# BACKGROUND

Lane detection main objective is safety and lane keeping. Typically, we have markers on roads which we call lanes. There are different types of lanes such as discontinuous white markers, continuous white markers, continuous yellow markers. Detecting these lines is referred as lane detection. Conventional lane detection method uses cameras, LIDARS to detect lanes.

A car parked on the side of a road

Description generated with very high confidence

Figure 1: A typical scenario of conventional lane detection method. This type of system uses cameras, LIDARS to detect lanes.

Source: www.google.com

Cameras play a crucial role in lane detection since the input image/video is captured using cameras. On the other hand, LIDARS are used to mark lanes and then detect. Combination of these two are proved to be effective in lane detection. A typical algorithm for lane detection involves image capturing, converting to usable format, noise reduction and algorithm to detect lines.

A screenshot of a cell phone

Description generated with very high confidence

Figure 2: Shows the algorithm for lane detection.

Capturing an image is a color picture succession taken from a moving vehicle. A color camera is mounted inside the vehicle at the front-view mirror along the focal line. It takes the pictures of nature in front of the vehicle, including the street, vehicles on the street, roadside, and occasionally occurrences on the street. The path location framework peruses the picture succession from the memory and starts processing.

Conversion to Gray Scale is converting RGB space into Grey space as it is difficult to process and put edge detection algorithms on RGB image as well as the processing time required for grey image processing is comparatively less compared to RGB space.

Noise Reduction is important as noise removal is a pre-requisite for efficient edge detection and presence of noise in our system will hinder the correct edge detection. Input image might contain shadows which is noise for our system. These shadows contain distinctive boundary which helps in removing it from image. A shadow edge image can be created by applying edge-detection on the invariant image and the original image, and selecting the edges that exist in the original image but not in the invariant image and to reconstruct the shadow free image by removing the edges from the original image using a pseudo-inverse filter.

Edge Detection Lane boundary is defined by sharp contrast between the road surface and painted lines which is high frequency component of the image. The retrieved edges in an image is the lane.

Line Detection is acquiring lanes from the edges. The edges are present in certain area of interest hence anything outside this area of interest can be rejected. Left and right lanes are then detected using dominant lines in each half of the edge image at 45° and then apply Hough transform to get Hough lines and project their intersection at horizon.

A sign on the side of a road

Description generated with high confidence

Figure 3: Shows Left and right lanes are detected using dominant lines in each half of the edge image at 45° and Hough lines are projected and their intersection at horizon are shown.

Source: www.google.com

An example of lane detection system (LDS) based on software and hardware co-design. In combining both hardware and software designs, it can achieve a real-time lane detection within a processing time of less than 50ms. The hardware implemented by FPGA chip captures the lane image from CCD camera within a time less than 10ms, which is faster than by software captures. In software design, the global edge detection can transfer the gray level image to binary pattern and show the edge of the object. Then, using this binary pattern find out the traffic lane location with following algorithm like the peak-finding and grouping, edge connecting, lane segment combination, lane boundaries selection.

Another lane detection method is through deep learning using Python and OpenCV. We follow the same procedure as we do for conventional lane detection plus we train our AI about surrounding lanes from which after few training miles the car should navigate by itself in the lanes.

Even with all these systems there is one major area where these systems fail that is when it snows or rains. This is due to lack of visibility to cameras or LIDARS. To solve this issue, we can use thermal imaging along with color camera to detect lanes even in bad weather conditions and get better lane detection results.

# METHOD

## PROPOSED ARCHITECTURE:

Camera

Captured Image

A screenshot of a cell phone

Description generated with high confidence

Figure 4: Shows our solution for lane tracking using IR imaging.

# RESULTS

A screen shot of a computer

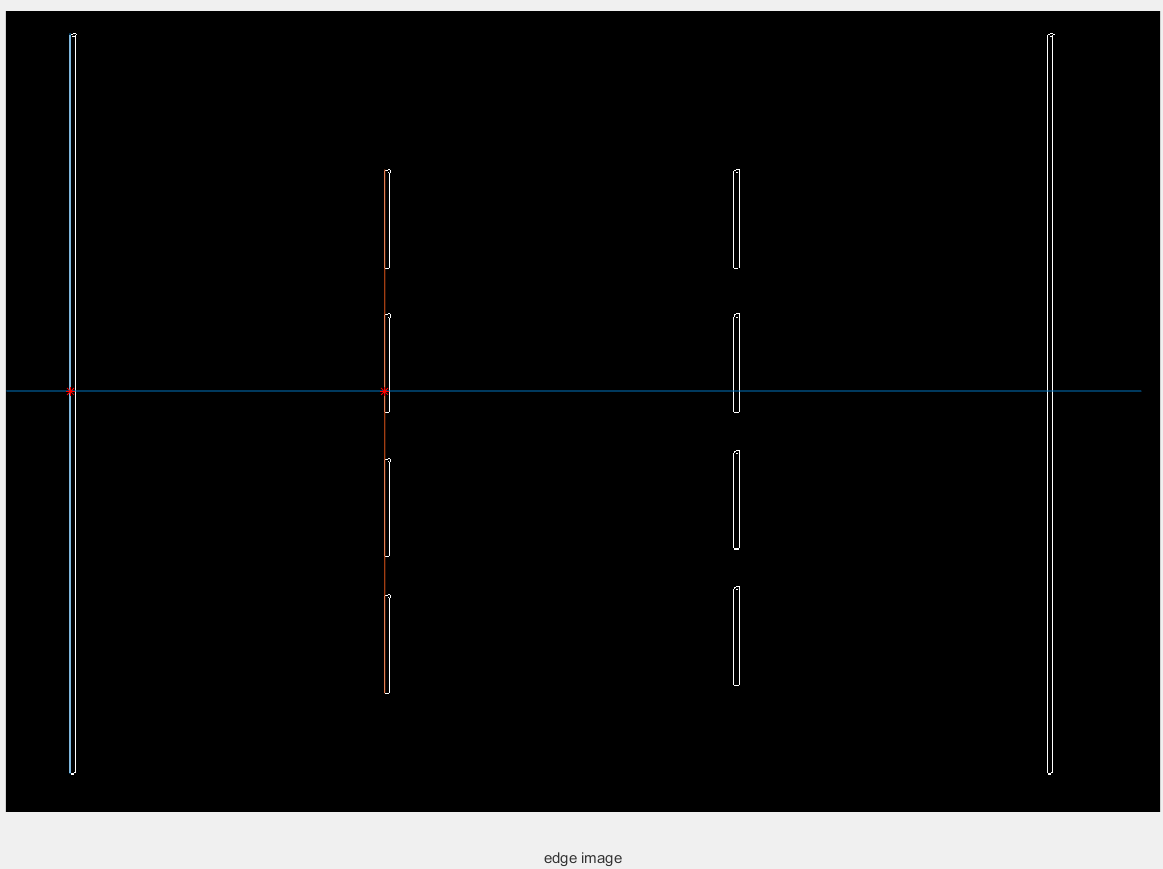
Description generated with very high confidence

Figure 5: Case 1: four lines parallel to each other, and the lines are placed depicting lanes on a road. We see that the edges are detected along with x intercepts shown by red dots.

A screenshot of a computer screen

Description generated with very high confidenceA screenshot of a computer screen

Description generated with very high confidence

Figure 6: Case 2: four lines parallel to each other, and the lines are placed depicting lanes on a road. We see that the edges are detected along with error message showing car is going out of lane.

A screenshot of a computer

Description generated with very high confidenceA screenshot of a computer screen

Description generated with very high confidence

Figure 7: Case 3: two lines showing real life like view of lanes captured, and the lines are placed depicting lanes on a road. We see that the edges are detected along with error message showing car is going out of lane.

# EXPECTED OUTCOME:



Expected results will be like this. Detection of the lane is first done by the thermal image sensing camera with the thermal source laid on the road and then with the help of the thermal sources the green pattern is obtained and the white dotted lines are the direction prediction with respect to the green lines. The image source was taken from youtube video which is using RANSAC and quadratic equation where, HSV filter is used for prediction. Normally it works for white lanes.

# APPENDIX:

import cv2

clc;

close all;

clear all;

f= imread('C:\Users\Kaushik\Pictures\trail32.png');

b2=[-1 -2 1; -2 0 2 ; -1 2 1];

r2 = imfilter(f, b2, 'replicate');

figure(1)

imshow(f),xlabel('original image')

R3=rgb2gray(r2);

the\_edge = edge(R3);

figure(2)

imshow(the\_edge),xlabel('edge image')

[y, x] = find(the\_edge);

if x(100)> 100

h = msgbox('Check lane', 'Warning','error');

end

for j=1: length(x)

if x(j) > 50 && x(j) < 100

xLine(j) = x(j);

yLine(j) = y(j);

end

end

i = 1;

for j=1: length(x)

if x(j) > 300 && x(j) < 350

xLine1(i) = x(j);

yLine1(i) = y(j);

i = i+1;

end

end

x1= [1 1000];

y1=[335 335];

% linear polynomial

p1 = polyfit(x1,y1,1);

p2 = polyfit(xLine,yLine,1);

p3 = polyfit(x1,y1,1);

p4 = polyfit(xLine1,yLine1,1);

% calculate intersection

x\_intersect = fzero(@(x1) polyval(p1-p2,x1),1);

y\_intersect = polyval(p1,x\_intersect);

x\_intersect1 = fzero(@(x1) polyval(p3-p4,x1),1);

y\_intersect1 = polyval(p3,x\_intersect1);

if x\_intersect > 70 || x\_intersect1 < 70

h = msgbox('Check lane', 'Warning','error');

end

for line in lines:

for x1,y1,x2,y2 in line:

cv2.line(r2,(x1,y1),(x2,y2),(255,0,0),10)

lines\_edges = cv2.addWeighted(the\_edge, 0.8, r2, 1, 0)

imshow(lines\_edges);

line(x1, y1);

hold on;

plot(xLine, yLine);

hold on;

plot(x\_intersect,y\_intersect,'r\*')

hold on;

plot(xLine1, yLine1);

hold on;

plot(x\_intersect1,y\_intersect1,'r\*')