Advanced Programming Practice

Autonomous Driving

-Lane detection-

2023 Fall

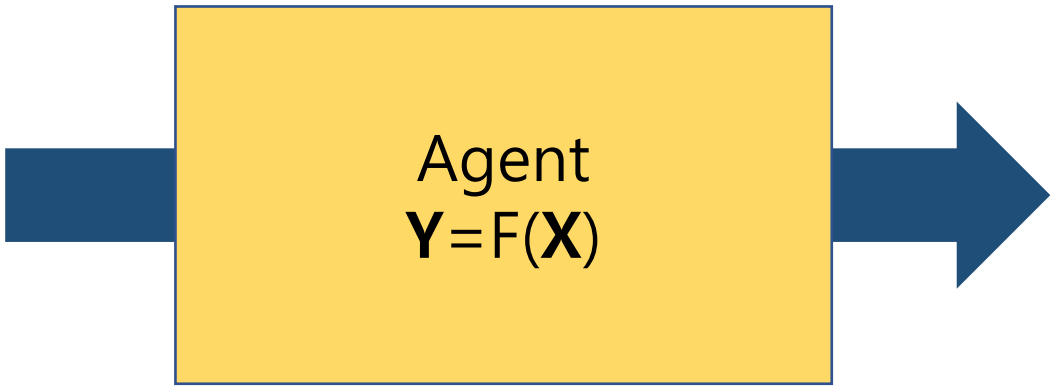
Sogang University

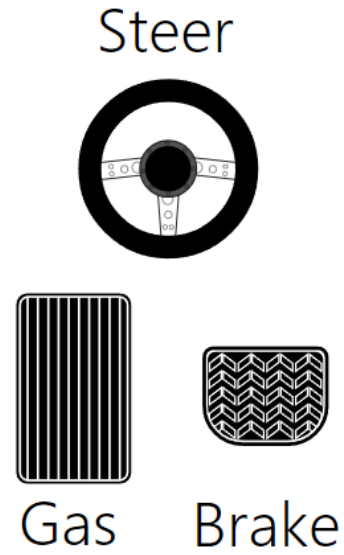
Credit to Dr. Andreas Geiger

Goal of Autonomous Driving

- Driving safely for a given scenario.
- 주어진 상황(state)에 따라 자동차를 안전히 조종하자.

Sensor Input **X**





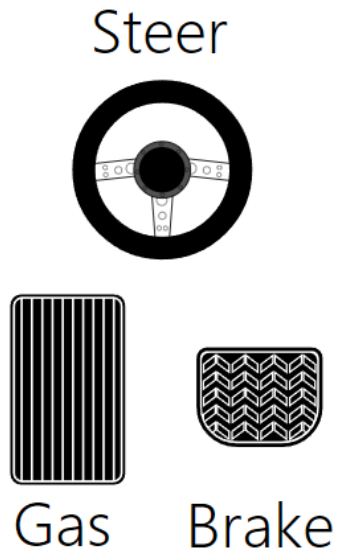
Control **Y**
(Action)

https://medium.com/apache-mxnet/analyze-driving-scenes-with-gluoncv-0-8-56ac37664977

Major Paradigms for Autonomous Driving

Sensor Input **X**

Mapping Function
Sensor input → Action
Y=F(**X**)



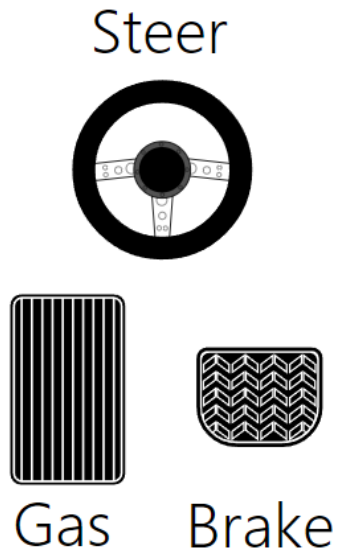
Control **Y**
(Action)

- Modular Pipelines
- End-to-End Learning
- Direct Perception

Major Paradigms for Autonomous Driving

Sensor Input **X**

Mapping Function
Sensor input → Action
Y=F(**X**)

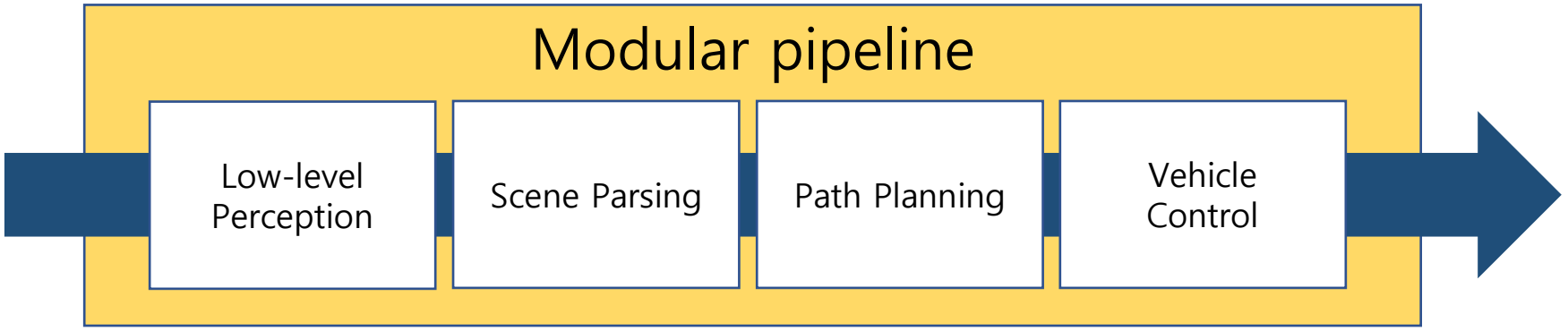


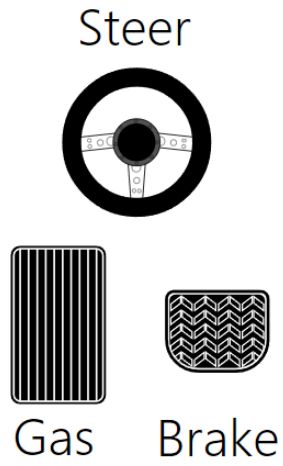
Control **Y**
(Action)

- **Modular Pipelines**
- End-to-End Learning
- Direct Perception

Modular Pipeline

Sensor Input **X**





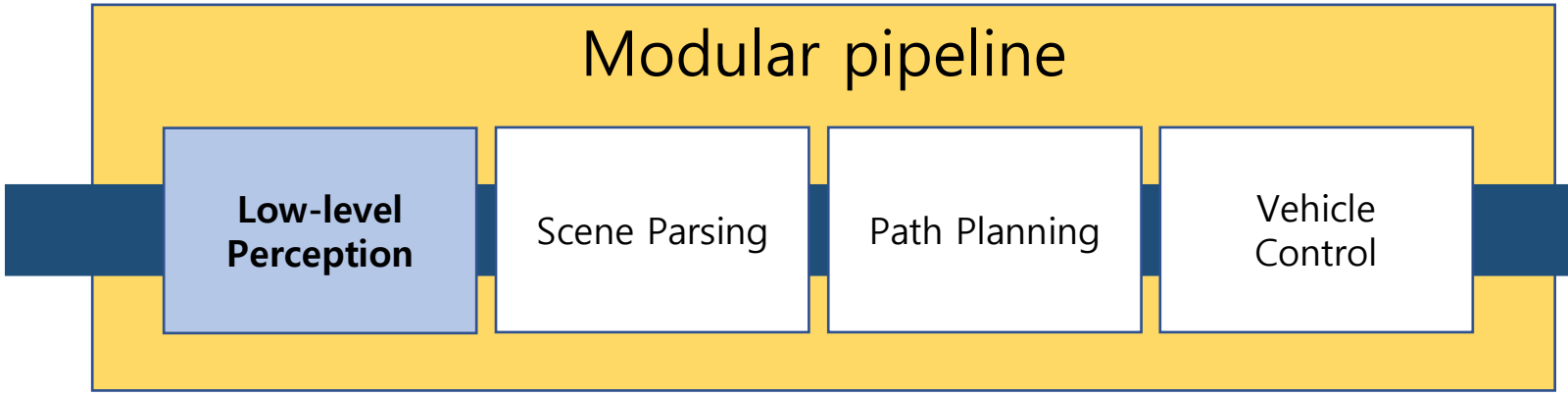
Control **Y**
(Action)

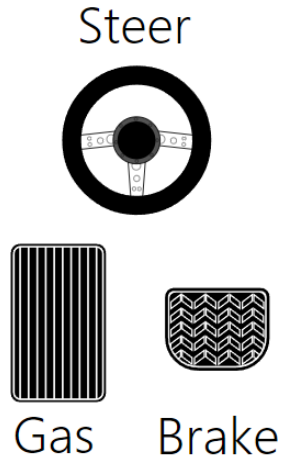
Each module produces input of the next module.

- Low-level Perception
- Scene Parsing
- Path Planning
- Vehicle Control

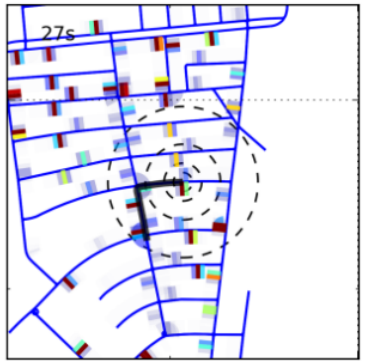
Modular Pipeline

Sensor Input **X**

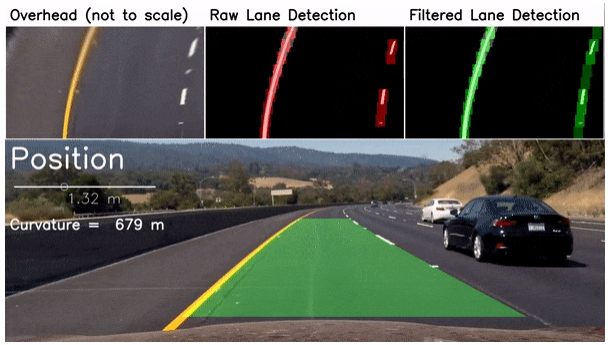




Control **Y** (Action)



GPS Location

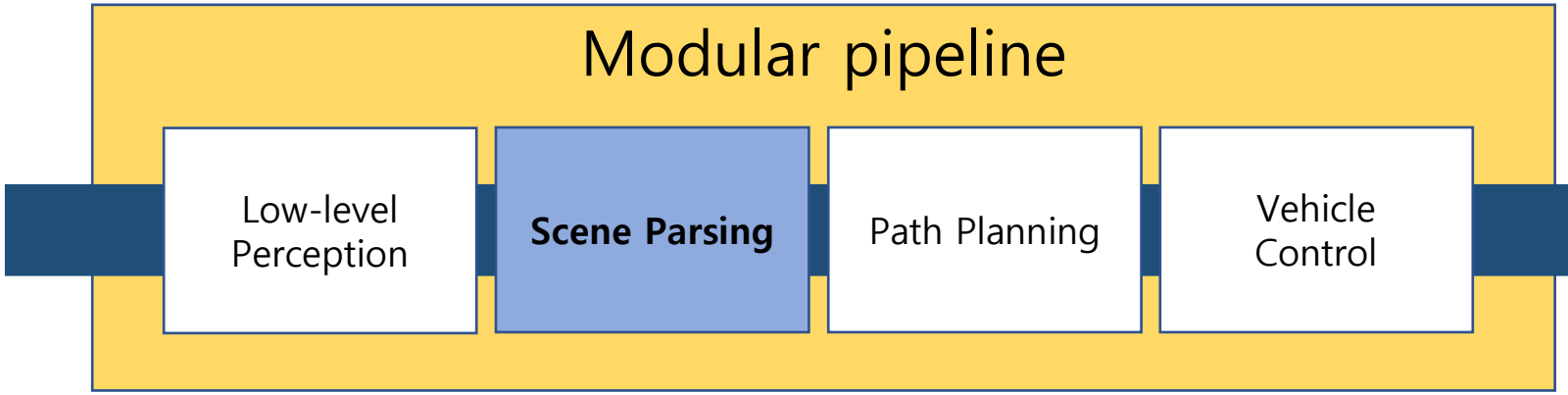


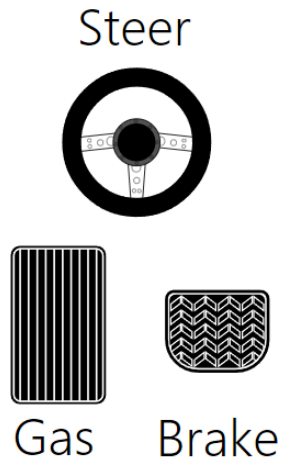
Lane detection

Detecting moving cars

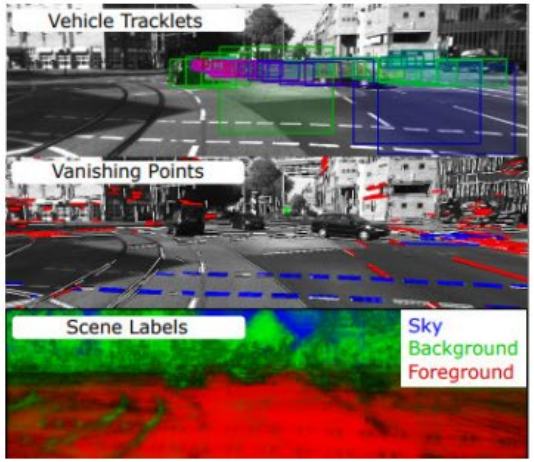
Modular Pipeline

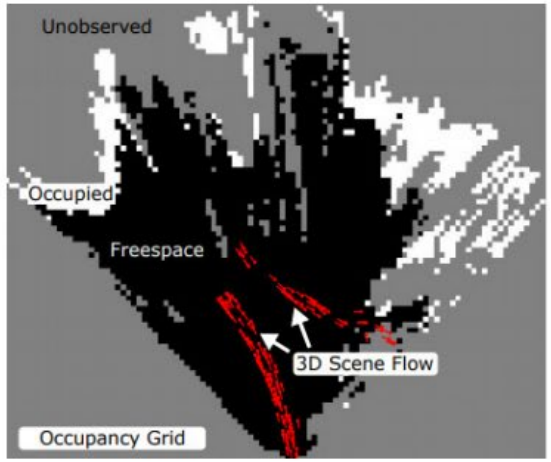
Sensor Input **X**

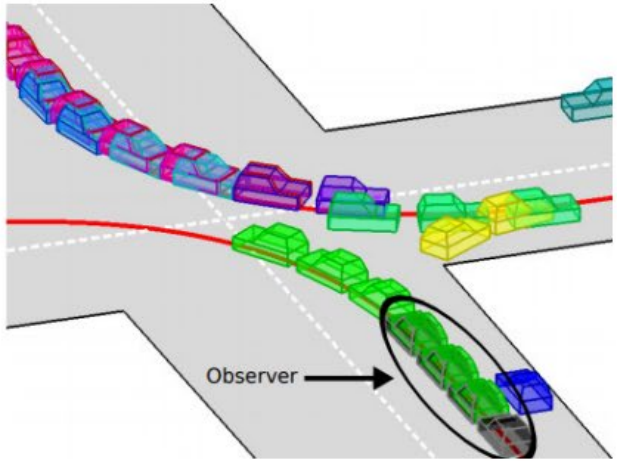




Control **Y** (Action)



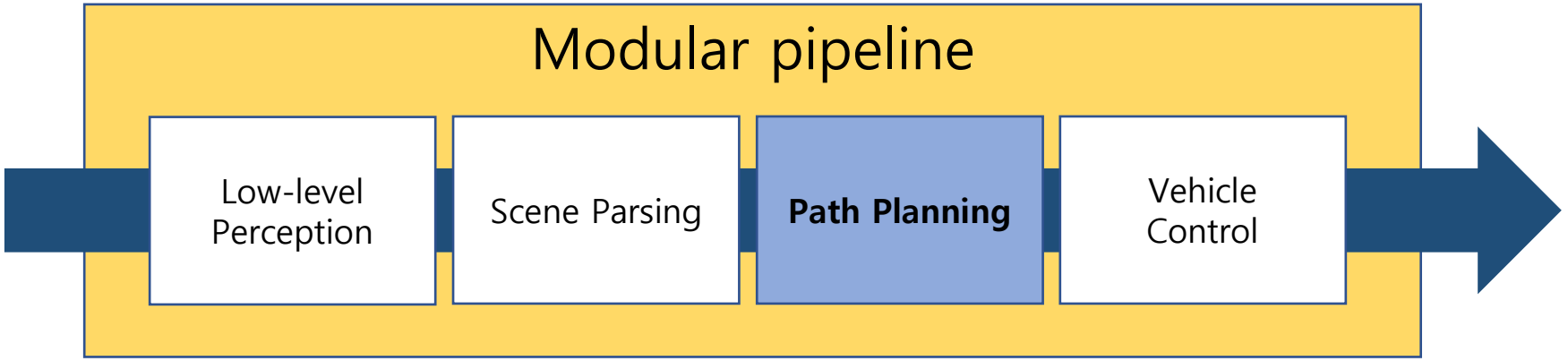


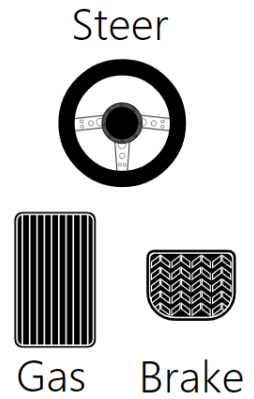


Parsing detected objects into one world space

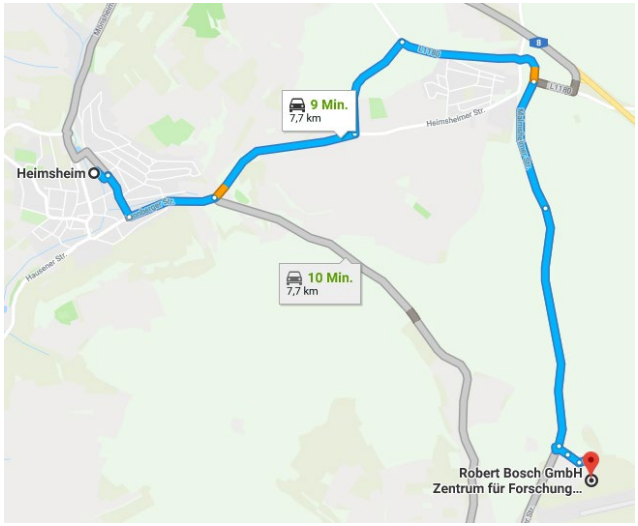
Modular Pipeline

Sensor Input **X**

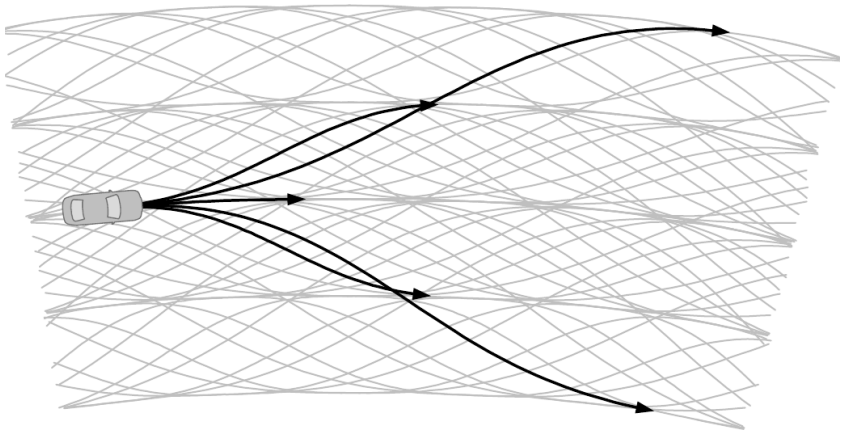




Control **Y** (Action)



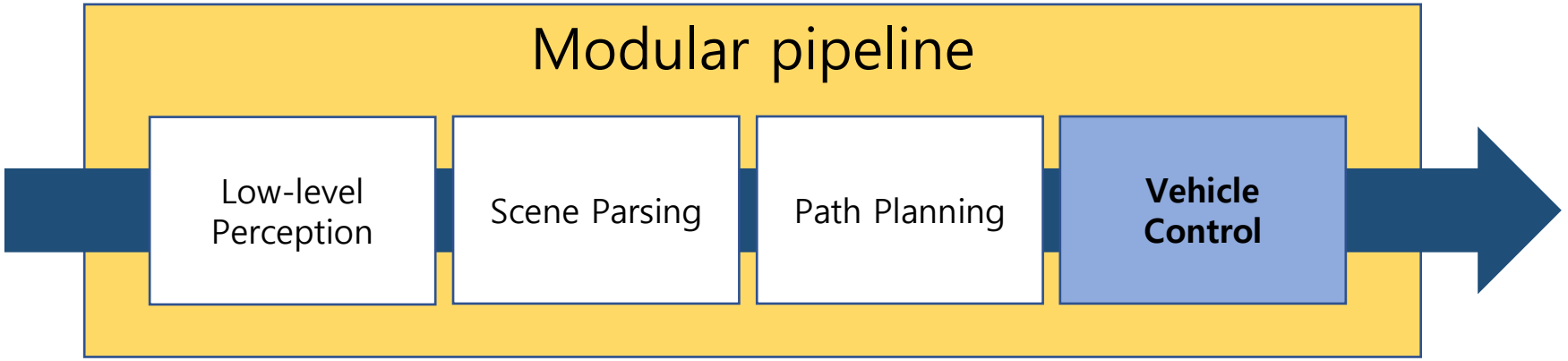
Which path is the fastest one?

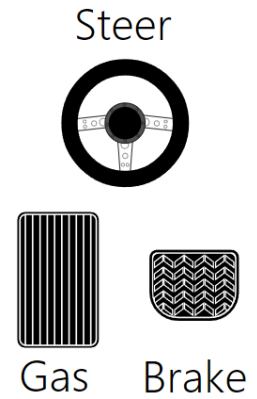


How to move a car on the road?

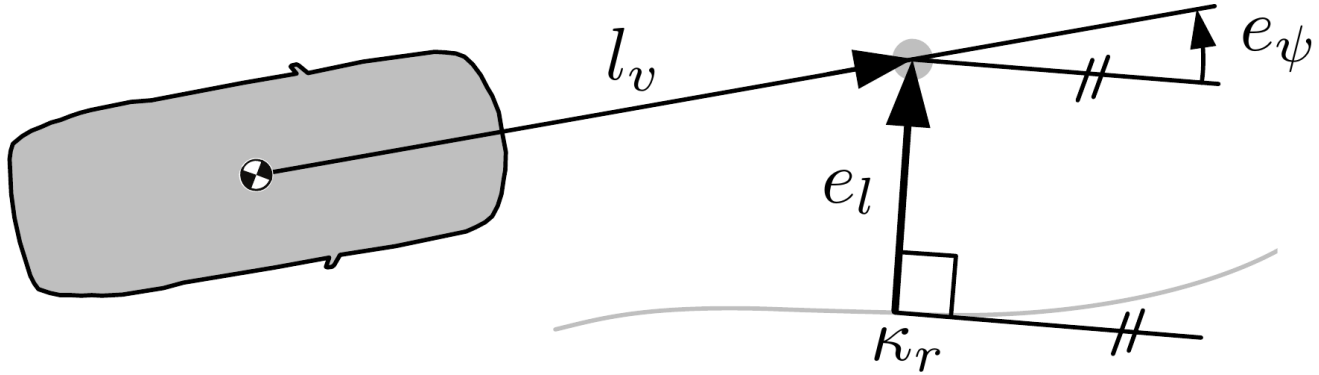
Modular Pipeline

Sensor Input **X**





Control **Y** (Action)



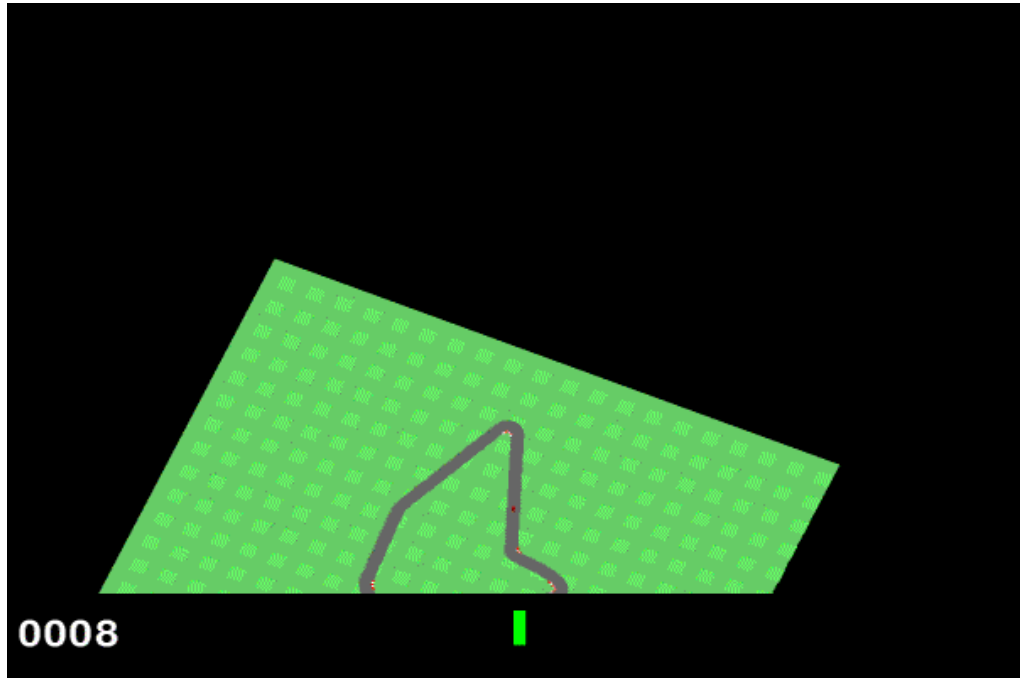
In order to follow the selected path,
How much should we steer the handle?
What speed should we go in?

Our Car Environment

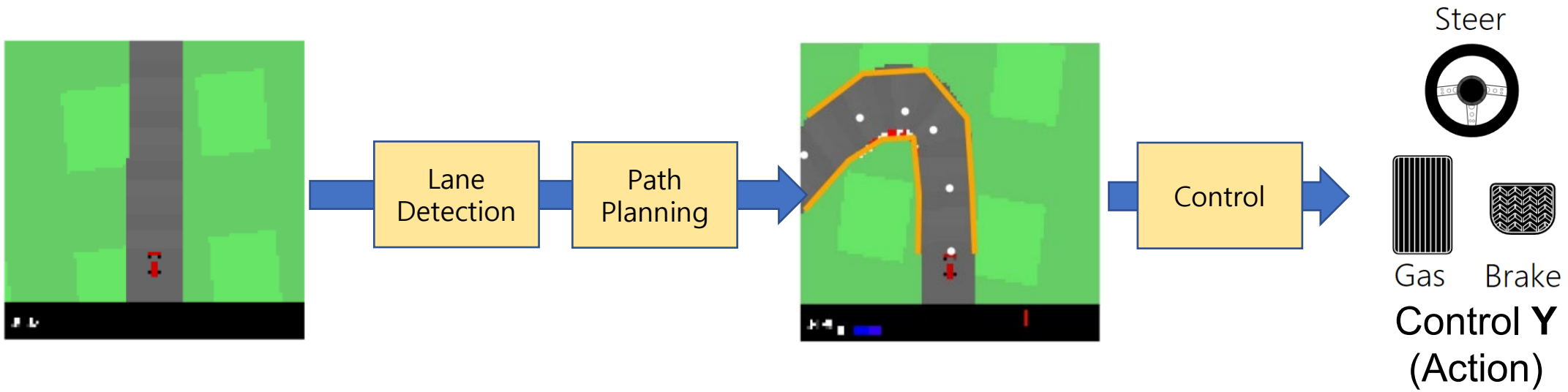
- Goal: implement a modular pipeline framework.

- Simulator: **OpenAI GYM**
  - https://www.gymlibrary.ml/
  - We will use Box2D-CarRacing

- CarRacing information
  - Action: steering, acceleration, brake
  - Sensor input: 96x96x3 screen
    - It shows car's states and path information.



Modular Pipeline Overview



Implement simplified version of modular pipeline.

You will understand basic concepts and get experiences of developing a simple self-driving application.

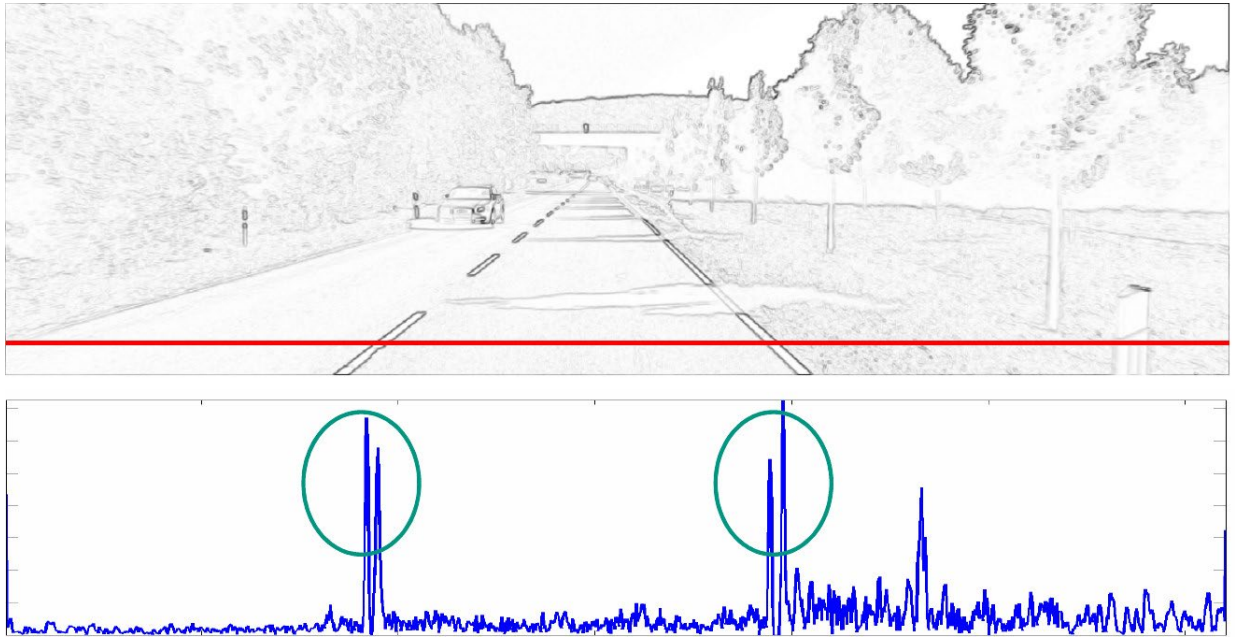
Lane Marking & Lane Detection

Lane Mark Detection

Using a gradient map or an edge-filtered image, we can detect lane marks by thresholding.

Consider points for which opposite gradient exists in vicinity.

An road image and its gradient map



An1D profile for lane marking detection

Edge Detection

By convolving an image with edge filters, two directional gradient maps are obtained.

Other edge kernels can be used ([-1 0 1] or [-1, 1]).

-1	0	+1
-2	0	+2
-1	0	+1

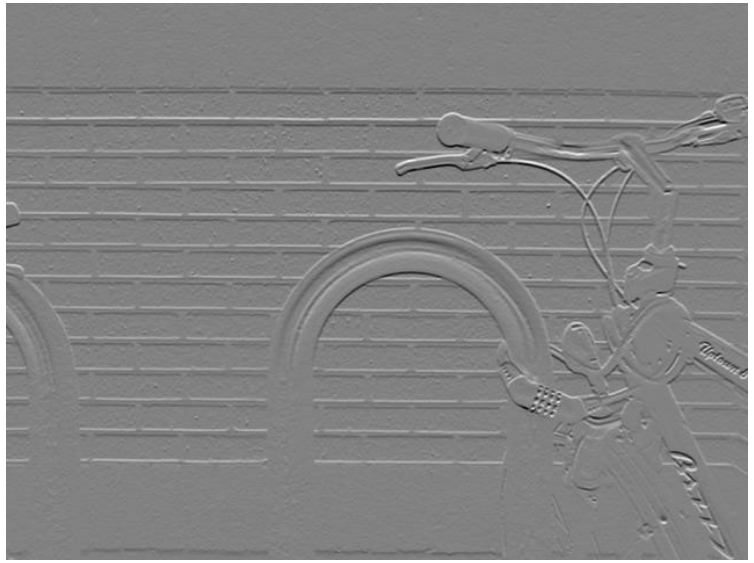
Gx

+1	+2	+1
0	0	0
-1	-2	-1

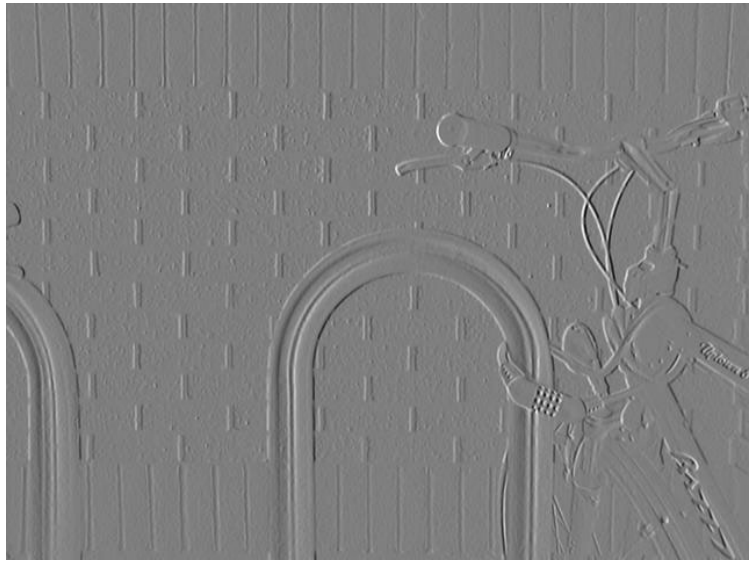
Gy

Sobel edge kernels

RGB Image



Gradient y

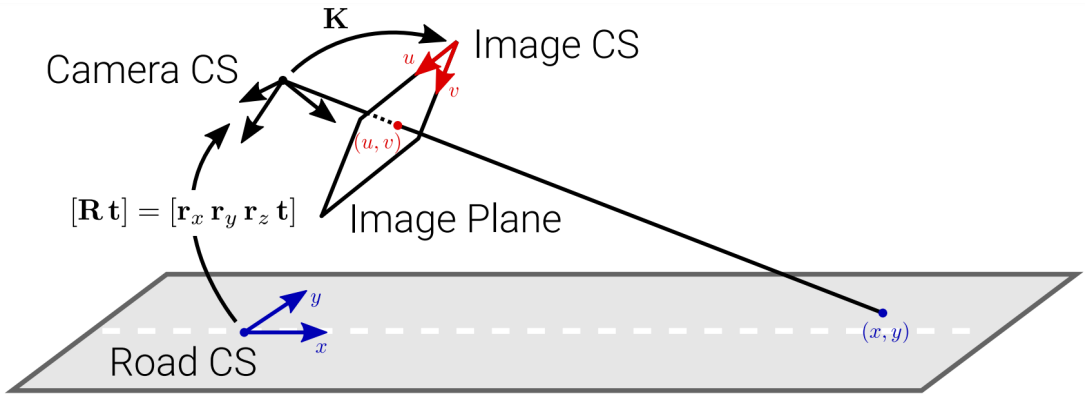


Gradient x

Inverse Perspective Mapping

In common, roads are on the plane.

If the 3D transform is known, we can project the road image on the ground plane.



The 3D overview of inverse perspective mapping

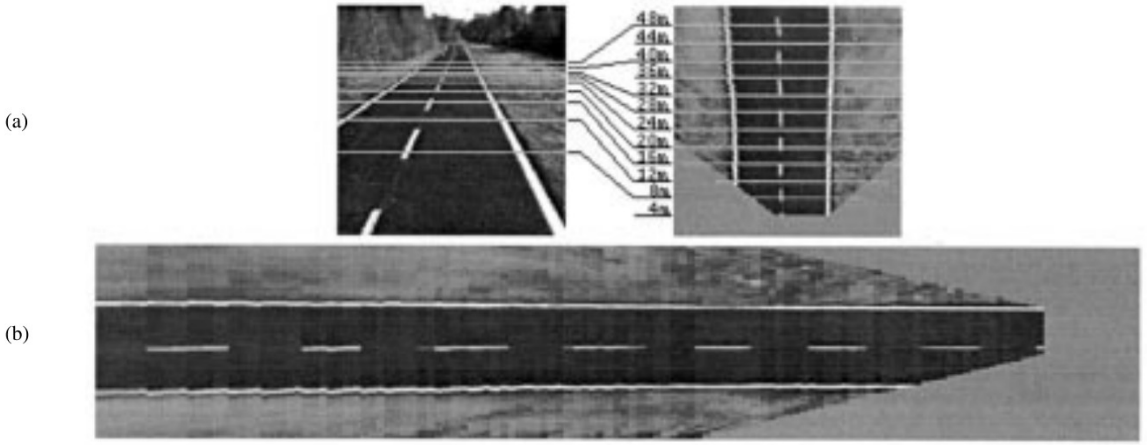
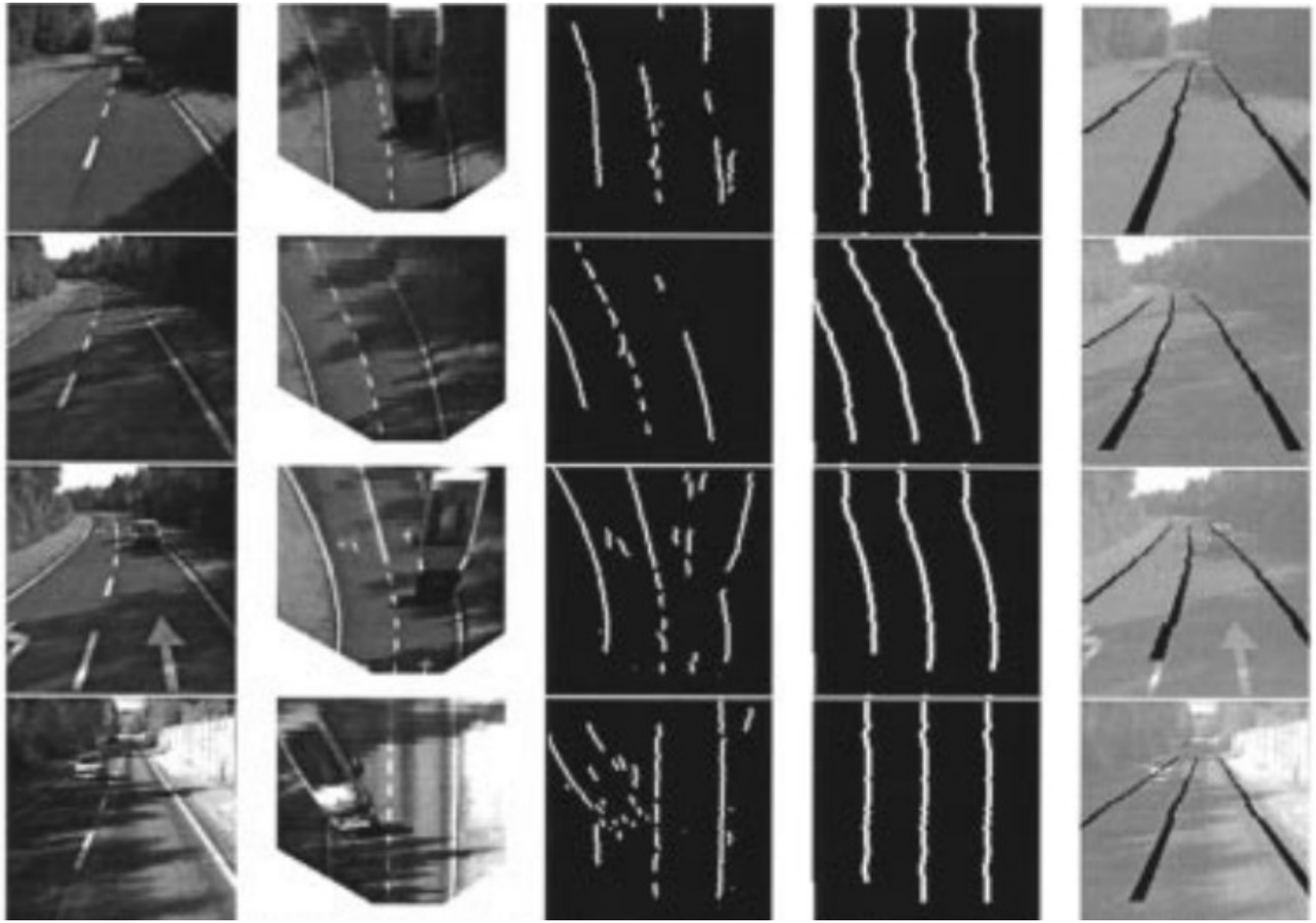


Fig. 8. (a) Horizontal calibration of the MOB-LAB vision system. (b) Rotated version of the remapped image considering an aspect ratio of 1 : 1.

Inverse Perspective Mapping

Inverse Perspective Mapping



Projected road maps and detected lanes

Parametric Lane Marking Estimation

In order to navigate the car, we need to fit detected mark pixels to a more semantically meaningful curve model.

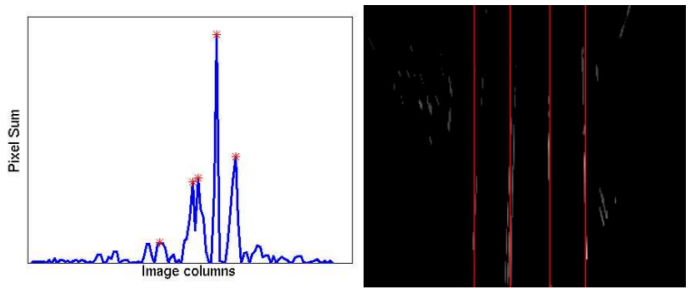


Fig. 5. Hough line grouping. Left: the sum of pixels for each column of the thresholded image with local maxima in red. Right: detected lines after grouping.

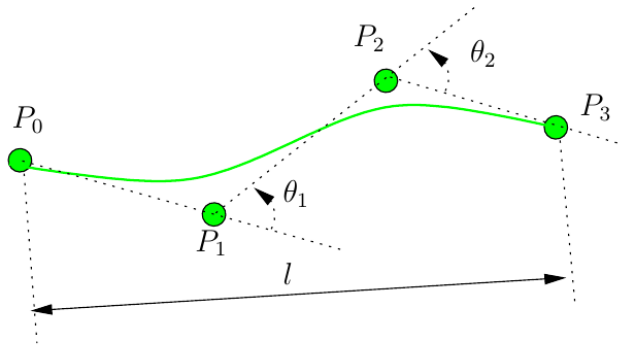


Fig. 7. Spline score computation.

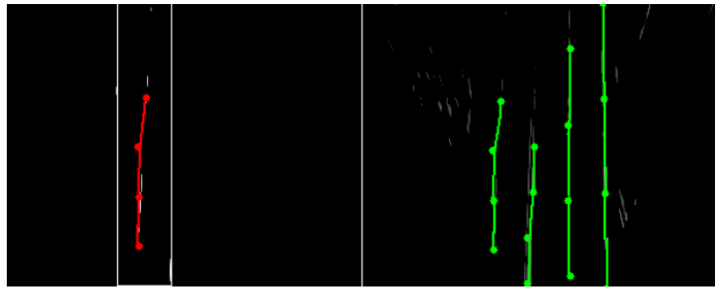


Fig. 8. RANSAC Spline fitting. Left: one of four windows of interest (white) obtained from previous step with detected spline (red). Right: the resulting splines (green) from this step

Algorithm 1 RANSAC Spline Fitting

```
for i = 1 to numIterations do
    points=getRandomSample()
    spline=fitSpline(points)
    score=computeSplineScore(spline)
    if score > bestScore then
        bestSpline = spline
    end if
end for
```

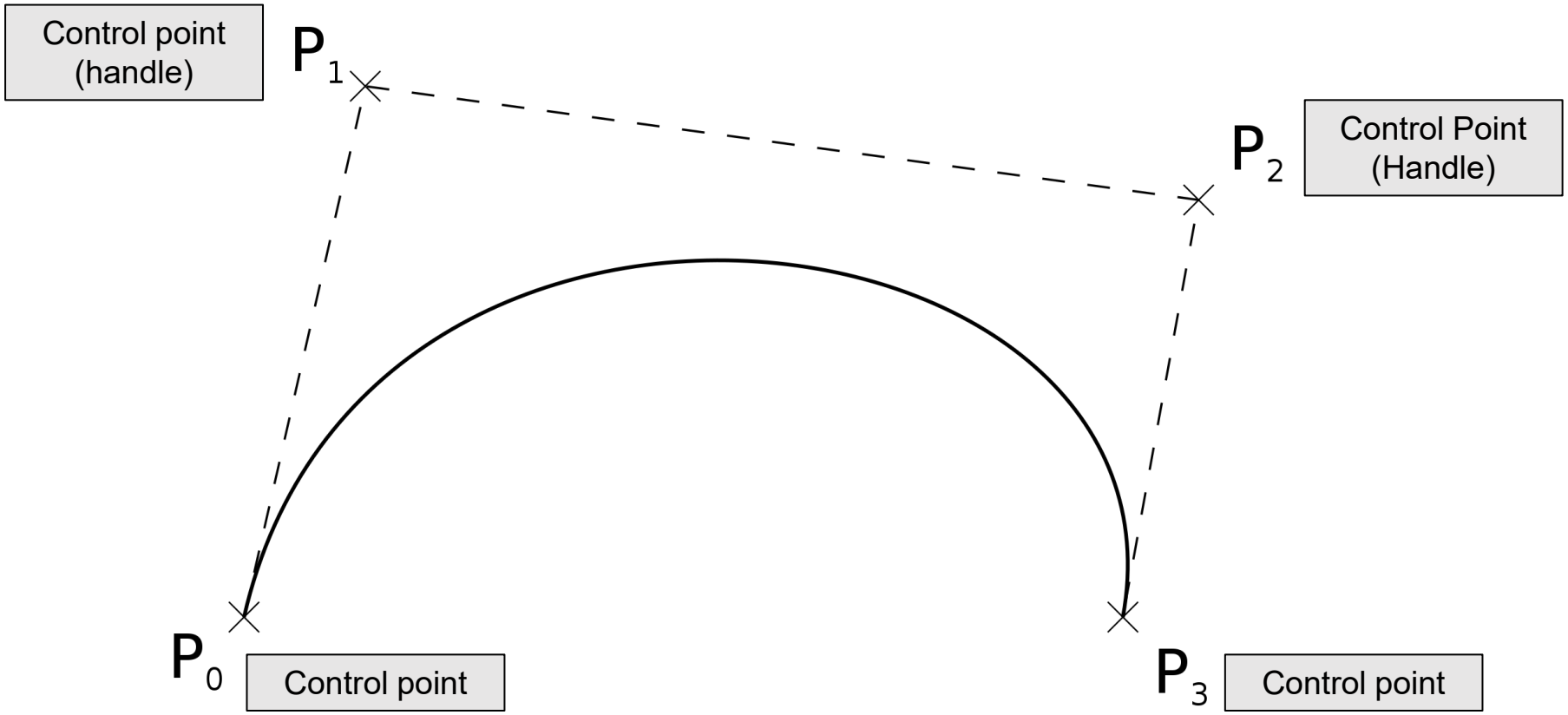
Fig. 10. Post-processing splines. Left: splines before post-processing in blue. Right: splines after post-processing in green. They appear longer and localized on the lanes.

Real time Detection of Lane Markers in Urban Streets, 2014

Curve

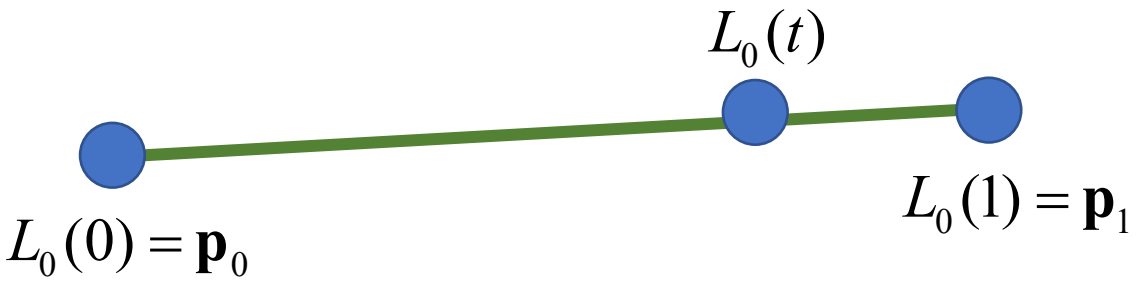
Bezier Curve

- A polynomial curve defined by control points.



Linear Bezier Curve

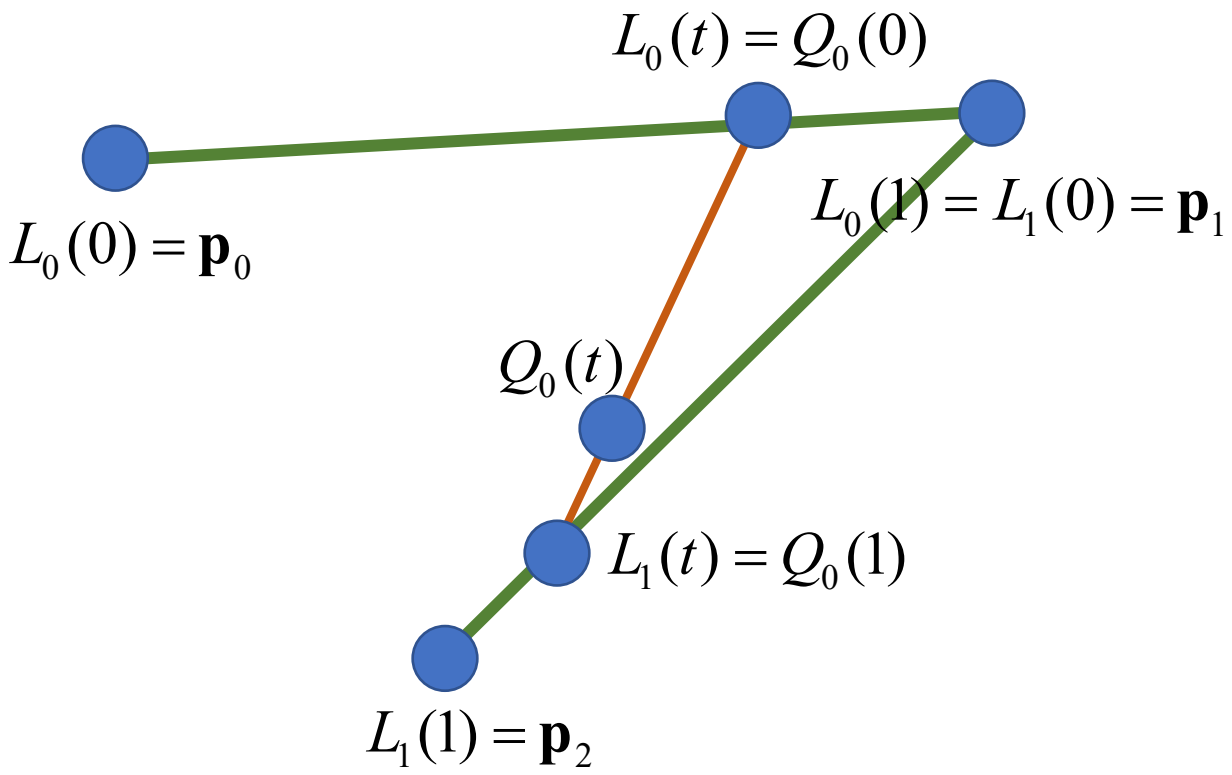
- Analogous to linear interpolation



$$L_0(t) = (1-t)\mathbf{p}_0 + t\mathbf{p}_1$$

Quadratic Bezier Curve

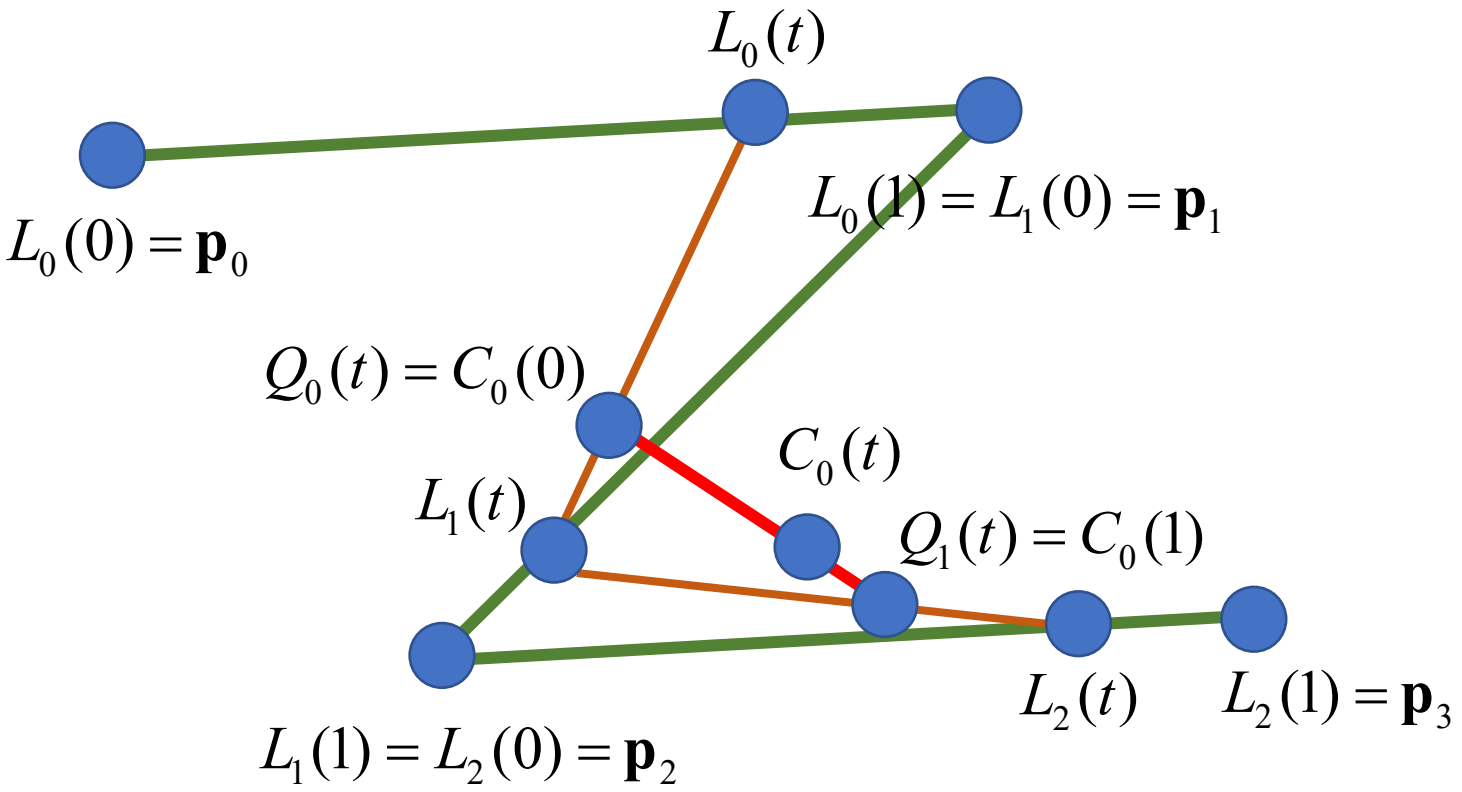
- Interpolation of two linearly interpolated points.



$$L_0(t) = (1-t)\mathbf{p}_0 + t\mathbf{p}_1$$
$$L_1(t) = (1-t)\mathbf{p}_1 + t\mathbf{p}_2$$
$$Q_0(t) = (1-t)L_0(t) + tL_1(t)$$
$$Q_0(t) = (1-t)^2\mathbf{p}_0 + 2(1-t)t\mathbf{p}_1 + t^2\mathbf{p}_2$$

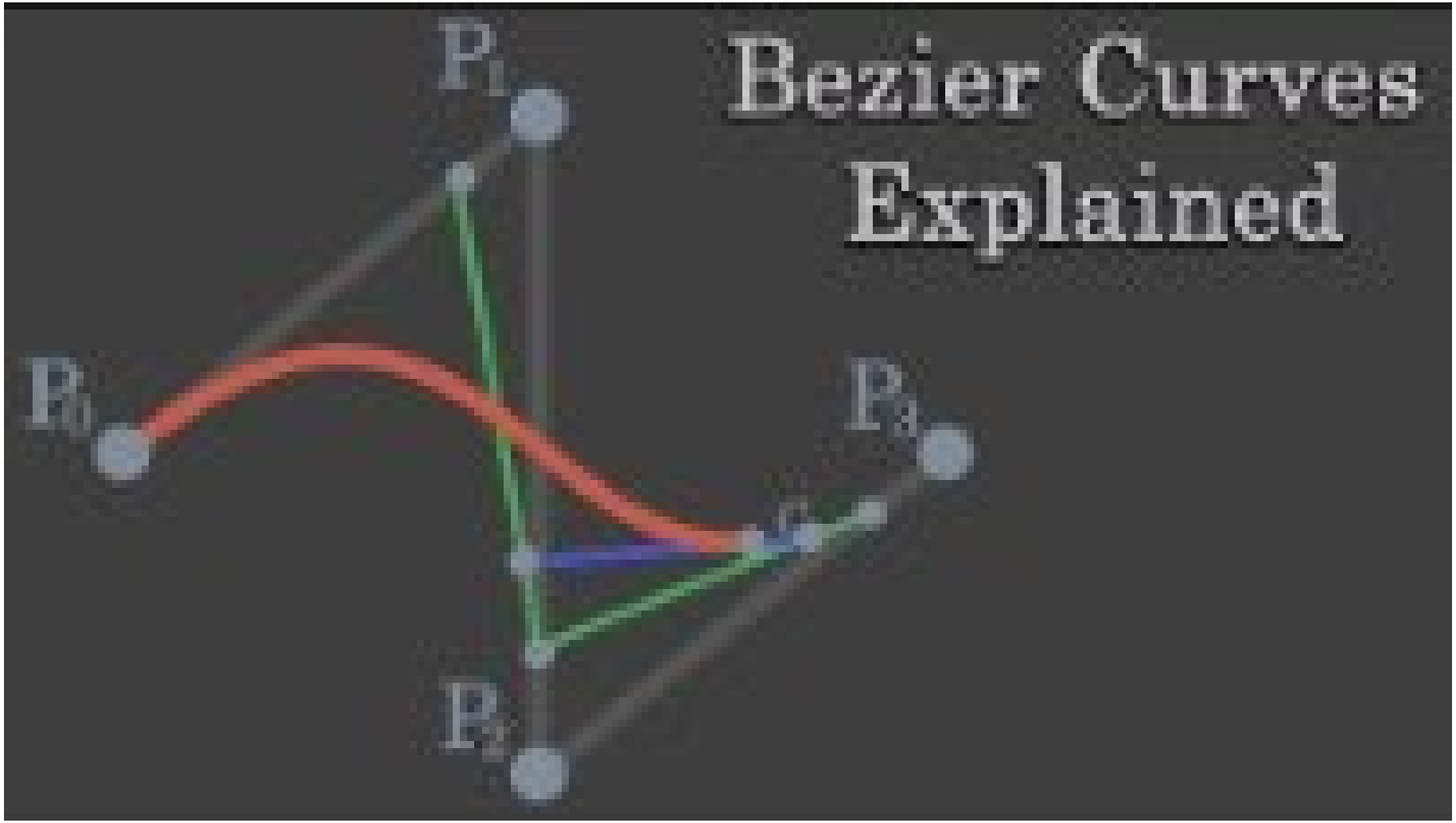
Cubic Bezier Curve

- Interpolation of quadratic points.



$$L_0(t) = (1-t)\mathbf{p}_0 + t\mathbf{p}_1$$
$$L_1(t) = (1-t)\mathbf{p}_1 + t\mathbf{p}_2$$
$$L_2(t) = (1-t)\mathbf{p}_2 + t\mathbf{p}_3$$
$$Q_0(t) = (1-t)L_0(t) + tL_1(t)$$
$$Q_1(t) = (1-t)L_1(t) + tL_2(t)$$
$$C_0(t) = (1-t)Q_0(t) + tQ_1(t)$$

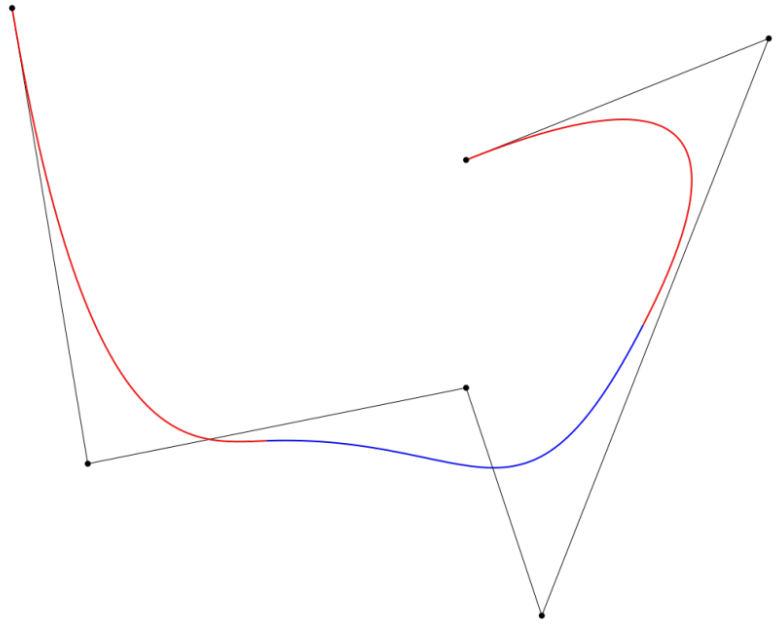
Bezier Curve animation



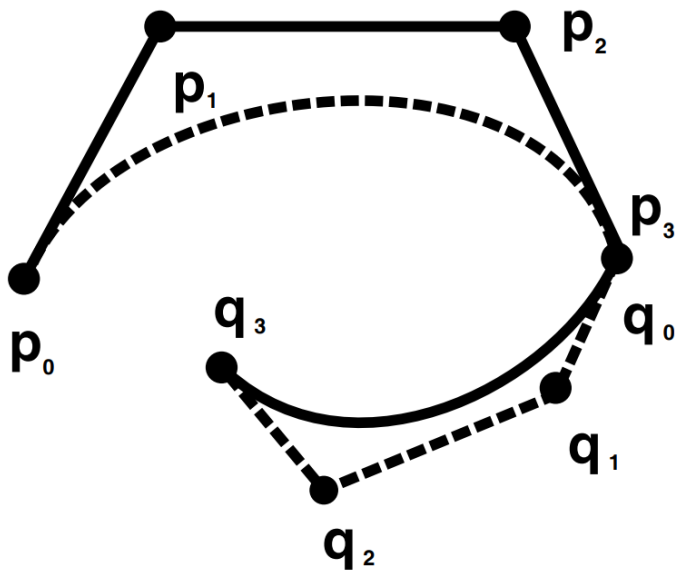
https://www.youtube.com/watch?v=pnYccz1Ha34

B(Basis)-Spline Curve

- Curve defined by a list of control points and the degree.
- A piece-wise polynomial curve (not same as a piece-wise Bezier curve).



B-spline curve

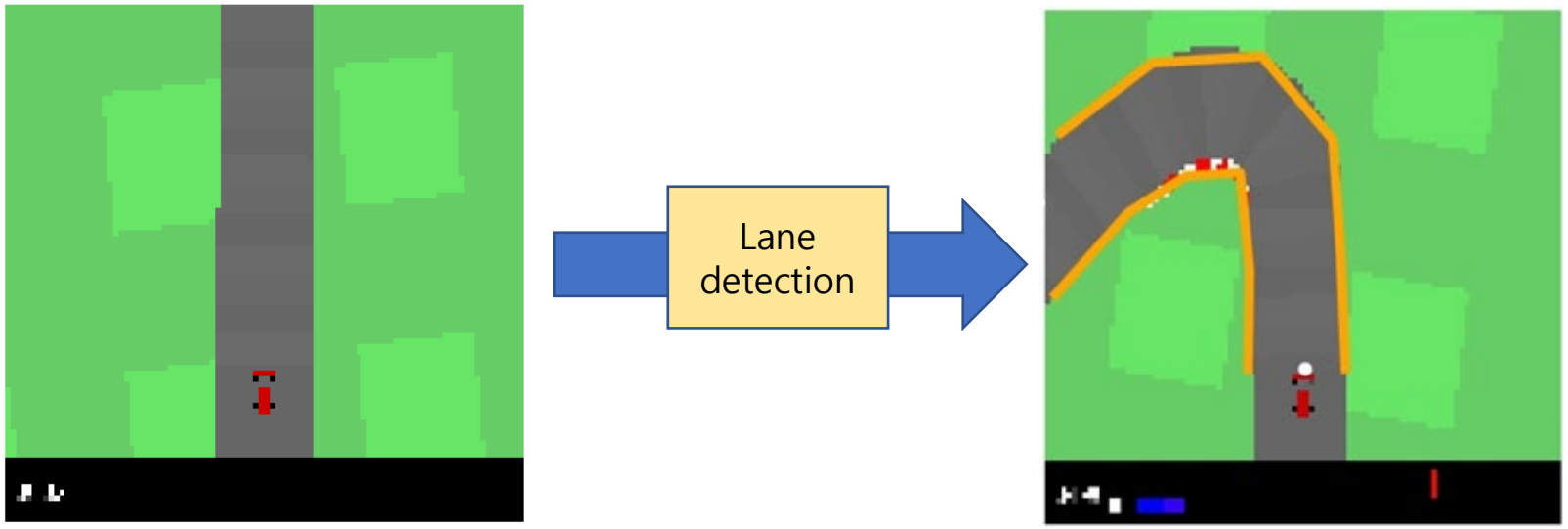


Piece-wise Bezier curve

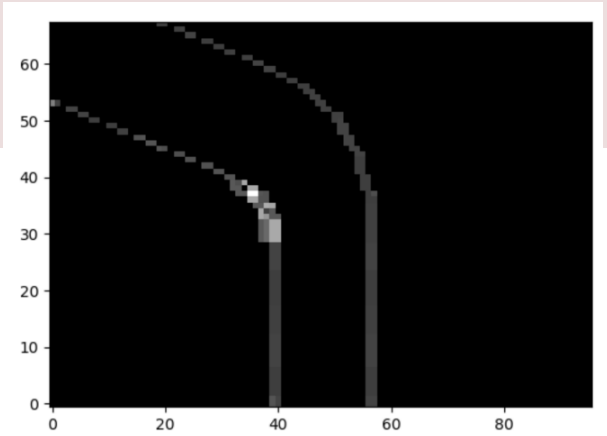
Experiment

Lane Detection

- Three steps
  - Edge detection: gradient thresholding
  - Assign edges to lane boundaries: successive nearest edge search
  - Spline fitting: parametric spline curve fitting



Lane Detection



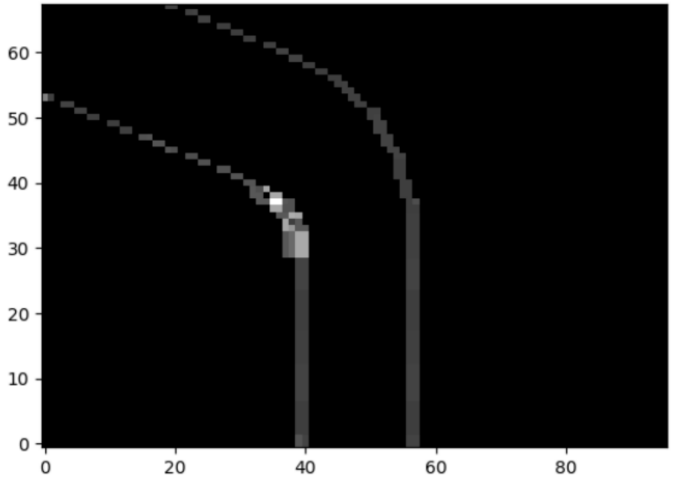
An gradient image with clipping

- Template
  - *lane detection.py*
  - *test lane detection.py* for testing
- **Edge detection:**
  - Translate the state image to a grey scale image and crop out the part above the car
    - *LaneDetection.cut_gray()*
  - Derive the absolute value of the gradients of the grey scale image and apply thresholding to ignore unimportant gradients.
    - *LaneDetection.edge_detection()*
  - Determine arguments of local maxima of absolute gradient per pixel row
    - *LaneDetection.find_maxima_gradient_rowwise()*

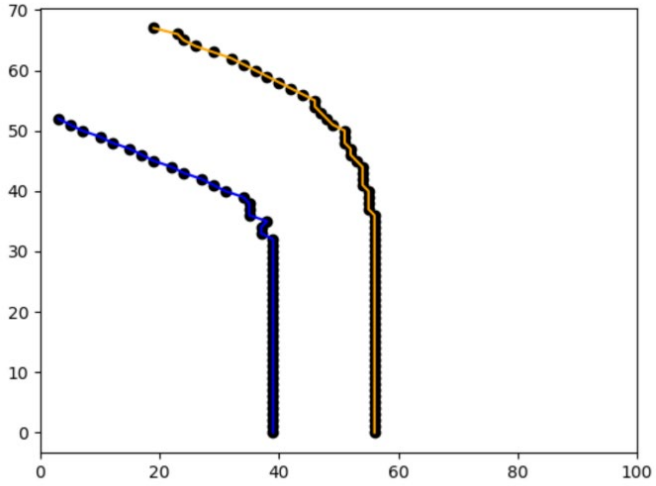
    Hint: use for example scipy.signal.find_peaks()
    https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

Lane Detection

- **Assign Edges to Lane Boundaries**:
  - Find arguments of local maxima in the image row closest to the car
    - 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
    - LaneDetection.lane_detection()



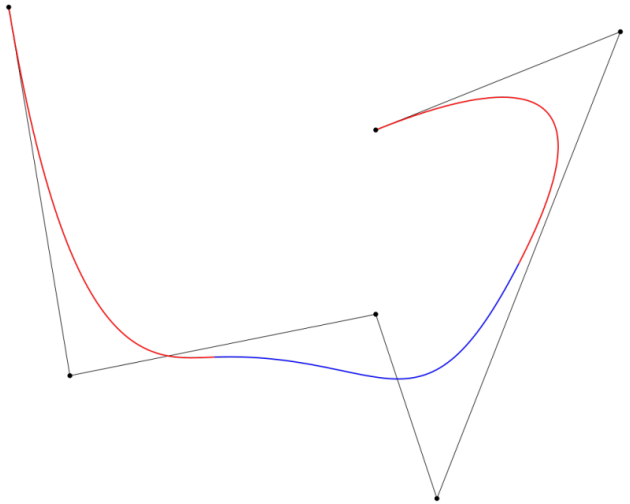
An gradient image with clipping



Peak points

Lane Detection

- **Spline Fitting**:
  - Fit a parametric spline to each lane boundary
    - LaneDetection.lane detection()
  - Use *scipy.interpolate.splprep* for fitting and *scipy.interpolate.splev* for evaluation (https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.splprep.html#id3)



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

Correct Lane detection

Spline Fitting

- *scipy.interpolate.splprep* will return parameters but we will take only the first one.

```
lane_boundary,_ = splprep([lane_boundary_points[1:,0], lane_boundary_points[1:,1]], s=self.spline_smoothness, k=2)
```

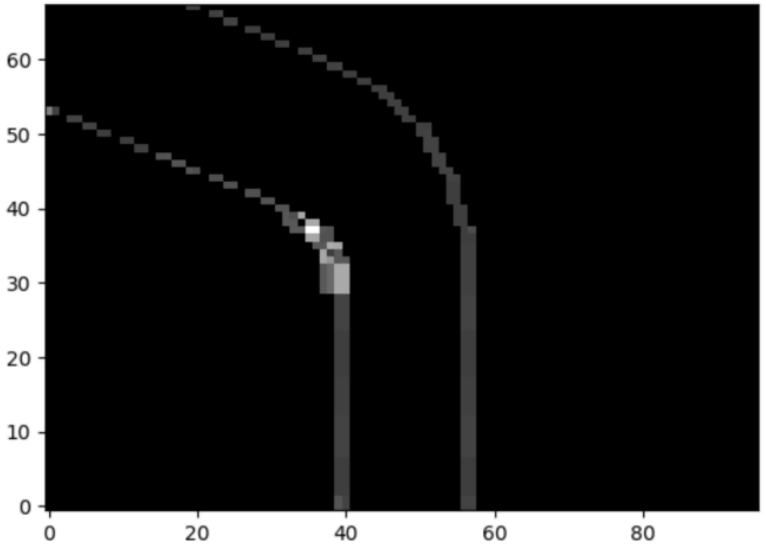
- With those parameters and function *scipy.interpolate.splev*, we can extract points on the curve.

```
t = np.linspace(0, 1, 5) # t = [0, 0.25, 0.5, 0.75, 1]
Interpolated_lane_boundary_points = np.array(splev(t, self.lane_boundary))
```

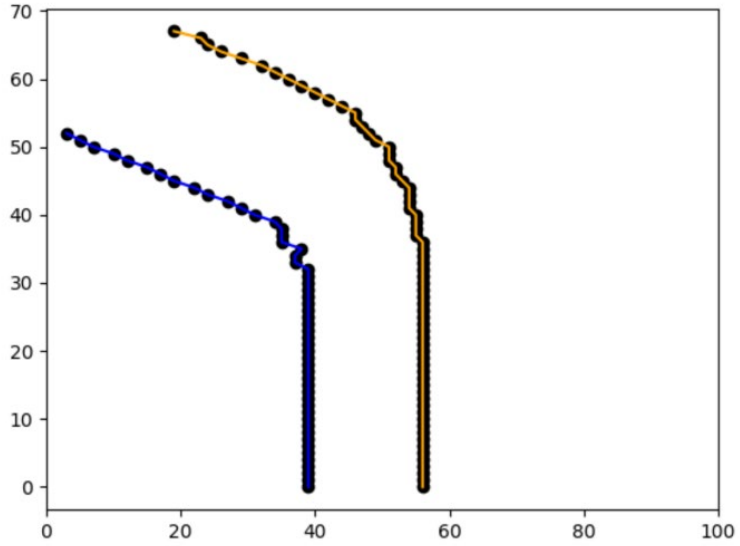
Lane Detection

- **Testing:**
  - Find a good crop for the part above the car, a good approach to assign edges to lane boundaries and a good choice of parameters for the gradient threshold and the spline smoothness.
  - Try to find failure cases

Lane Detection



An gradient image with clipping



Peak points

Correct Lane detection