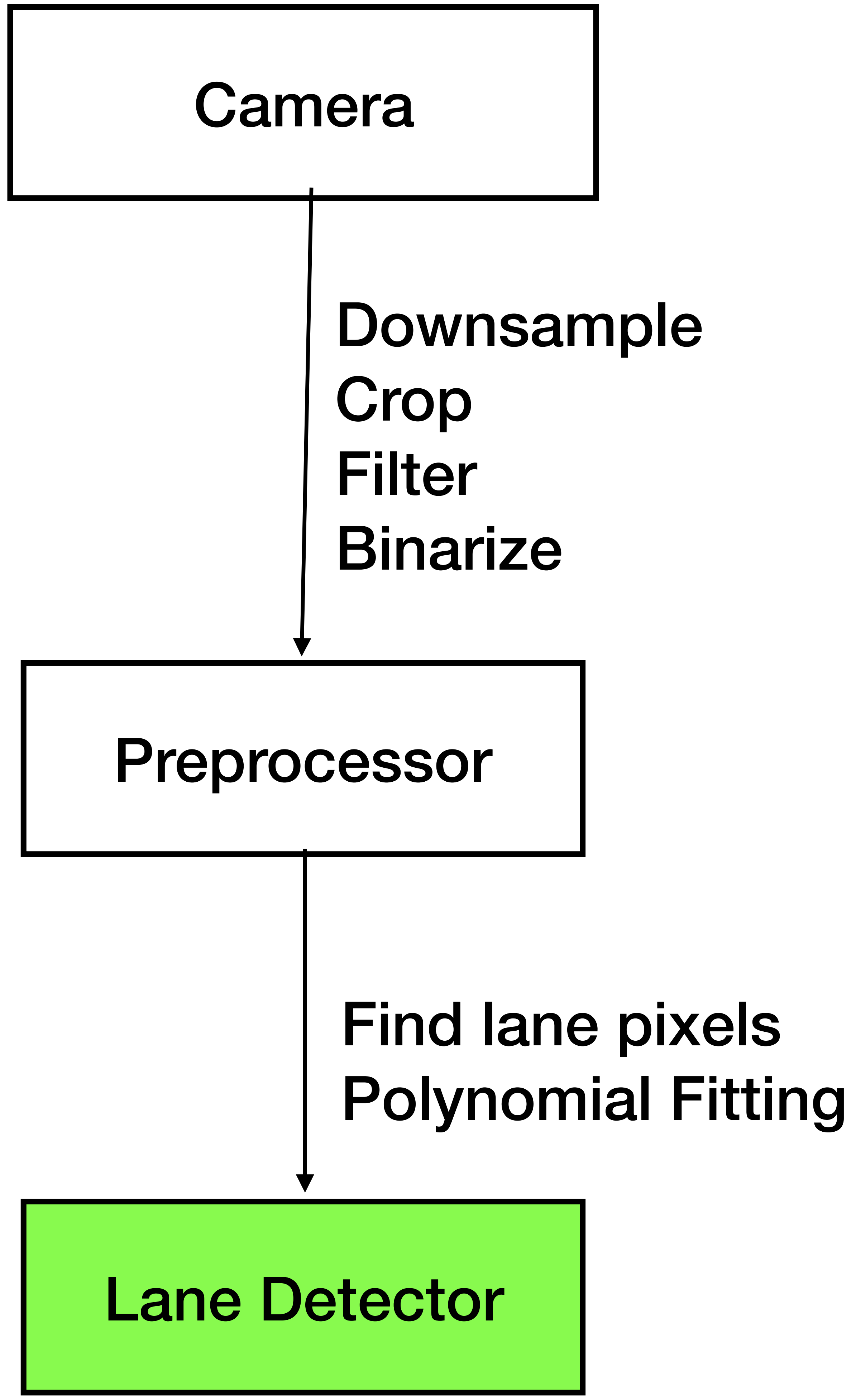
Indicators of a Lane

Detect pixels within windows

Lane Pixels around the Indicators

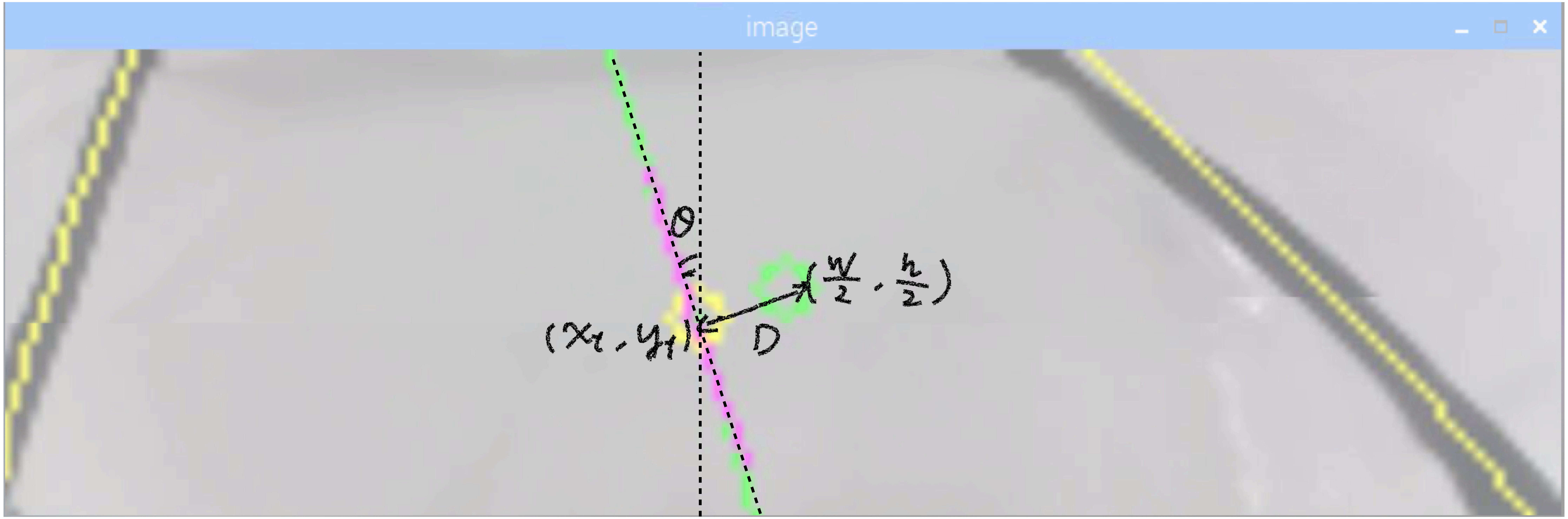
Polynomial Fitting

Fitted Parameters:
polynomial curve (order 2)
represented with 3 parameters: *w_0, w_1, w_2*



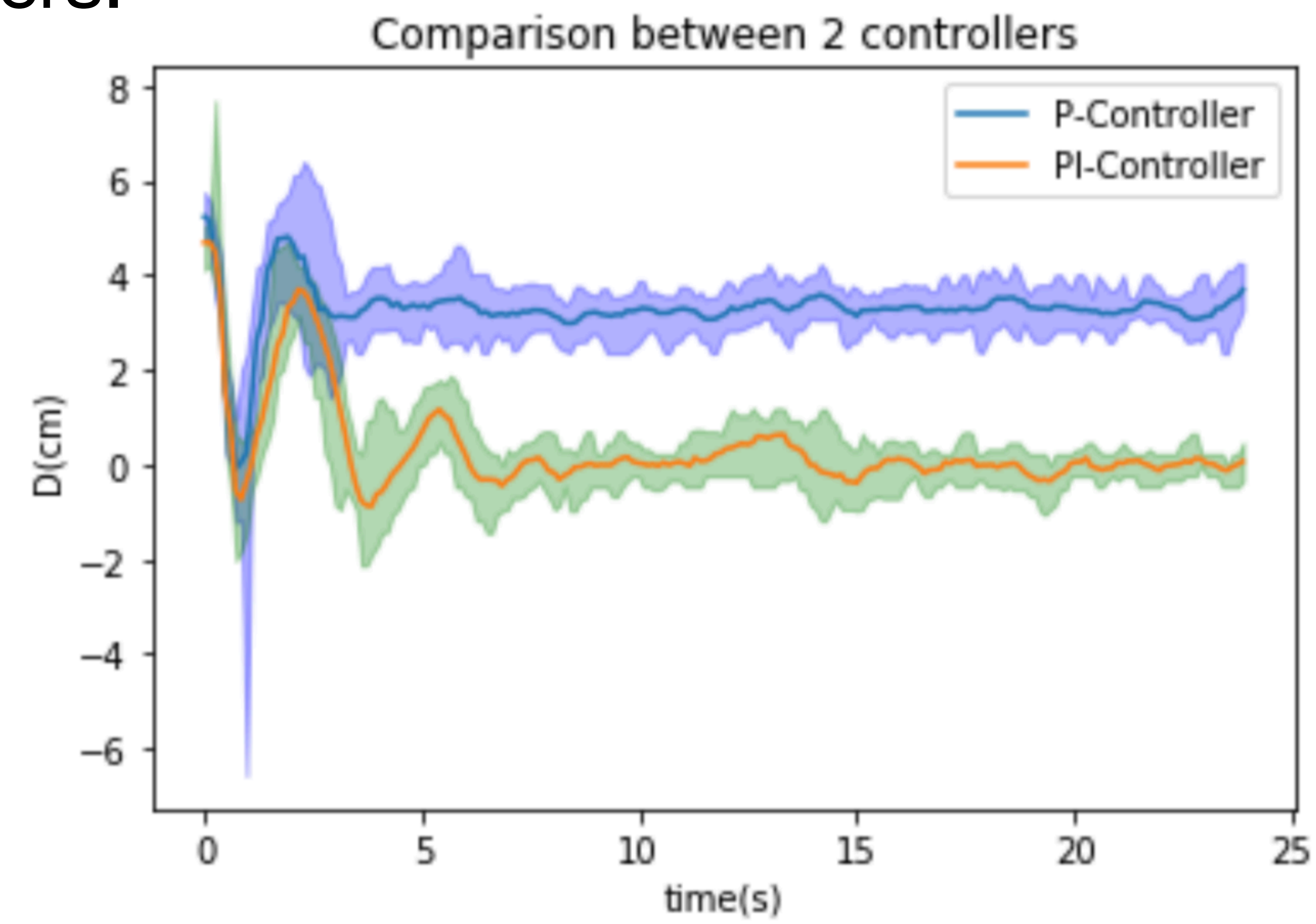
II Processing Flow

- *D*: Distance to the **centerline**. $min_{x_t, y_t}((\frac{w}{2} - x_t)^2 + (\frac{h}{2} - y_t)^2)$
- *Theta*: Angle between the tangent and the vertical line. $\theta = \frac{\partial x}{\partial y}(y_t) = atan(w_1 + 2w_2 y_t)$



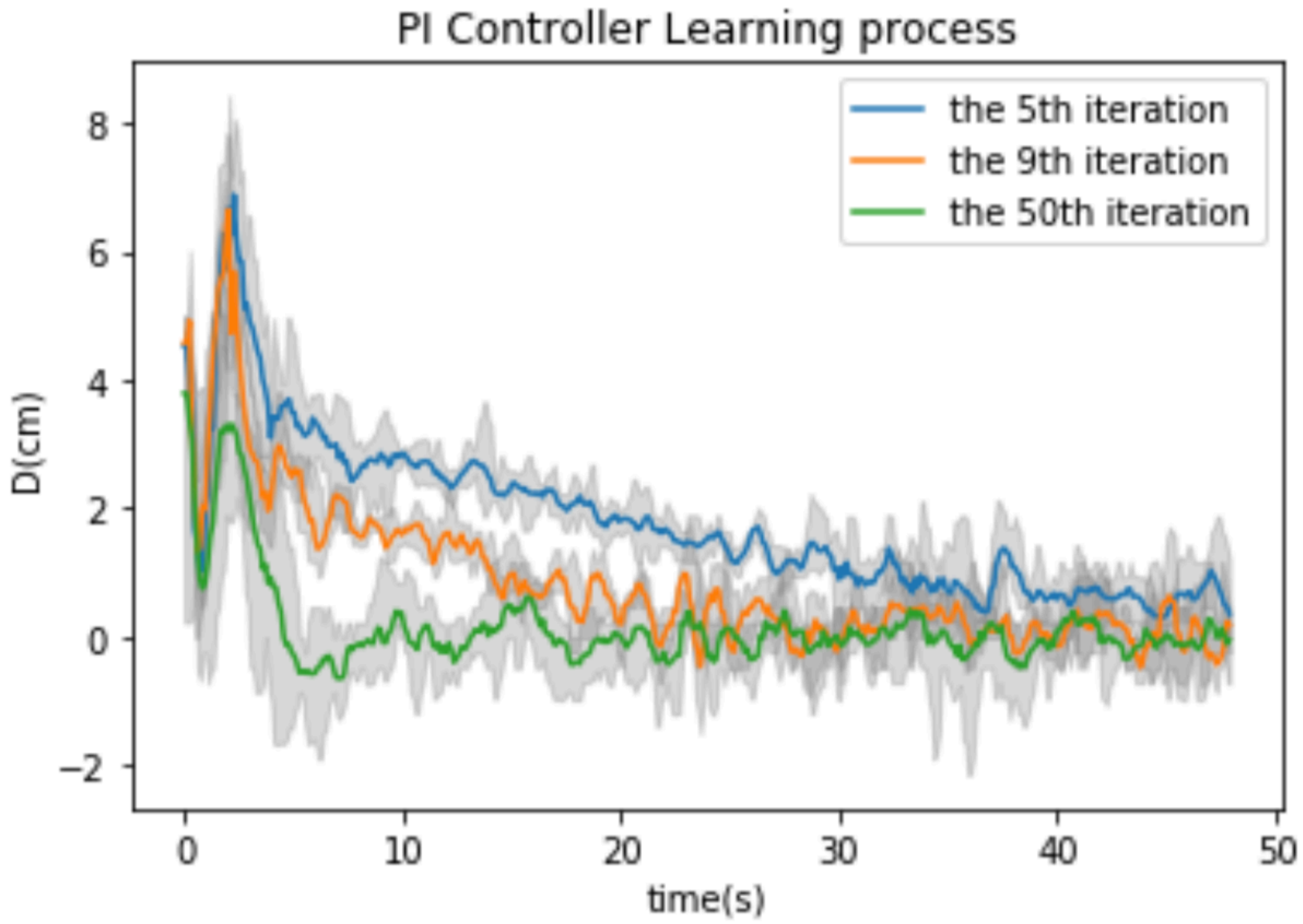
III Comparison

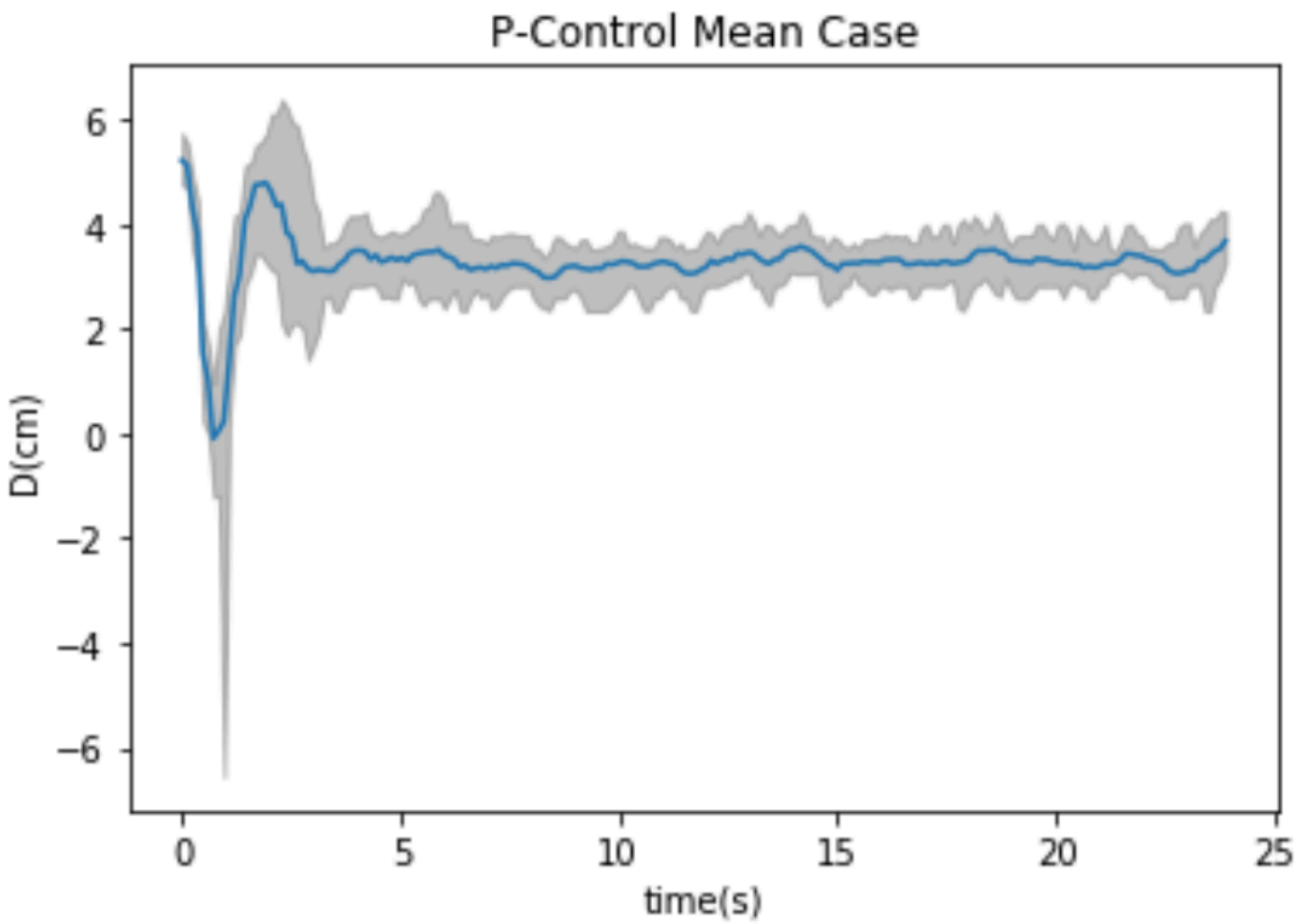
- Comparison between different controllers:
- Manually tuned **P-Controller**
  - Only used *D* to control.
  - Manually tuned *K.*
  - Collected data for training.
- **PI-Controller** trained from ADP
  - Control parameters: *D, Theta, Integral*
  - Learned *K_1, K_2, K_3*



IV Learning Process

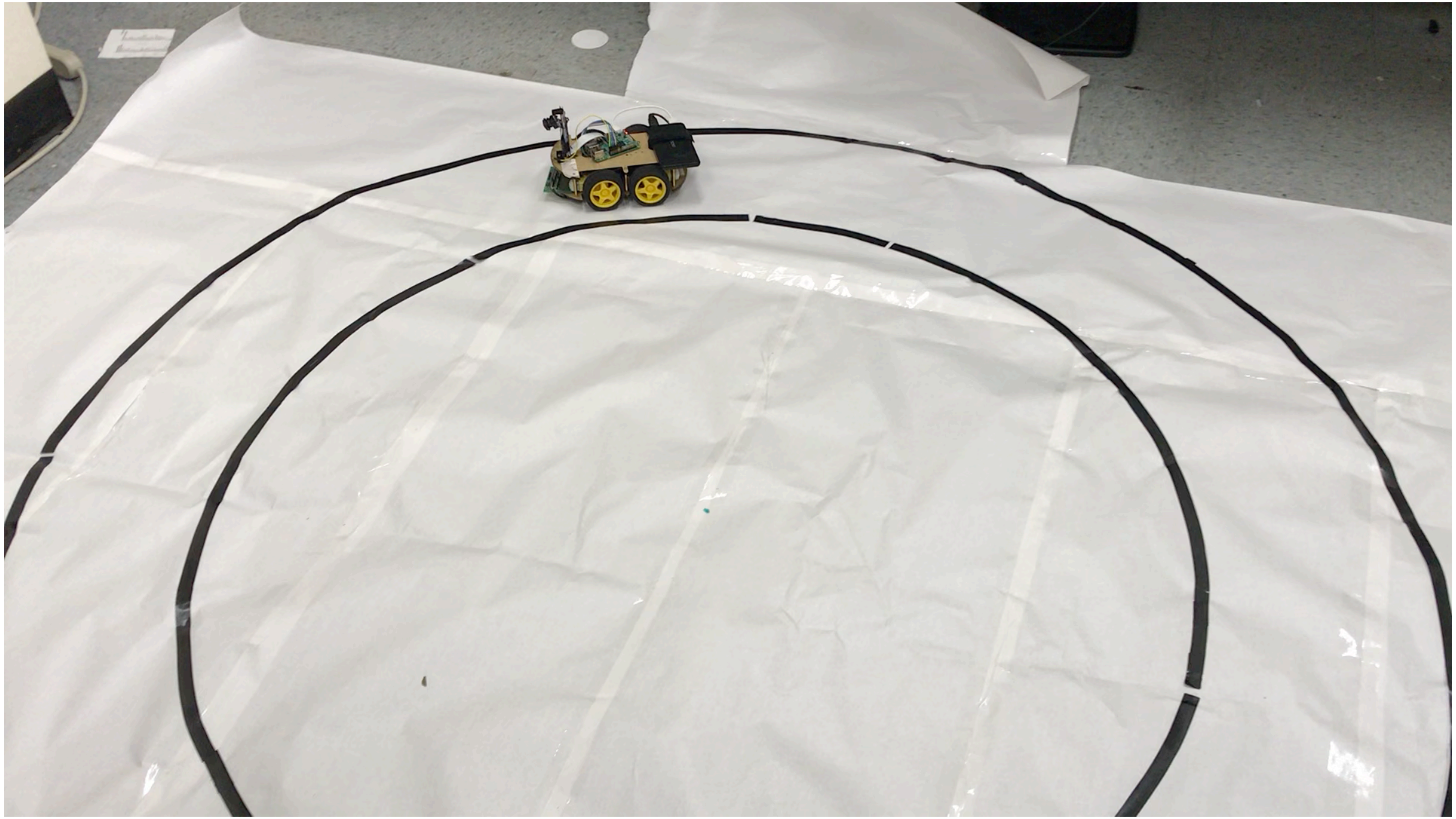
- Choose the controller learned from the 5th, 9th, 59th iteration
- The converge speed varies.

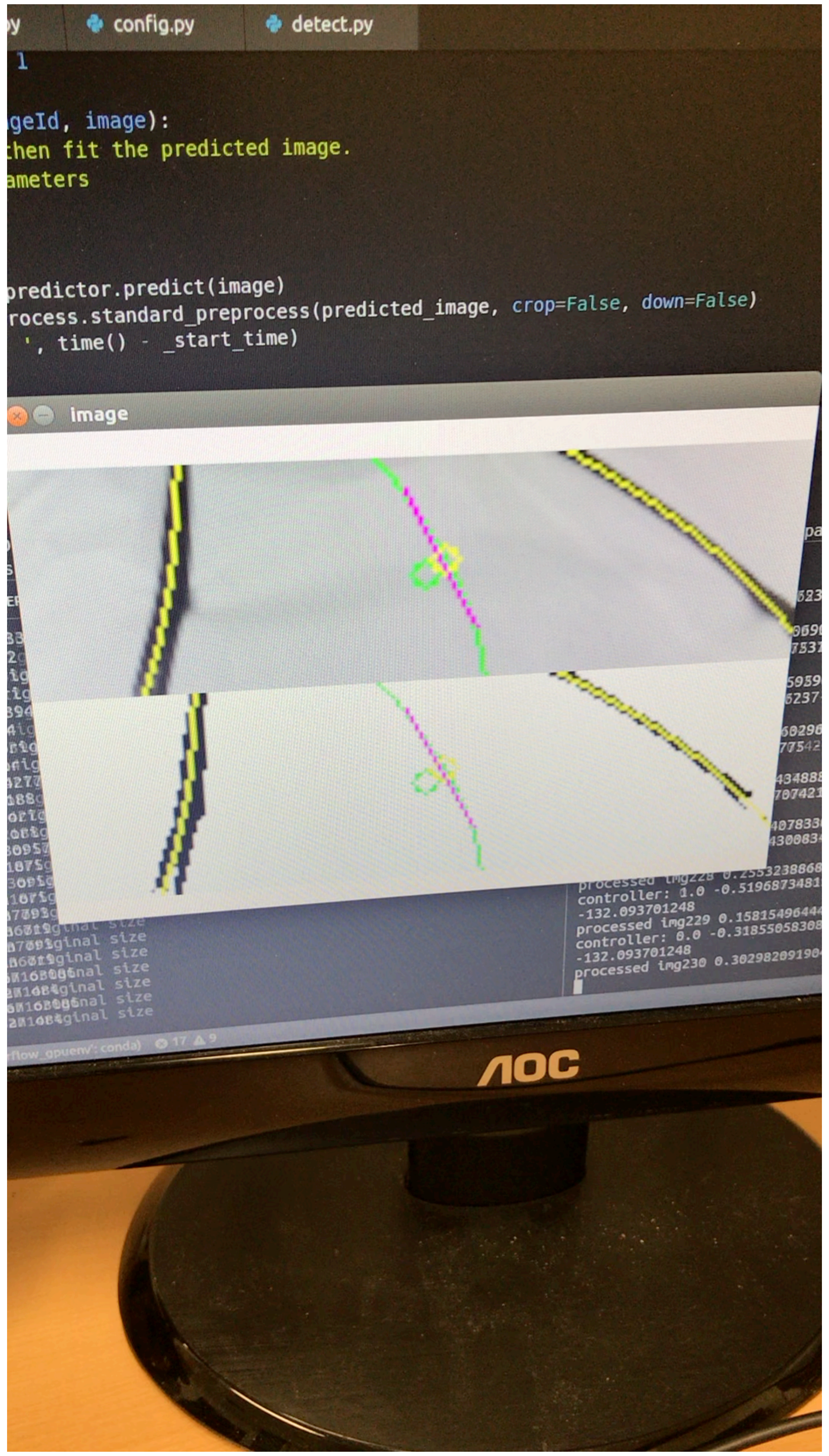




Thank You

Ex. Videos

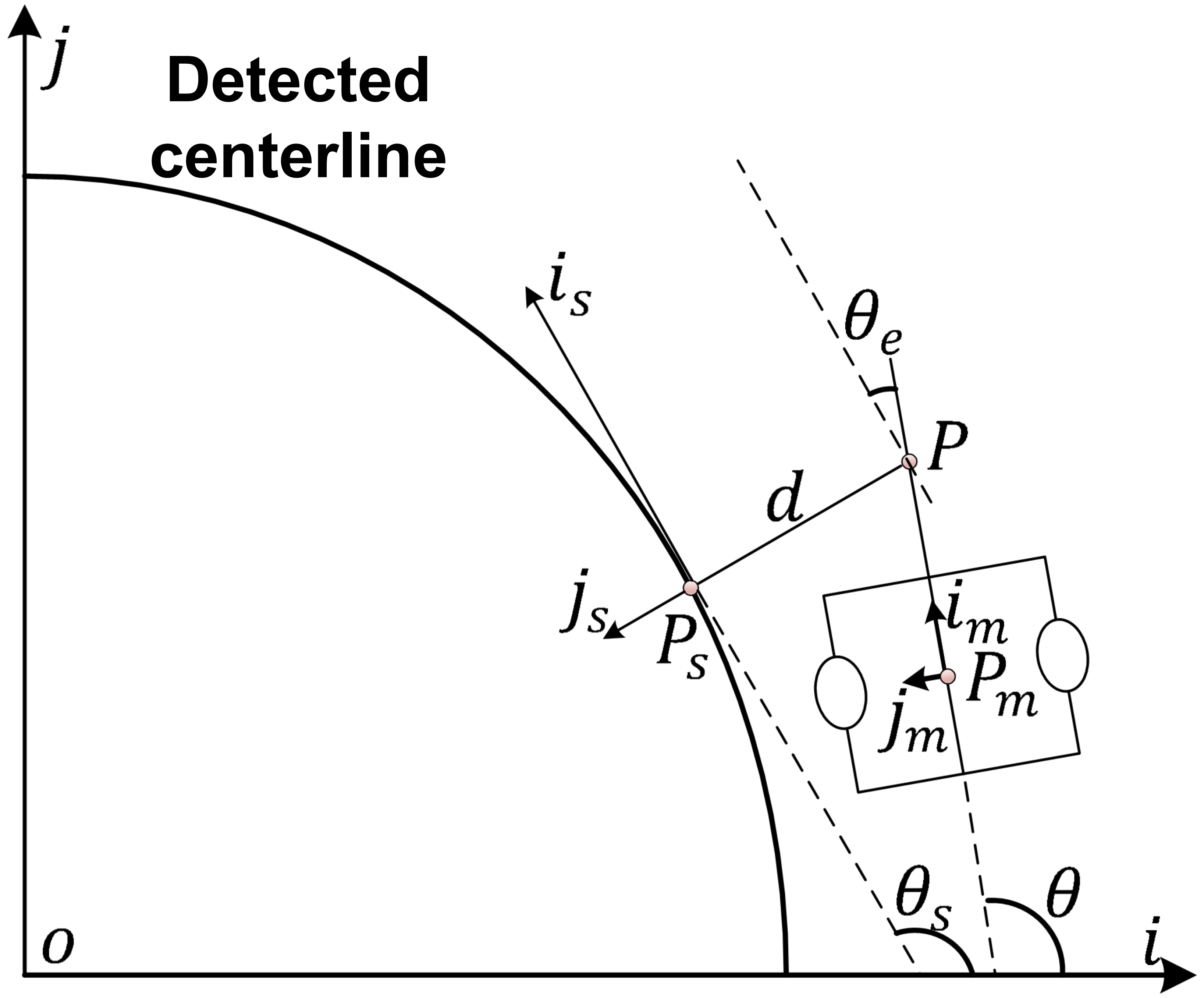



config.py
detect.py
geId, image):
then fit the predicted image.
ameters
predictor.predict(image)
rocess.standard_preprocess(predicted_image, crop=False, down=False)
', time() - _start_time)
image
AOC

Ex. Improvement

- Improvement for online mode
  - Ad-hoc network
  - Avoid bad channel
  - Powerful Server

We consider a unicycle-type model for our mobile robot:



Kinematic Model of Mobile Robot

$$\begin{aligned} \dot{X} &= v\cos\theta, \\ \dot{Y} &= v\sin\theta, \\ \dot{\theta} &= \omega, \end{aligned} \tag{1}$$

This is not direct applicable to our problem, since X, Y, θ are not measurable.

We will have to design controller based local information, i.e., the image of the monocular camera.

Specifically, we will control the robot orientation by controller the motors through different pulse width modulation (PWM) input.

The linear velocity v and the angular velocity ω are regulated by the actuated wheels via the following relations

$$v = \frac{r}{2}(\omega_r + \omega_l), \tag{2}$$

$$\omega = \frac{r}{2L}(\omega_r - \omega_l), \tag{3}$$

where r is the wheels' radius, L is the distance between wheels of both sides, ω_l and ω_r are the angular velocities of the left and right wheels.

In this study, we keep v constant.

For simplicity, suppose that the relationship between the pulse width modulation (**PWM**) input to the motors that the drive the robot and the **angular velocity of wheels** can be described by

$$\omega_i = bm_i, \tag{4}$$

where $i = \{l, r\}$, m_i is the pulse width modulation (PWM) input to the micro-controller that regulates the motor speed, and b is the coefficient.

Define the PWM difference between both sides $u = m_r - m_l$ as the control input.

The resultant kinematic model is

$$\begin{aligned}\dot{X} &= v\cos\theta,\\ \dot{Y} &= v\sin\theta,\\ \dot{\theta} &= b_m u.\end{aligned} \tag{5}$$

where $b_m = \dfrac{rb}{2L}$.

We introduce the following three variables to characterize the robot motion with respect to the curve:

1. s is curvilinear coordinate of P_s;
2. d is the distance between P and the curve, i.e., the ordinate of the point P in the frame $\mathcal{F}_s$;
3. $\theta_e = \theta - \theta_s$ is the orientation angle error of the robot with respect to the road.

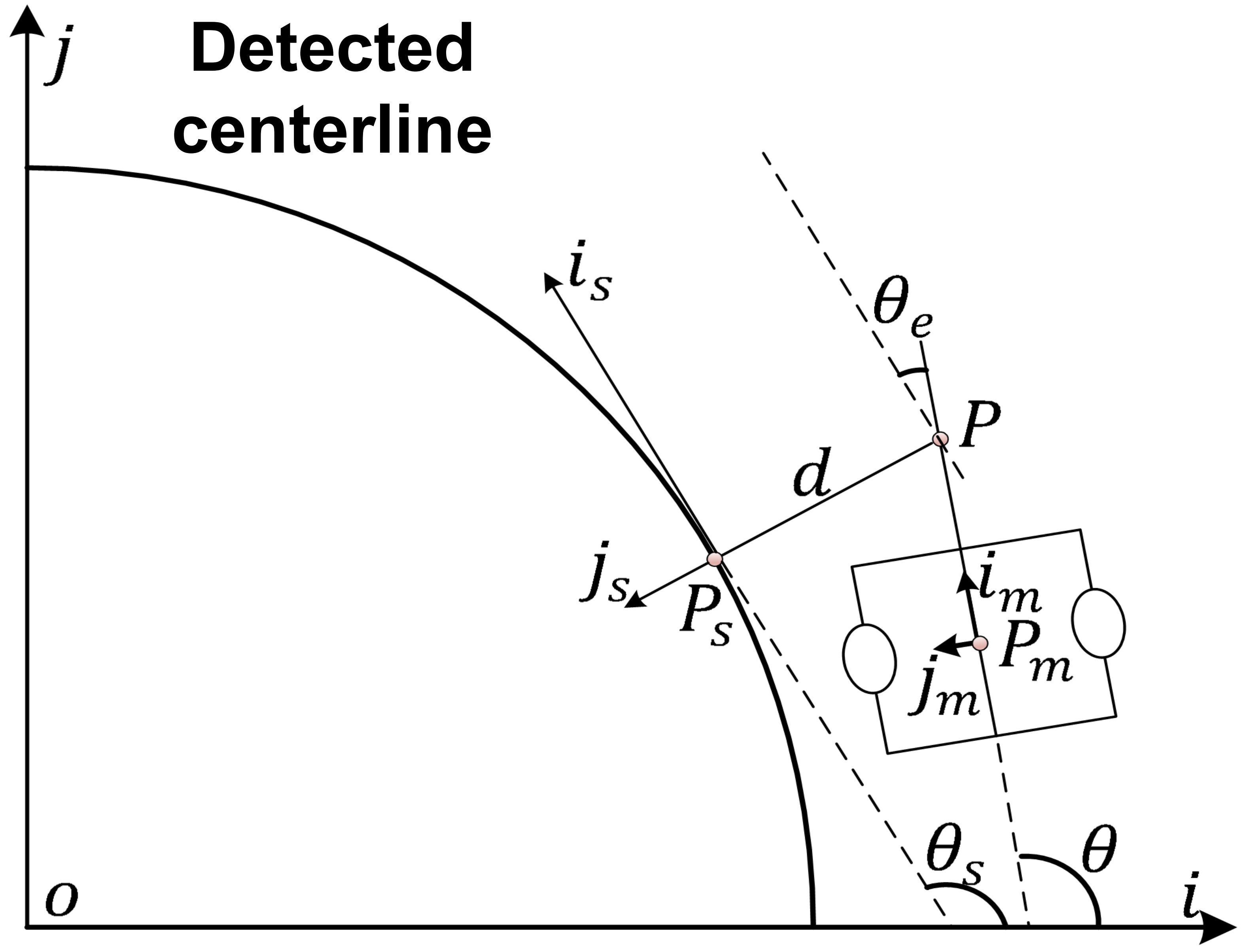
The model of robot motion is described by

$$\dot{s} = \frac{1}{1 - dc}\left[v - l_1 b_m u \sin(\theta_e)\right], \tag{2}$$

$$\dot{d} = v \sin(\theta_e) + l_1 b_m u \cos(\theta_e), \tag{3}$$

$$\dot{\theta}_e = b_m u - \dot{s}c \tag{4}$$

where c is the constant curvature.



Kinematic Model of Mobile Robot

[1] A. Eskandarian, Handbook of intelligent vehicles. London, U.K.: Springer, 2012.