MP1 Walkthrough

Minghao Jiang

Eric Liang

Professor Katie Driggs-Campbell

MP1 – Lane Detection

- Demo due 03/04/2021, report due 03/05/2021
- In this MP, there are 3 written questions and 4 coding questions
- The written questions will be related to convolution and filtering
- For the coding section, you will be implementing a lane detection algorithm running in ROS/Gazebo

Written Problems 1 - 2

Problem 1 Convolution (5 points) Prove that convolution is commutative. That is given two functions (or arrays) show that $f * g = g * f$. You may consider the functions to be one or two dimensional. State your assumptions clearly.

Problem 2 Filtering (5 points) Write down a 5×5 Gaussian derivative kernel in the x-direction. Does it find vertical or horizontal edges? Is this filter separable? Explain your answer.

Written Problem 3

Problem 3 Filtering in OpenCV (5 points) Under `mp1/src` folder, you will find a script named `filter_main.py`. Implement the `filter_gaussian` and `filter_median` filter functions and test their performances on salt-and-pepper noise and Gaussian(White) noise. Which filter is better for filtering out salt-and-pepper noise? Which filter is better for filtering out Gaussian(White) noise? Attach your result images in the report and explain why you think the specific filters work better for certain noises? (You will need to do `source devel/setup.bash` at this step. Refer to later sections for more detail)

(a) (b)

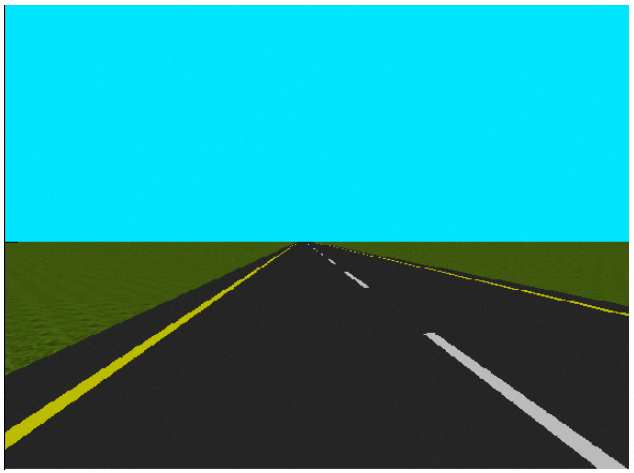
Figure 1: (a) Salt and Pepper Noise (b) Gaussian(White) Noise

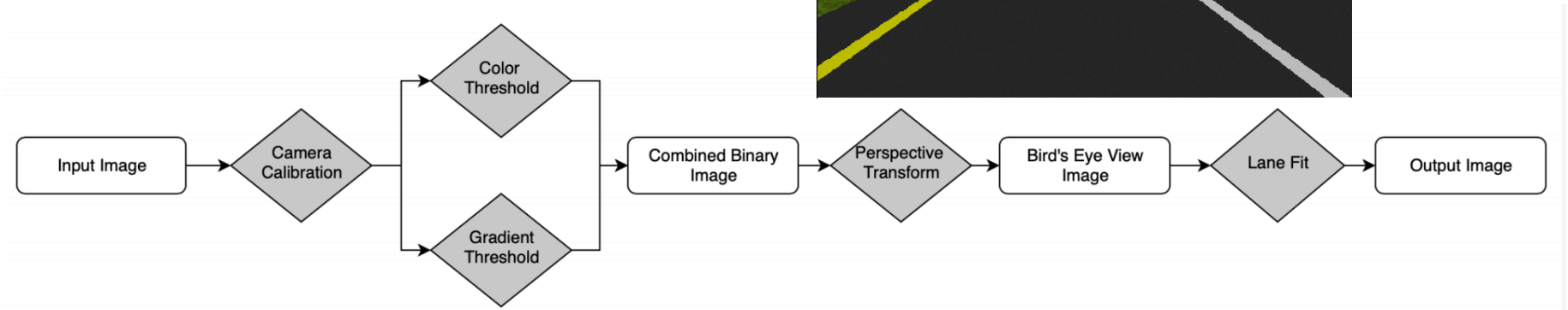
Written Problem 3

- Please attach result images in the report
- Before execute “filter_main.py”, you will need to ”source devel/setup.bash”

Implementation – Lane Detection Pipeline

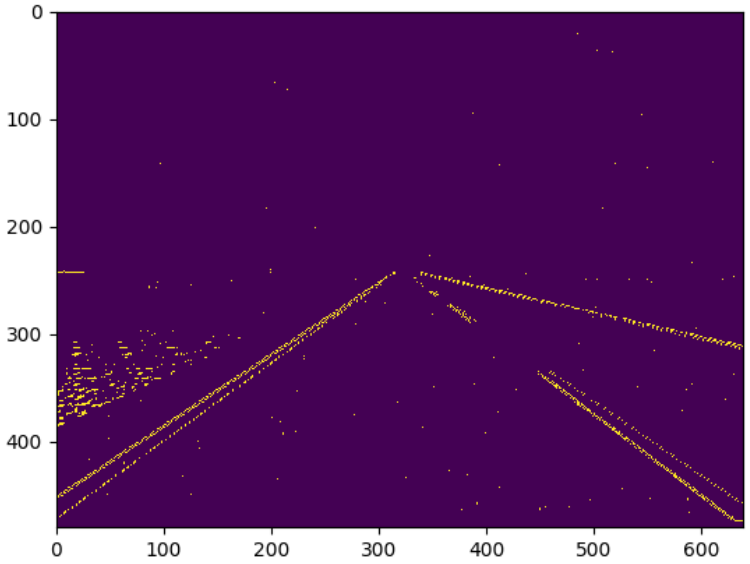
- Gradient Threshold
- Color Threshold
- Perspective Transform
- Lane Fit





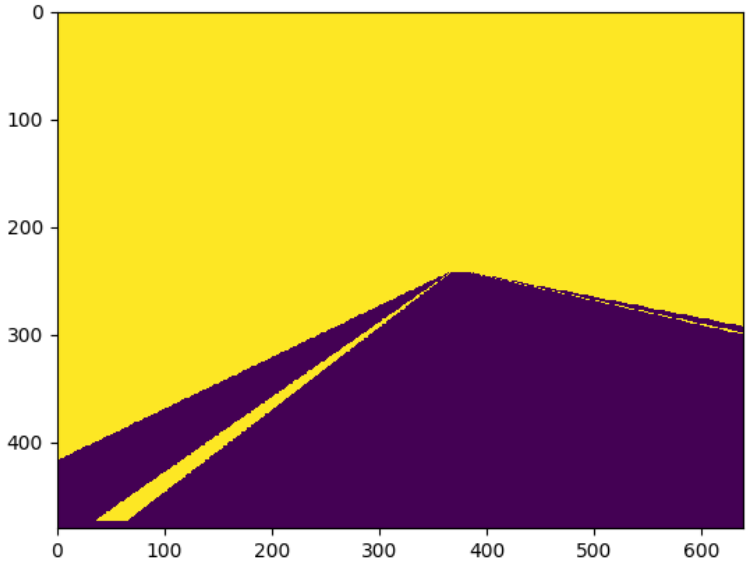
Gradient Threshold

- Goal: detect interesting features
- Inputs come from Camera
- Convert to grey-scale
- Use *Sobel* operator to find gradient in x and y axis
  - Hint: cv2.sobel
- Apply thresholds on the gradient values



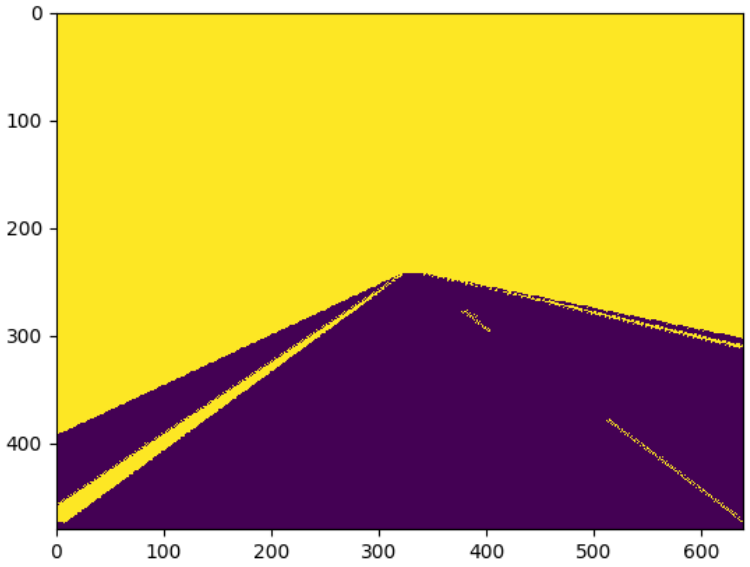
Color Threshold

- Goal: highlight white and yellow color
- Inputs come from camera
- Convert the image to different color spaces
  - E.g. RGB, HSL, HSV, etc.
  - Many people use HSL, but feel free to try other options
- Apply threshold on the different channels in the converted color spaces



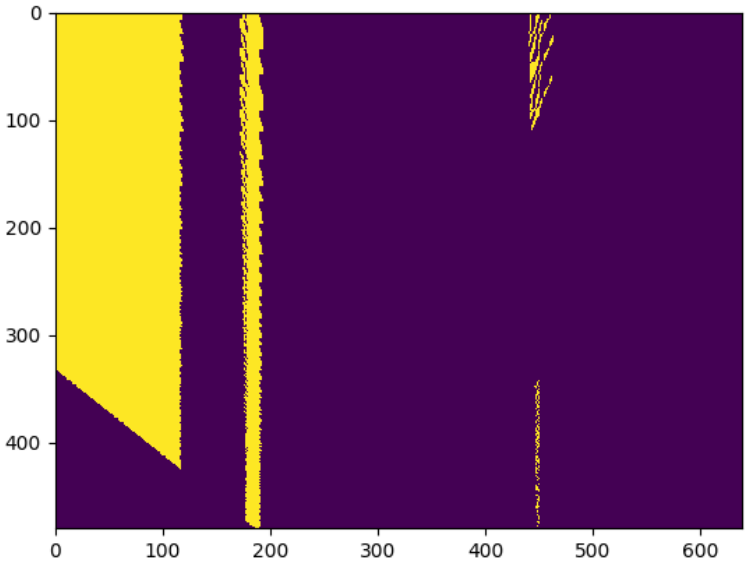
Combined Binary Image

- Combine the resulted binary images from previous steps



Perspective Transform

- Goal: convert the camera-view image to birds-eye view (look down from top)
- Inputs come from combination of results from Gradient and Color Threshold
- Fitting lanes in this view is much easier



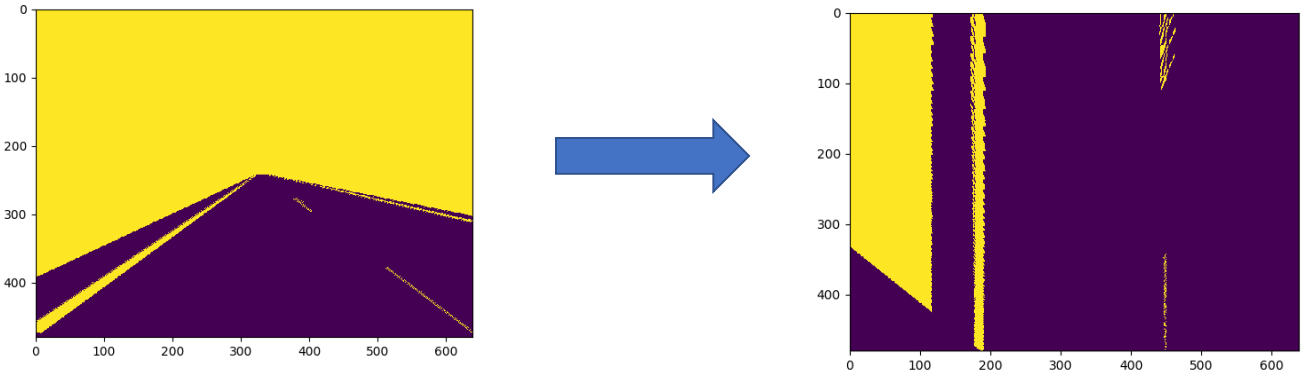
Perspective Transform

- Preserve
  - Collinearity: points lie on the same line before and after transformation
  - Ratio of distance: midpoint of a segment remains before and after transformation
- If (u, v) and (x, y) are the coordinates of the same point in the coordinate systems of the original perspective and new perspective, then:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

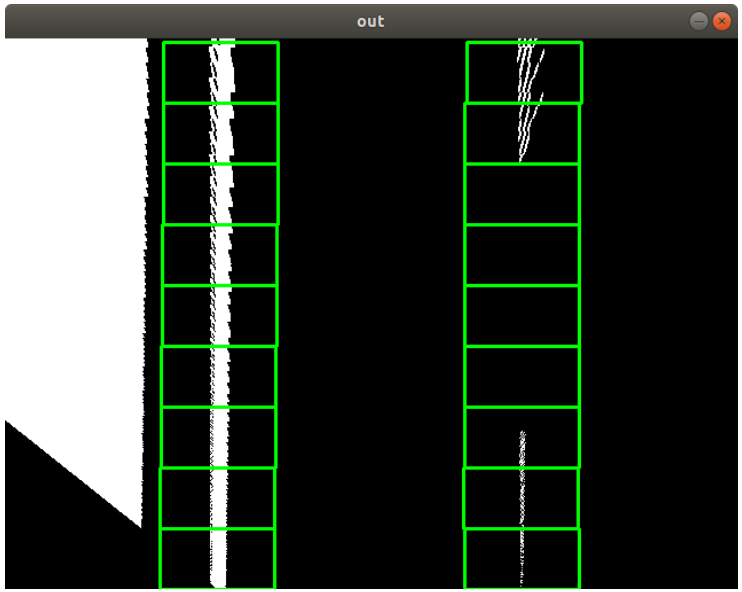
Perspective Transform

- In order to find the matrix, we need to find 4 points on the original images and map the same 4 points on the Bird's Eye View (in total 8 points)
- Then, use cv2.getPerspectiveTransform() to find the matrix of perspective transform
- Use cv2.warpPerspective() to get the Bird's Eye View image



Lane Fit

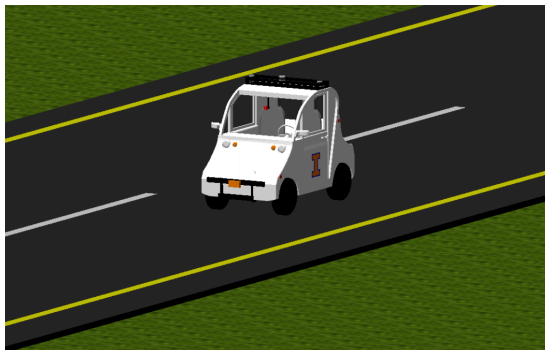
- Goal: find the lane in the birds-eye view image
- Input: from previous perspective transform image
- From bottom to up, identify lanes using sliding window approach
- Inside each window, try to find nonzero pixels
- Follow the comments from line_fit.py
- Hint: use cv2.imshow to print out the intermediate results

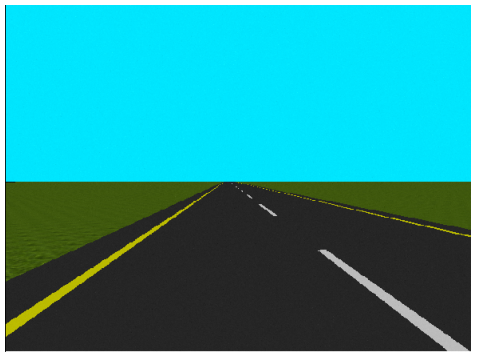


2 Lane Detection Scenarios

- Rosbag videos

- Gazebo simulation





Implementation Problems

Problem 4 (15 points) What are some interesting design choices you considered and executed in creating the different functions in your lane detection modules? E.g. which color spaces do you use? how do you determine the source points and destination points for perspective transform?

Problem 5 (25 points) In order to detect the lanes in both scenarios, you will need to modify some parameters. Please list and compare all parameters you have to modify and explain why do you think altering them will be helpful?

Problem 6 (35 points) Record 2 short videos of Rviz window and Gazebo to show that your code works for both scenarios. You can either use screen recording software or smart phone to record.

Problem 7 (10 points) One of the provided rosbags `(0484_sync.bag)` is recorded in snowfall condition. Your lane detector might encounter difficulties when trying to fit the lane in this specific case. If your lane detector works, please report what techniques you used to accomplish that. If not, try to find the possible reasons and explain them. (Note that you will not be evaluated by whether your model fits the lane in this case; instead we will evaluate on the reasoning you provide.)

Implementation Problem 7

- The lane is nearly invisible!
- Your lane detector may fail, but don’t panic
- Your solution will not be judged by whether your lane detector works
- Instead, we will evaluate your explanation on why it doesn’t work

Submission Guideline

- Write your solution in mp1_*groupname*.pdf
- Record 2 videos of both rosbag and gazebo scenarios
  - For lane detection in gazebo, you must show your lane detector works while the car is moving
- Upload your code (only *src/mp1/src*) to Google Drive/Box Folder and include the sharable link in your report
  - Do not upload entire workspace; rosbags are very large

Questions?