Self-Driving Cars

Exercise 3 - Modular Pipeline

Michael Oechsle

Autonomous Vision Group
MPI-IS / University of Tübingen

December 21, 2018

University of Tübingen
MPI for Intelligent Systems
Autonomous Vision Group

Exercise Setup

Download a new version of the gym from

`https://owncloud.tuebingen.mpg.de/index.php/s/DaZJ4xmsPToZdzd`

Download `exercise_03_modular_pipeline_exercise.zip` which contains:

- Exercise sheet & slides
- Code template

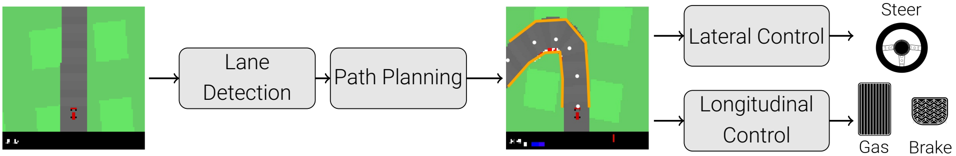
Submit `.zip` folder which contains:

- Your report of up to 5 pages (`.pdf`)
- Your all your parameters in (`modular_pipeline.py`)
- Your Python code (`.py`)

Deadline: **Wed, 30. January 2019 - 21:00**

Modular Pipeline

Modular Pipeline



3.1

Lane Detection

Lane Detection

Template

- `lane_detection.py`
- `test_lane_detection.py` for testing

a) **Edge Detection:**

- Translate the state image to a grey scale image and crop out the part above the car
  $\rightarrow$ `LaneDetection.cut_gray()`
- Derive the absolute value of the gradients of the grey scale image and apply thresholding to ignore unimportant gradients.
  $\rightarrow$ `LaneDetection.edge_detection()`
- Determine arguments of local maxima of absolute gradient per pixel row
  $\rightarrow$ `LaneDetection.find_maxima_gradient_rowwise()`
  Hint: use for example `scipy.signal.find_peaks`

Lane Detection

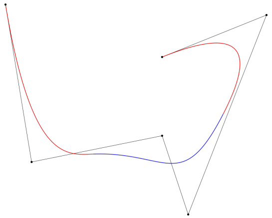
b) **Assign Edges to Lane Boundaries:**

- Find arguments of local maxima in the image row closest to the car
  $\rightarrow$ `LaneDetection.find_first_lane_point()` (already implemented)
- Assign the edges to the lane boundaries by successively searching for the nearest neighboring edge/maximum along each boundary
  $\rightarrow$ `LaneDetection.lane_detection()`

Lane Detection

c) **Spline Fitting:**

- Fit a parametric spline to each lane boundary
  $\rightarrow$ `LaneDetection.lane_detection()`
- Use `scipy.interpolate.splprep` for fitting and `scipy.interpolate.splev` for evaluation



Given a list of points, which represents a curve in 2-dimensional space parametrized by s, find a smooth approximating spline curve $g(s)$.

Lane Detection

d) **Testing:**

- Find a good choice of parameters for the gradient threshold and the spline smoothness.
- Determine failure cases
- Add picture to your report

3.2

Path Planning

Path Planning

Template

- `waypoint_prediction.py`
- `test_waypoint_prediction.py` for testing

a) **Road Center:**

- Use the lane boundary splines and derive lane boundary points for 6 equidistant spline parameter values
  $\rightarrow$ `waypoint_prediction()`
- Determine the center between lane boundary points with the same spline parameter
  $\rightarrow$ `waypoint_prediction()`

Path Planning

b) **Path Smoothing:**

- Improve the path by minimizing the following objective regarding the waypoints x given the center waypoints y

$$\underset{x_1,\ldots,x_N}{\operatorname{argmin}} \sum_i |y_i - x_i|^2 - \beta \sum_n \frac{(x_{n+1} - x_n) \cdot (x_n - x_{n-1})}{|x_{n+1} - x_n|\,|x_n - x_{n-1}|}. \tag{1}$$

- Explain the effect of the second term
- Implement second term

  $\rightarrow$ `curvature()`

c) **Target Speed Prediction:**

- Implement a function that outputs the target speed for the predicted path in the state image , using

$$v_{\text{target}}(x_1, ..., x_N) = (v_{\text{max}} - v_{\text{min}}) \exp\left[-K_v \cdot \left| N - 2 - \sum_n \frac{(x_{n+1} - x_n) \cdot (x_n - x_{n-1})}{|x_{n+1} - x_n|\,|x_n - x_{n-1}|} \right|\right] + v_{\text{min}},$$

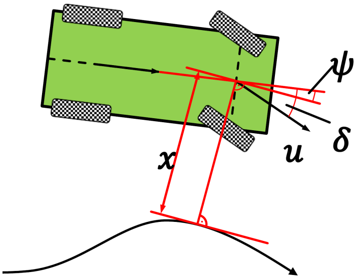
As initial parameters use: $v_{\text{max}} = 60,\quad v_{\text{min}} = 30$ and $K_v = 4.5.$

$\rightarrow$ `target_speed_prediction()`

3.3

Lateral Control

Lateral Control



$$\delta_{SC}(t) = \psi(t) + \arctan\left(\frac{k \cdot d(t)}{v(t)}\right) \tag{2}$$

where $\psi(t)$ is the orientation error, $v(t)$ is the vehicle speed, $d(t)$ is the cross track error and k the gain parameter.

Lateral Control

Template

- `lateral_control.py`
- `test_lateral_control.py` for testing

a) **Stanley Controller:**

- Read section 9.2 in the Stanley paper
  http://isl.ecst.csuchico.edu/DOCS/darpa2005/DARPA%202005%20Stanley.pdf
- Explain the parts of the heuristic control law

Lateral Control

b) **Stanley Controller:**

- Implement controller function given waypoints and speed
  $\rightarrow$ `LateralController.stanley()`
- Orientation error $\psi(t)$ is the angle between the first path segment to the car orientation
- Cross track error $d(t)$ is distance between desired waypoint at a spline parameter of zero to the position of the car
- Prevent division by zero by adding as small epsilon
- Describe the behavior of your car

Lateral Control

c) **Damping:**

- Damping the difference between the steering command and the steering wheel angle of the previous step

$$\delta(t) = \delta_{SC}(t) - D \cdot (\delta_{SC}(t) - \delta(t-1)) \,. \tag{3}$$

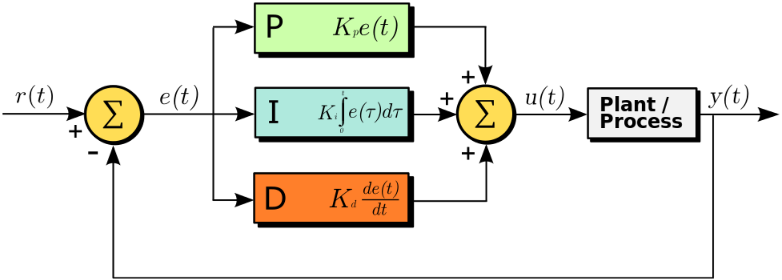
- Describe the behavior of your car

3.4

Longitudinal Control

Longitudinal Control

Proportional - Integral - Derivative Controller for gas and braking



Longitudinal Control

Template

- `longitudinal_control.py`
- `test_longitudinal_control.py` for testing

a) **PID Controller:**

- Implement a PID control step for gas and braking
- Use a discretized version:

$$e(t) = v_{\text{target}} - v(t)$$

$$u(t) = K_p \cdot e(t) + K_d \cdot [e(t) - e(t-1)] + K_i \cdot \left[\sum_{t_i=0}^{t} e(t_i) \right]$$

where $u(t)$ is the control signal and $e(t)$ error signal

Longitudinal Control

a) **PID Controller:**

- Due to integral windup, implement an upper bound for integral term.
- From control signal to gas and brake action values

$$a_{\text{gas}}(t) = \begin{cases} 0 & u(t) < 0 \\ u(t) & u(t) \geq 0 \end{cases} \qquad a_{\text{brake}}(t) = \begin{cases} 0 & u(t) \geq 0 \\ -u(t) & u(t) < 0 \end{cases}$$

Longitudinal Control

b) **Parameter Search:**

- Run `test_lateral_control.py` and have a look at plots of the target speed and speed to tune parameters (K_p, K_i, K_d)
- Start with $(K_p = 0.01, K_i = 0, K_d = 0)$
- Only modify a single term at a time !!!!

Competition

Competition

- For the competition you are free to modify and improve your basic modular pipeline and the parameters.
- Once you are happy, run `modular_pipeline.py score` using your parameters and submit your obtained evaluation score.

Questions?