ECEN 5763 Embedded Computer Vision: Final project Report

Exercise 6

Submission Date:11th Aug 2024

Authors: Mohit Chaudhari
Hardware: Jetson Orin Nano

TABLE OF CONTENTS

Introduction.......... 3
Objectives 3
Functional/Capability Requirements 4
Machine Vision Requirements.......... 5
Functional Design Requirements.......... 6
Results and Analysis.......... 8
Conclusion 15
References.......... 16
Appendices.......... 16

LIST OF FIGURES

Figure 1 : Lane Detection Algorithm 6
Figure 2 : Haar Cascade Implementation 6
Figure 3 : Yolo implementation flow diagram 7
Figure 4 : Scheduler Flow 7
Figure 5 : Lane detection imshow and fps 8
Figure 6 : Lane detection fps without imshow 8
Figure 7 : CUDA Lane detection imshow and fps 9
Figure 8 : CUDA Lane Detection fps without imshow 9
Figure 9 : Haar Cascade true positive imshow and fps 9
Figure 10 : Haar Cascade false positive 10
Figure 11 : Haar Cascade fps without imshow 10
Figure 12 : CUDA Haar Cascade imshow and fps 11
Figure 13 : CUDA Haar Cascade fps without imshow 11
Figure 14 : Yolo v4 fps 12
Figure 15 : yolov4-tiny fps 12
Figure 16 : CUDNN yolov4 fps 13
Figure 17 : CUDNN yolov4-tiny fps 13
Figure 18 : Lane detection & CUDNN Yolov4-tiny Output 14
Figure 19 : Final Output 15

Introduction

Significant progress has been made in the field of computer vision in recent years, especially in the area of autonomous driving and Advanced Driver Assistance Systems (ADAS). These innovations are meant to advance driver comfort, increase vehicle safety, and lay the path for fully autonomous cars. These advancements are made possible by embedded computer vision, which allows for real-time processing and decision-making within the limitations of the systems installed inside the vehicle.

The creation of a dependable system for vehicle and lane detection was the main goal of this project. The method for lane detection required combining the probabilistic Hough line transform with Canny edge detection. This approach, which is covered in greater detail in the paper that goes with it, worked well for determining lane boundaries.

The first attempt used Haar Cascade classifiers for vehicle detection. But it soon became apparent that the accuracy was inadequate, and it was difficult to locate well-trained models online. YOLOv4 was used to solve this problem, which greatly increased detection accuracy. Unfortunately, the method is not practicable for real-time applications due to the excessively low frame rate caused by the significant computational burden on the CPU.

In order to address this problem, the approach shifted to YOLOv4-tiny, a scaled-down model intended for greater frames per second (FPS). Even if this was an improvement, the needed responsiveness was still not achieved. As a result, CUDNN (CUDA Deep Neural Network library) was included to expedite processing even more and improve the system's overall responsiveness and frame rate.

Comparing the performance and outcomes of these implementations with those obtained using OpenCV's CUDA implementation was also a major goal of this study. The purpose of this study was to evaluate the performance and efficacy of GPU-accelerated processing for real-time vehicle and lane recognition tasks.

Objectives

1. Implementation of lane detection and vehicle detection
2. Comparison of these algorithms with OpenCV’s CUDA implementation.

Functional/Capability Requirements

Goal	Feature 1	Feature 2	Performance	Reliability
Minimum	Lane Detection	OpenCV CUDA basic algorithms comparison (canny, bgr to gray scale and Probabilistic Hough line)	8 frames/ Seconds	70%
Target	Steering based on center of the road with Incoming vehicle detection and distance estimation.	Implementing Lane detection algorithm using OpenCV's CUDA extension	12 frames/second	80%
Optimal	Lane detection and steering with broken center line/curves and generate warnings for braking based on vehicle detection.	Adaptation of the complete algorithm (lane detection, vehicle detection and warnings) using OpenCV's CUDA	20+ frames/second	90%+

Minimal Objective:
The bare minimum goal was achieved using the probabilistic Hough line algorithm and canny edge detection to develop a simple lane detection function. The basic algorithm was also implemented using OpenCV's CUDA extension.

Target Objective
Incoming vehicle detection was achieved by using Haar cascade. The same was implemented using OpenCV's CUDA extension.

Optimal Objective
The performance and reliability goals were achieved using both OpenCV and OpenCV's CUDA for lane detection. But Haar Cascade had around 5% of false positive rate, due to which alternate route of Yolo implementation was taken. By using the lane detection algorithm with yolo and CUDNN for car/truck detection, optimal goal was achieved.

Extra Implementation:
The yolov4 implementation was really slow due to which the yolov4-tiny was tested which gave faster frame rate as compared to yolov4 but was still not enough. Due to this, CUDNN was used to achieve the desired framerate and reliability.

Note:
Even though the CUDA implementation is straight forward, the OpenCV has to be reinstalled which took lot of effort and permutations of the CMAKE commands. This is the final configuration used for installation:

```
cmake -D CMAKE_BUILD_TYPE=Release      -D CMAKE_INSTALL_PREFIX=/usr/local
-D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib/modules      -D
WITH_CUDA=ON      -D WITH_CUDNN=ON      -D OPENCV_DNN_CUDA=ON      -D
CUDA_ARCH_BIN=8.7 \  -D CUDA_ARCH_PTX=8.7 \ -D WITH_TBB=ON \ -D
WITH_EIGEN=OFF \ -D WITH_V4L=ON \ -D BUILD_TIFF=ON \ -D
WITH_FFMPEG=ON \ -D WITH_GSTREAMER=ON \ -D BUILD_TESTS=OFF \ -D
BUILD_PERF_TESTS=OFF \ -D BUILD_opencv_python2=OFF \ -D
BUILD_opencv_python3=ON \ -D BUILD_EXAMPLES=OFF -D WITH_CUBLAS=ON -D
ENABLE_FAST_MATH=ON ENABLE_NEON=ON -D WITH_OPENMP=ON -D
WITH_OPENCL=OFF -D BUILD_TBB=ON -D WITH_LIBV4L=ON -D
OPENCV_ENABLE_NONFREE=ON -D OPENCV_GENERATE_PKGCONFIG=ON -D
WITH_GTK=ON -D WITH_OPENGL=ON -S ~/opencv/ ..
```

Machine Vision Requirements

Canny Edge Detection
Canny edge detection is an edge detection algorithm that identifies the boundaries of objects within images. It functions in multiple stages: edge tracking via hysteresis, which uses dual thresholds to identify possible edges, gradient computation to detect edge intensity and direction, and non-maximum suppression to thin out the edges. The Gaussian filter is used for noise reduction.

Probabilistic Hough Line
The Probabilistic Hough Line Transform is a method for detecting lines in an image. It is more computationally efficient than the typical Hough Transform since it randomly samples a fraction of edge points to identify lines. The endpoint coordinates are obtained for detected lines and utilized to draw the lines.

Lane Detection
Lane detection is the process of identifying lane boundaries on roads in images or video streams, commonly used in autonomous driving. Usually, it starts with preprocessing operations such as edge detection and uses methods such as the Hough Line Transform to find and highlight lane markers.

Haar Cascade
Using Haar-like features, Haar Cascade is a machine learning-based object detection method that can identify faces and cars in images. In order to assess whether the window contains the object of interest, a window is slid across the image, and a classifier is used. Cascade steps are used to improve the efficiency and accuracy of detection.

YOLO (You Only Look Once)
YOLO is a real-time object detection algorithm that divides an image into a grid and predicts bounding boxes and class probabilities for objects within each grid cell. Because YOLO examines the full image in a single pass, unlike other detectors, it is incredibly quick and appropriate for real-time applications like as autonomous driving.

OpenCV's CUDA
Through the use of NVIDIA's CUDA API, OpenCV's CUDA extension speeds up image and video processing by shifting processing duties to the GPU. This feature is perfect for real-time applications that demand high computational performance since it enables significant speed improvements in tasks like object recognition, edge detection, and filtering.

Functional Design Requirements

Firstly, lane detection and Haar cascade were implemented using OpenCV and OpenCV's CUDA extension. The flow diagram for both the algorithms are attached below:

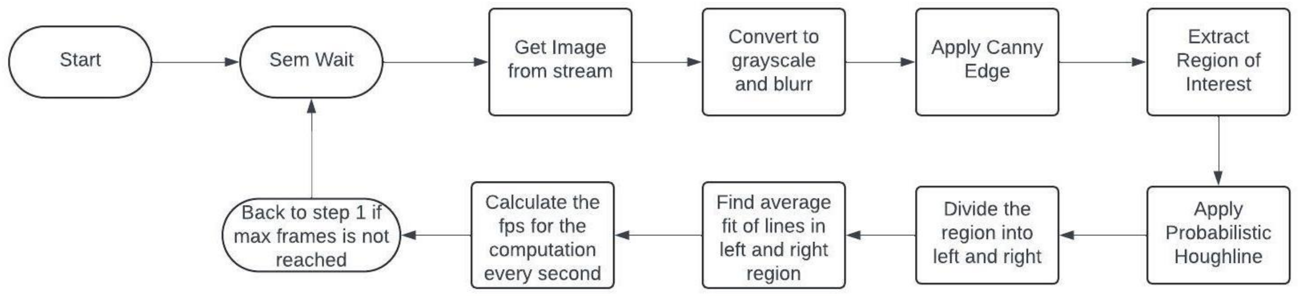

Start
Sem Wait
Get Image from stream
Convert to grayscale and blurr
Apply Canny Edge
Extract Region of Interest
Apply Probabilistic Houghline
Divide the region into left and right
Find average fit of lines in left and right region
Calculate the fps for the computation every second
Back to step 1 if max frames is not reached

Figure 1 : Lane Detection Algorithm

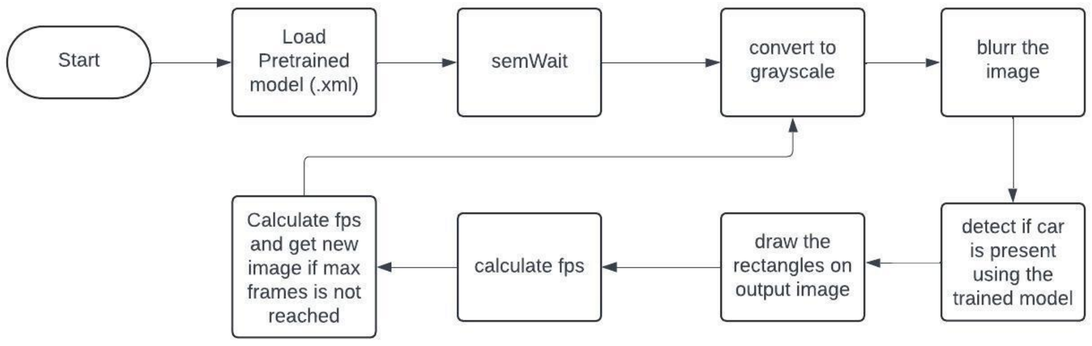

Start
Load Pretrained model (.xml)
semWait
convert to grayscale
blurr the image
detect if car is present using the trained model
draw the rectangles on output image
calculate fps
Calculate fps and get new image if max frames is not reached

Figure 2 : Haar Cascade Implementation

As the Haar Cascade had around 5% False Positive rate, yolov4 was used. The implementation flow diagram is attached below:

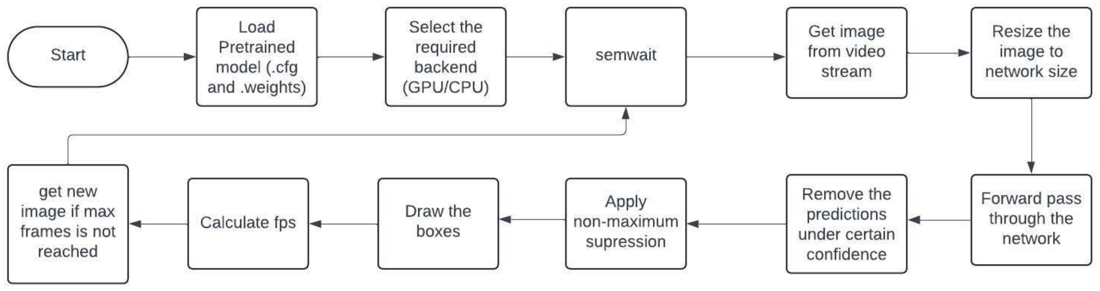

Start
Load Pretrained model (.cfg and .weights)
Select the required backend (GPU/CPU)
semwait
Get image from video stream
Resize the image to network size
Forward pass through the network
Remove the predictions under certain confidence
Apply non-maximum supression
Draw the boxes
Calculate fps
get new image if max frames is not reached

Figure 3 : Yolo implementation flow diagram

Three versions of the codes were prepared for comparison and real-time implementation. Initial goal was to run the codes on same core and use RM scheduling, but because of the fps bottleneck it was decided to use separate cores for each thread. Thus, the thread scheduling is straightforward was shown below.

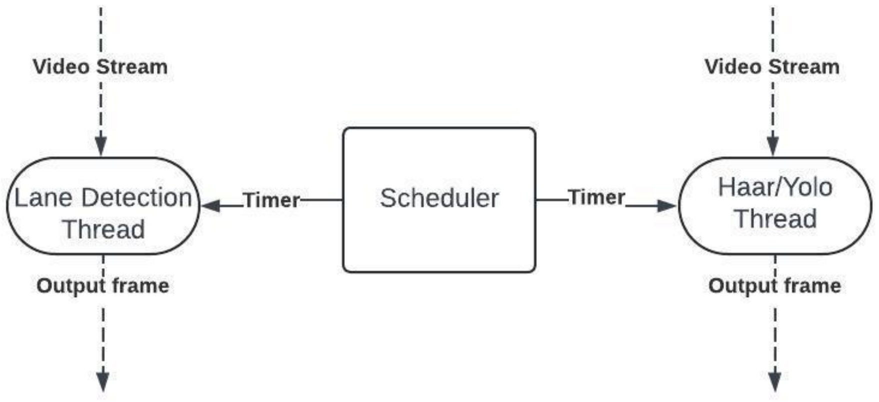


Figure 4 : Scheduler Flow

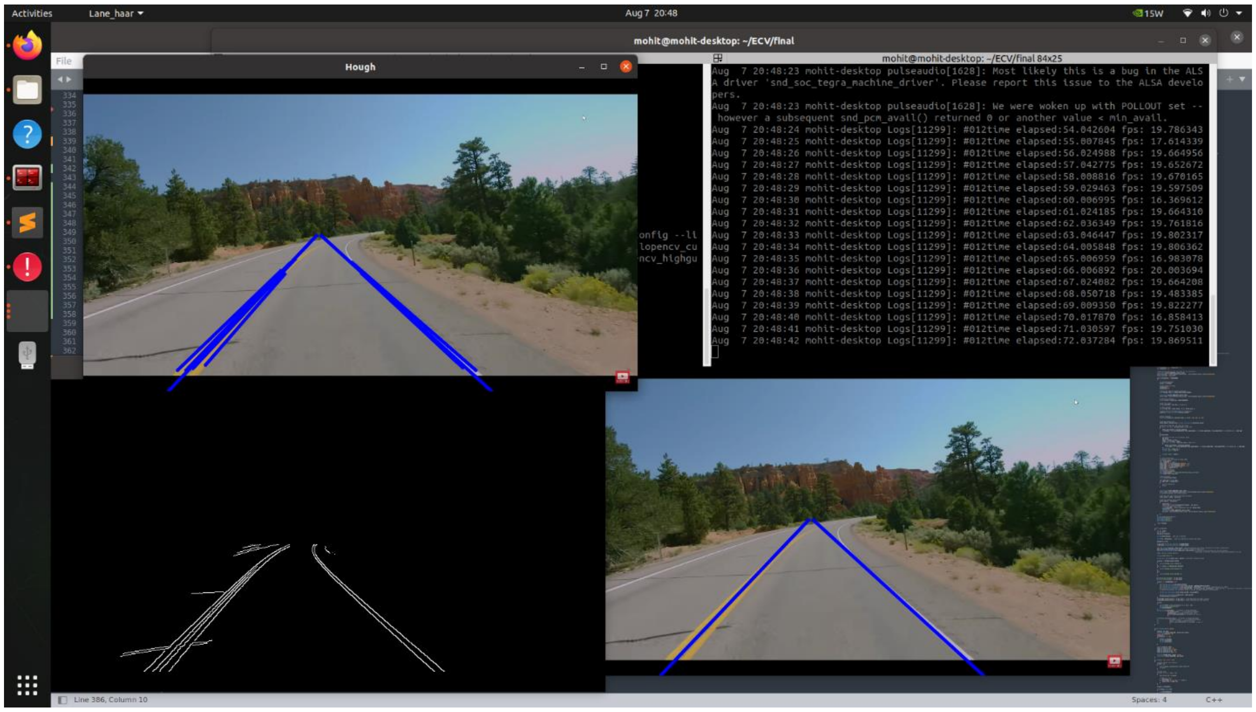
Three Implementations:

1. Lane Detection and Haar Cascade
2. CUDA Lane Detection and Haar Cascade
3. Lane Detection and CUDNN yolov4 tiny

All of the programs generate frames in prespecified location which are then encoded into a video using ffmpeg.

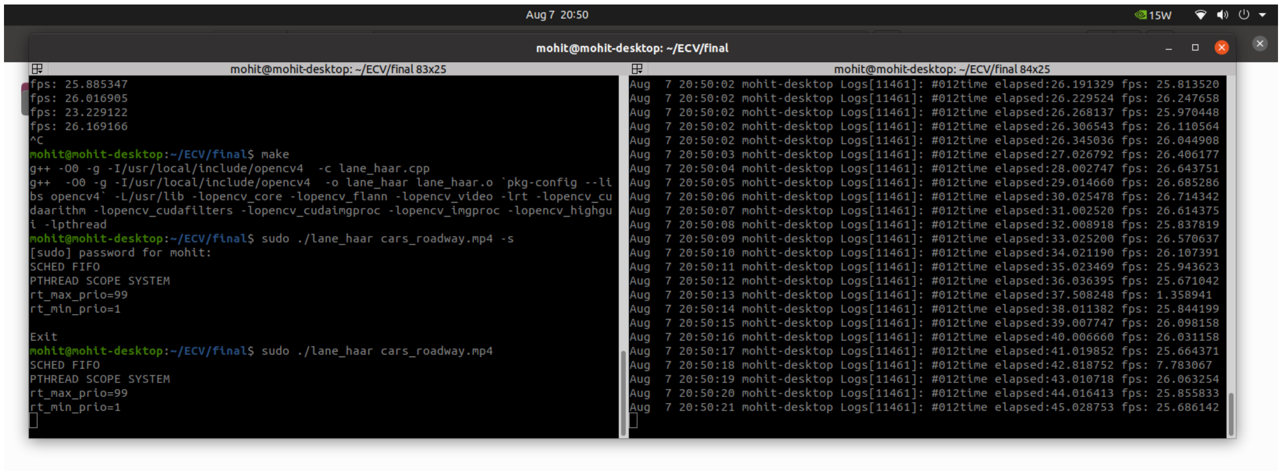
Results and Analysis

Lane detection:


Activities
Lane_haar
Aug 7 20:48
15W
mohit@mohit-desktop: ~/ECV/final
Hough
mohit@mohit-desktop: ~/ECV/final 84x25
Line 386, Column 10
Spaces: 4
C++

```
Aug  7 20:48:23 mohit-desktop pulseaudio[1628]: Most likely this is a bug in the ALS
A driver 'snd_soc_tegra_machine_driver'. Please report this issue to the ALSA develo
pers.
Aug  7 20:48:23 mohit-desktop pulseaudio[1628]: We were woken up with POLLOUT set --
 however a subsequent snd_pcm_avail() returned 0 or another value < min_avail.
Aug  7 20:48:24 mohit-desktop Logs[11299]: #012time elapsed:54.042604 fps: 19.786343
Aug  7 20:48:25 mohit-desktop Logs[11299]: #012time elapsed:55.007845 fps: 17.614339
Aug  7 20:48:26 mohit-desktop Logs[11299]: #012time elapsed:56.024988 fps: 19.664956
Aug  7 20:48:27 mohit-desktop Logs[11299]: #012time elapsed:57.042775 fps: 19.652672
Aug  7 20:48:28 mohit-desktop Logs[11299]: #012time elapsed:58.008816 fps: 19.670165
Aug  7 20:48:29 mohit-desktop Logs[11299]: #012time elapsed:59.029463 fps: 19.597509
Aug  7 20:48:30 mohit-desktop Logs[11299]: #012time elapsed:60.006995 fps: 16.369612
Aug  7 20:48:31 mohit-desktop Logs[11299]: #012time elapsed:61.024185 fps: 19.664310
Aug  7 20:48:32 mohit-desktop Logs[11299]: #012time elapsed:62.036349 fps: 19.761816
Aug  7 20:48:33 mohit-desktop Logs[11299]: #012time elapsed:63.046447 fps: 19.802317
Aug  7 20:48:34 mohit-desktop Logs[11299]: #012time elapsed:64.005848 fps: 19.806362
Aug  7 20:48:35 mohit-desktop Logs[11299]: #012time elapsed:65.006959 fps: 16.983078
Aug  7 20:48:36 mohit-desktop Logs[11299]: #012time elapsed:66.006892 fps: 20.003694
Aug  7 20:48:37 mohit-desktop Logs[11299]: #012time elapsed:67.024082 fps: 19.664208
Aug  7 20:48:38 mohit-desktop Logs[11299]: #012time elapsed:68.050718 fps: 19.483385
Aug  7 20:48:39 mohit-desktop Logs[11299]: #012time elapsed:69.009350 fps: 19.822277
Aug  7 20:48:40 mohit-desktop Logs[11299]: #012time elapsed:70.017870 fps: 16.858413
Aug  7 20:48:41 mohit-desktop Logs[11299]: #012time elapsed:71.030597 fps: 19.751030
Aug  7 20:48:42 mohit-desktop Logs[11299]: #012time elapsed:72.037284 fps: 19.869511
```

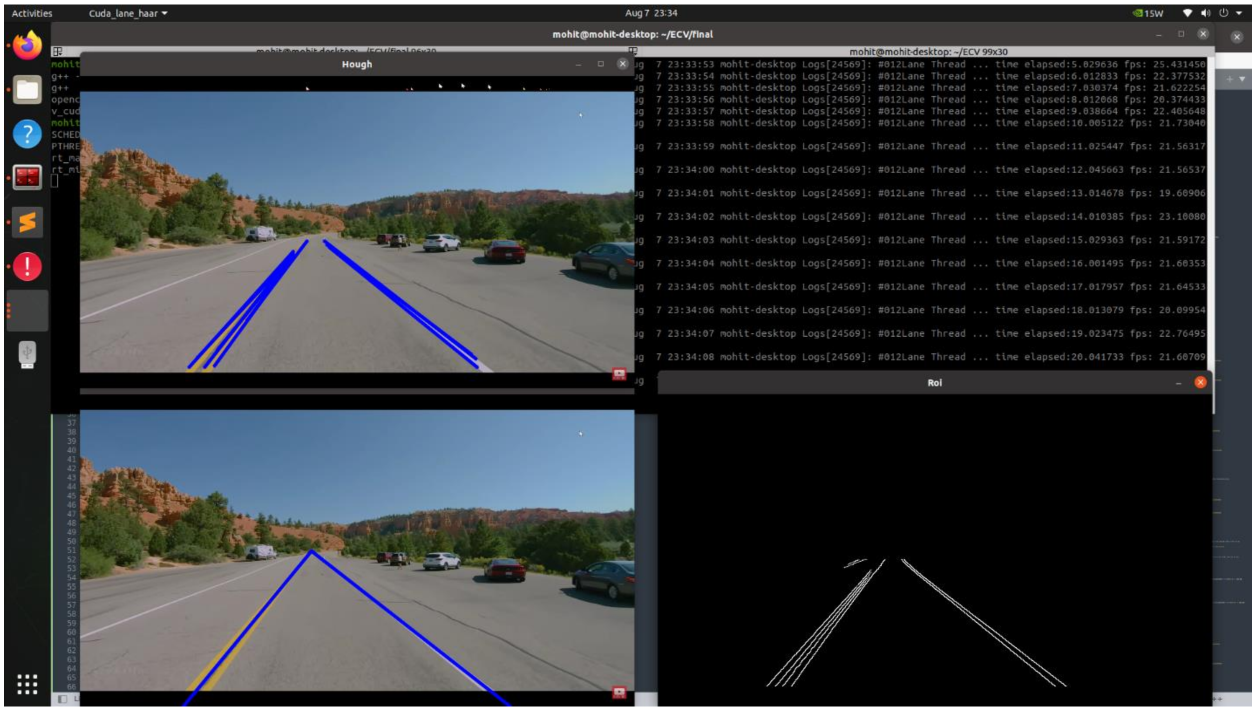
Figure 5 : Lane detection imshow and fps


Aug 7 20:50
15W
mohit@mohit-desktop: ~/ECV/final
mohit@mohit-desktop: ~/ECV/final 83x25
mohit@mohit-desktop: ~/ECV/final 84x25

```
fps: 25.885347
fps: 26.016905
fps: 23.229122
fps: 26.169166
^C
mohit@mohit-desktop:~/ECV/final$ make
g++ -O0 -g -I/usr/local/include/opencv4  -c lane_haar.cpp
g++  -O0 -g -I/usr/local/include/opencv4  -o lane_haar lane_haar.o `pkg-config --li
bs opencv4` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video -lrt -lopencv_cu
daarithm -lopencv_cudafilters -lopencv_cudaimgproc -lopencv_imgproc -lopencv_highgu
i -lpthread
mohit@mohit-desktop:~/ECV/final$ sudo ./lane_haar cars_roadway.mp4 -s
[sudo] password for mohit:
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1

Exit
mohit@mohit-desktop:~/ECV/final$ sudo ./lane_haar cars_roadway.mp4
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
```

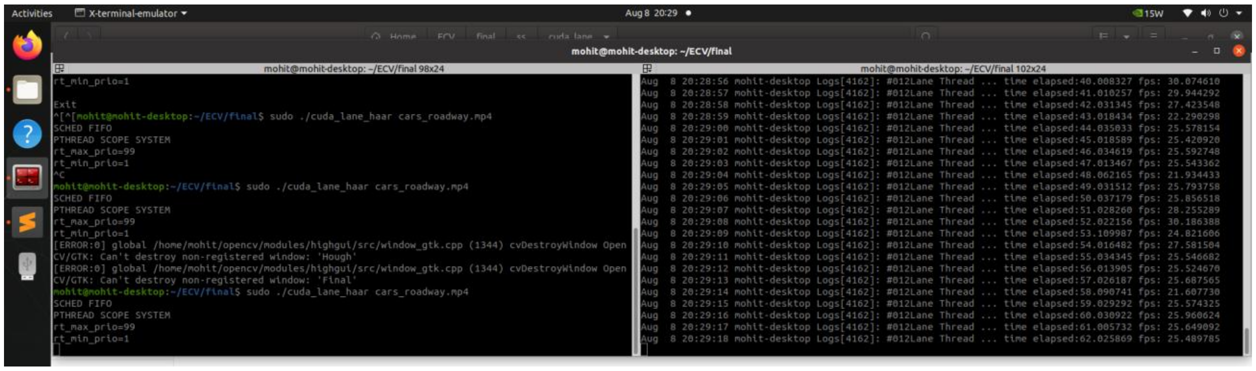
```
Aug  7 20:50:02 mohit-desktop Logs[11461]: #012time elapsed:26.191329 fps: 25.813520
Aug  7 20:50:02 mohit-desktop Logs[11461]: #012time elapsed:26.229524 fps: 26.247658
Aug  7 20:50:02 mohit-desktop Logs[11461]: #012time elapsed:26.268137 fps: 25.970448
Aug  7 20:50:02 mohit-desktop Logs[11461]: #012time elapsed:26.306543 fps: 26.110564
Aug  7 20:50:02 mohit-desktop Logs[11461]: #012time elapsed:26.345036 fps: 26.044908
Aug  7 20:50:03 mohit-desktop Logs[11461]: #012time elapsed:27.026792 fps: 26.406177
Aug  7 20:50:04 mohit-desktop Logs[11461]: #012time elapsed:28.002747 fps: 26.643751
Aug  7 20:50:05 mohit-desktop Logs[11461]: #012time elapsed:29.014660 fps: 26.685286
Aug  7 20:50:06 mohit-desktop Logs[11461]: #012time elapsed:30.025478 fps: 26.714342
Aug  7 20:50:07 mohit-desktop Logs[11461]: #012time elapsed:31.002520 fps: 26.614375
Aug  7 20:50:08 mohit-desktop Logs[11461]: #012time elapsed:32.008918 fps: 25.837819
Aug  7 20:50:09 mohit-desktop Logs[11461]: #012time elapsed:33.025200 fps: 26.570637
Aug  7 20:50:10 mohit-desktop Logs[11461]: #012time elapsed:34.021190 fps: 26.107391
Aug  7 20:50:11 mohit-desktop Logs[11461]: #012time elapsed:35.023469 fps: 25.943623
Aug  7 20:50:12 mohit-desktop Logs[11461]: #012time elapsed:36.036395 fps: 25.671042
Aug  7 20:50:13 mohit-desktop Logs[11461]: #012time elapsed:37.508248 fps: 1.358941
Aug  7 20:50:14 mohit-desktop Logs[11461]: #012time elapsed:38.011382 fps: 25.844199
Aug  7 20:50:15 mohit-desktop Logs[11461]: #012time elapsed:39.007747 fps: 26.098158
Aug  7 20:50:16 mohit-desktop Logs[11461]: #012time elapsed:40.006660 fps: 26.031158
Aug  7 20:50:17 mohit-desktop Logs[11461]: #012time elapsed:41.019852 fps: 25.664371
Aug  7 20:50:18 mohit-desktop Logs[11461]: #012time elapsed:42.818752 fps: 7.783067
Aug  7 20:50:19 mohit-desktop Logs[11461]: #012time elapsed:43.010718 fps: 26.063254
Aug  7 20:50:20 mohit-desktop Logs[11461]: #012time elapsed:44.016413 fps: 25.855833
Aug  7 20:50:21 mohit-desktop Logs[11461]: #012time elapsed:45.028753 fps: 25.686142
```

Figure 6 : Lane detection fps without imshow


Activities
Cuda_lane_haar
Aug 7 23:34
mohit@mohit-desktop: ~/ECV/final
Hough
Roi
mohit@mohit-desktop: ~/ECV 99x30

```
ug  7 23:33:53 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:5.029636 fps: 25.431450
ug  7 23:33:54 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:6.012833 fps: 22.377532
ug  7 23:33:55 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:7.030374 fps: 21.622254
ug  7 23:33:56 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:8.012068 fps: 20.374433
ug  7 23:33:57 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:9.038664 fps: 22.405648
ug  7 23:33:58 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:10.005122 fps: 21.73040

ug  7 23:33:59 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:11.025447 fps: 21.56317

ug  7 23:34:00 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:12.045663 fps: 21.56537

ug  7 23:34:01 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:13.014678 fps: 19.60906

ug  7 23:34:02 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:14.010385 fps: 23.10080

ug  7 23:34:03 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:15.029363 fps: 21.59172

ug  7 23:34:04 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:16.001495 fps: 21.60353

ug  7 23:34:05 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:17.017957 fps: 21.64533

ug  7 23:34:06 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:18.013079 fps: 20.09954

ug  7 23:34:07 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:19.023475 fps: 22.76495

ug  7 23:34:08 mohit-desktop Logs[24569]: #012Lane Thread ... time elapsed:20.041733 fps: 21.60709
```

Figure 7 : CUDA Lane detection imshow and fps

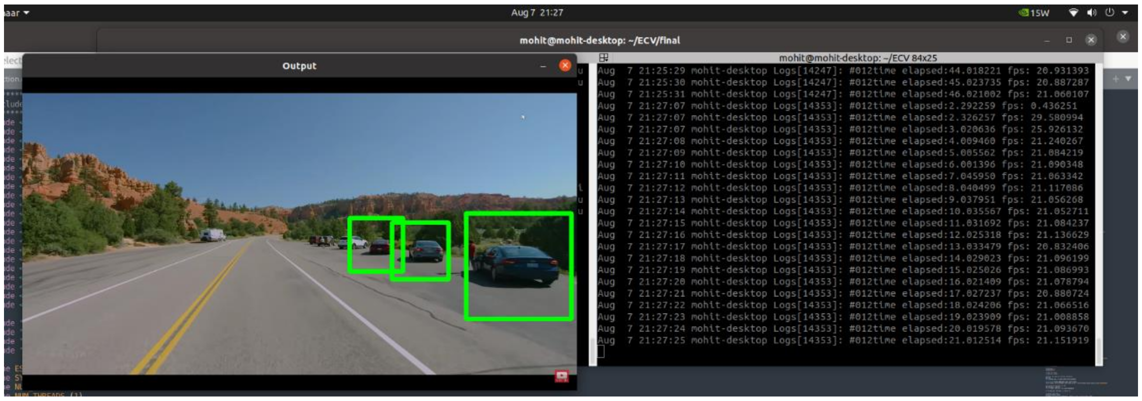

Activities
X-terminal-emulator
Aug 8 20:29
mohit@mohit-desktop: ~/ECV/final
mohit@mohit-desktop: ~/ECV/final 98x24
mohit@mohit-desktop: ~/ECV/final 102x24

```
rt_min_prio=1

Exit
^[^[mohit@mohit-desktop:~/ECV/final$ sudo ./cuda_lane_haar cars_roadway.mp4
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
^C
mohit@mohit-desktop:~/ECV/final$ sudo ./cuda_lane_haar cars_roadway.mp4
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
[ERROR:0] global /home/mohit/opencv/modules/highgui/src/window_gtk.cpp (1344) cvDestroyWindow Open
CV/GTK: Can't destroy non-registered window: 'Hough'
[ERROR:0] global /home/mohit/opencv/modules/highgui/src/window_gtk.cpp (1344) cvDestroyWindow Open
CV/GTK: Can't destroy non-registered window: 'Final'
mohit@mohit-desktop:~/ECV/final$ sudo ./cuda_lane_haar cars_roadway.mp4
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
```

```
Aug  8 20:28:56 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:40.008327 fps: 30.074610
Aug  8 20:28:57 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:41.010257 fps: 29.944292
Aug  8 20:28:58 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:42.031345 fps: 27.423548
Aug  8 20:28:59 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:43.018434 fps: 22.290298
Aug  8 20:29:00 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:44.035033 fps: 25.578154
Aug  8 20:29:01 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:45.018589 fps: 25.420920
Aug  8 20:29:02 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:46.034619 fps: 25.592748
Aug  8 20:29:03 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:47.013467 fps: 25.543362
Aug  8 20:29:04 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:48.062165 fps: 21.934433
Aug  8 20:29:05 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:49.031512 fps: 25.793758
Aug  8 20:29:06 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:50.037179 fps: 25.856518
Aug  8 20:29:07 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:51.028260 fps: 28.255289
Aug  8 20:29:08 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:52.022156 fps: 30.186388
Aug  8 20:29:09 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:53.109987 fps: 24.821606
Aug  8 20:29:10 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:54.016482 fps: 27.581504
Aug  8 20:29:11 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:55.034345 fps: 25.546682
Aug  8 20:29:12 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:56.013905 fps: 25.524670
Aug  8 20:29:13 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:57.026187 fps: 25.687565
Aug  8 20:29:14 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:58.090741 fps: 21.607730
Aug  8 20:29:15 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:59.029292 fps: 25.574325
Aug  8 20:29:16 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:60.030922 fps: 25.960624
Aug  8 20:29:17 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:61.005732 fps: 25.649092
Aug  8 20:29:18 mohit-desktop Logs[4162]: #012Lane Thread ... time elapsed:62.025869 fps: 25.489785
```

Figure 8 : CUDA Lane Detection fps without imshow

Haar Cascade:


Aug 7 21:27
mohit@mohit-desktop: ~/ECV/final
Output
mohit@mohit-desktop: ~/ECV 84x25

```
Aug  7 21:25:29 mohit-desktop Logs[14247]: #012time elapsed:44.018221 fps: 20.931393
Aug  7 21:25:30 mohit-desktop Logs[14247]: #012time elapsed:45.023735 fps: 20.887287
Aug  7 21:25:31 mohit-desktop Logs[14247]: #012time elapsed:46.021002 fps: 21.060107
Aug  7 21:27:07 mohit-desktop Logs[14353]: #012time elapsed:2.292259 fps: 0.436251
Aug  7 21:27:07 mohit-desktop Logs[14353]: #012time elapsed:2.326257 fps: 29.580994
Aug  7 21:27:07 mohit-desktop Logs[14353]: #012time elapsed:3.020636 fps: 25.926132
Aug  7 21:27:08 mohit-desktop Logs[14353]: #012time elapsed:4.009460 fps: 21.240267
Aug  7 21:27:09 mohit-desktop Logs[14353]: #012time elapsed:5.005562 fps: 21.084219
Aug  7 21:27:10 mohit-desktop Logs[14353]: #012time elapsed:6.001396 fps: 21.090348
Aug  7 21:27:11 mohit-desktop Logs[14353]: #012time elapsed:7.045950 fps: 21.063342
Aug  7 21:27:12 mohit-desktop Logs[14353]: #012time elapsed:8.040499 fps: 21.117086
Aug  7 21:27:13 mohit-desktop Logs[14353]: #012time elapsed:9.037951 fps: 21.056268
Aug  7 21:27:14 mohit-desktop Logs[14353]: #012time elapsed:10.035567 fps: 21.052711
Aug  7 21:27:15 mohit-desktop Logs[14353]: #012time elapsed:11.031692 fps: 21.084237
Aug  7 21:27:16 mohit-desktop Logs[14353]: #012time elapsed:12.025318 fps: 21.136629
Aug  7 21:27:17 mohit-desktop Logs[14353]: #012time elapsed:13.033479 fps: 20.832406
Aug  7 21:27:18 mohit-desktop Logs[14353]: #012time elapsed:14.029023 fps: 21.096199
Aug  7 21:27:19 mohit-desktop Logs[14353]: #012time elapsed:15.025026 fps: 21.086993
Aug  7 21:27:20 mohit-desktop Logs[14353]: #012time elapsed:16.021409 fps: 21.078794
Aug  7 21:27:21 mohit-desktop Logs[14353]: #012time elapsed:17.027237 fps: 20.880724
Aug  7 21:27:22 mohit-desktop Logs[14353]: #012time elapsed:18.024206 fps: 21.066516
Aug  7 21:27:23 mohit-desktop Logs[14353]: #012time elapsed:19.023909 fps: 21.008858
Aug  7 21:27:24 mohit-desktop Logs[14353]: #012time elapsed:20.019578 fps: 21.093670
Aug  7 21:27:25 mohit-desktop Logs[14353]: #012time elapsed:21.012514 fps: 21.151919
```

Figure 9 : Haar Cascade true positive imshow and fps

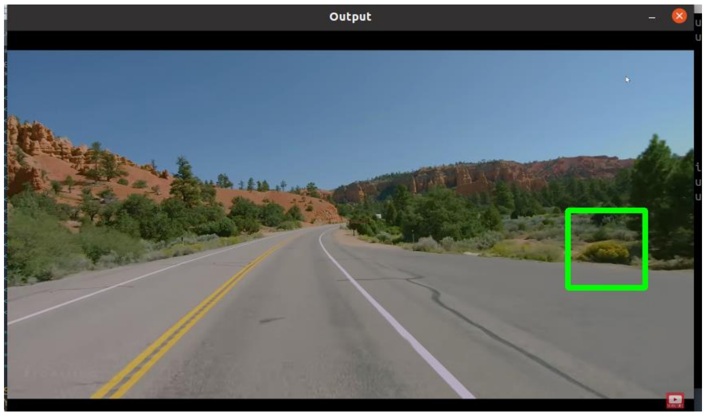
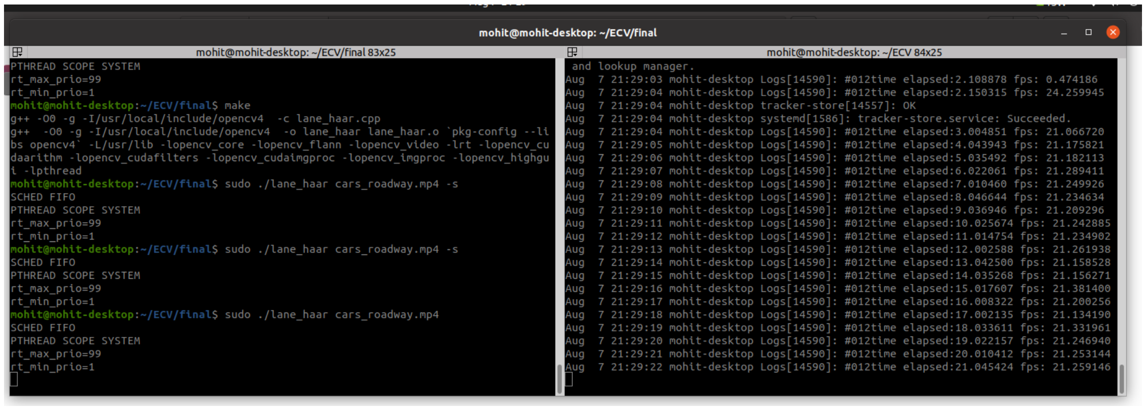


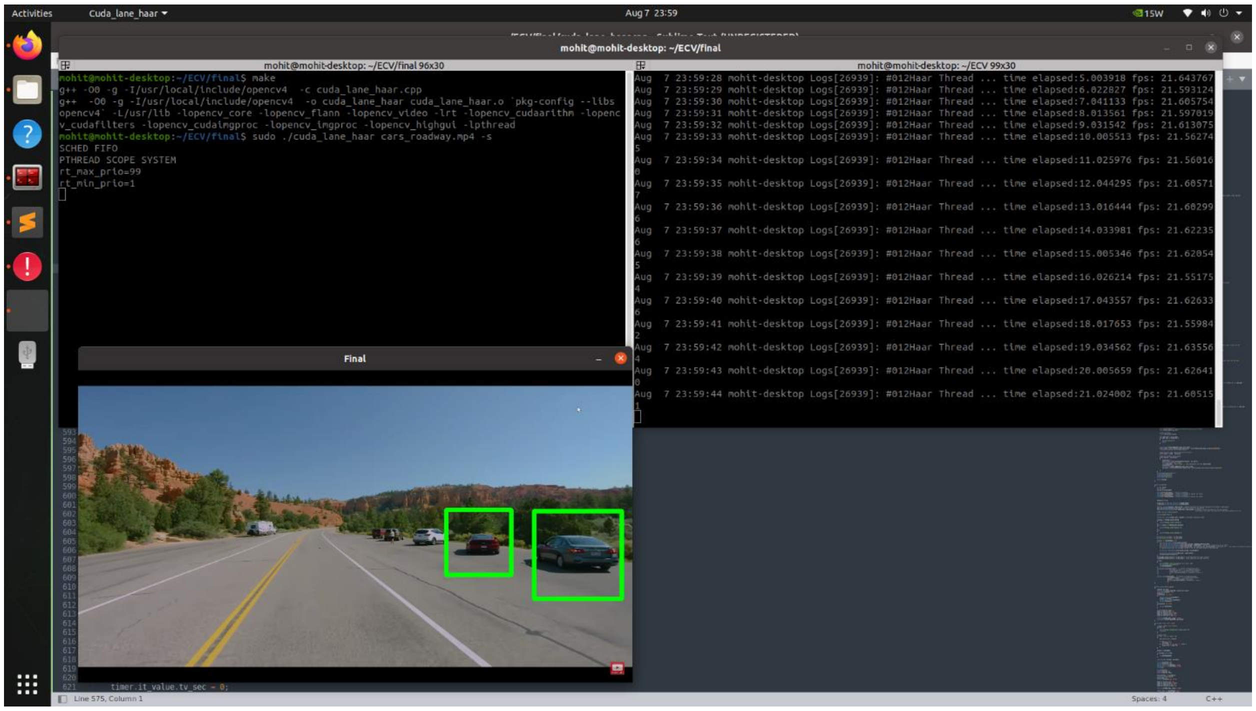
Figure 10 : Haar Cascade false positive


mohit@mohit-desktop: ~/ECV/final
mohit@mohit-desktop: ~/ECV/final 83x25
mohit@mohit-desktop: ~/ECV 84x25

```
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
mohit@mohit-desktop:~/ECV/final$ make
g++ -O0 -g -I/usr/local/include/opencv4  -c lane_haar.cpp
g++  -O0 -g -I/usr/local/include/opencv4  -o lane_haar lane_haar.o `pkg-config --li
bs opencv4` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video -lrt -lopencv_cu
daarithm -lopencv_cudafilters -lopencv_cudaimgproc -lopencv_imgproc -lopencv_highgu
i -lpthread
mohit@mohit-desktop:~/ECV/final$ sudo ./lane_haar cars_roadway.mp4 -s
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
mohit@mohit-desktop:~/ECV/final$ sudo ./lane_haar cars_roadway.mp4 -s
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
mohit@mohit-desktop:~/ECV/final$ sudo ./lane_haar cars_roadway.mp4
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
```

```
 and lookup manager.
Aug  7 21:29:03 mohit-desktop Logs[14590]: #012time elapsed:2.108878 fps: 0.474186
Aug  7 21:29:04 mohit-desktop Logs[14590]: #012time elapsed:2.150315 fps: 24.259945
Aug  7 21:29:04 mohit-desktop tracker-store[14557]: OK
Aug  7 21:29:04 mohit-desktop systemd[1586]: tracker-store.service: Succeeded.
Aug  7 21:29:04 mohit-desktop Logs[14590]: #012time elapsed:3.004851 fps: 21.066720
Aug  7 21:29:05 mohit-desktop Logs[14590]: #012time elapsed:4.043943 fps: 21.175821
Aug  7 21:29:06 mohit-desktop Logs[14590]: #012time elapsed:5.035492 fps: 21.182113
Aug  7 21:29:07 mohit-desktop Logs[14590]: #012time elapsed:6.022061 fps: 21.289411
Aug  7 21:29:08 mohit-desktop Logs[14590]: #012time elapsed:7.010460 fps: 21.249926
Aug  7 21:29:09 mohit-desktop Logs[14590]: #012time elapsed:8.046644 fps: 21.234634
Aug  7 21:29:10 mohit-desktop Logs[14590]: #012time elapsed:9.036946 fps: 21.209296
Aug  7 21:29:11 mohit-desktop Logs[14590]: #012time elapsed:10.025674 fps: 21.242885
Aug  7 21:29:12 mohit-desktop Logs[14590]: #012time elapsed:11.014754 fps: 21.234902
Aug  7 21:29:13 mohit-desktop Logs[14590]: #012time elapsed:12.002588 fps: 21.261938
Aug  7 21:29:14 mohit-desktop Logs[14590]: #012time elapsed:13.042500 fps: 21.158528
Aug  7 21:29:15 mohit-desktop Logs[14590]: #012time elapsed:14.035268 fps: 21.156271
Aug  7 21:29:16 mohit-desktop Logs[14590]: #012time elapsed:15.017607 fps: 21.381400
Aug  7 21:29:17 mohit-desktop Logs[14590]: #012time elapsed:16.008322 fps: 21.200256
Aug  7 21:29:18 mohit-desktop Logs[14590]: #012time elapsed:17.002135 fps: 21.134190
Aug  7 21:29:19 mohit-desktop Logs[14590]: #012time elapsed:18.033611 fps: 21.331961
Aug  7 21:29:20 mohit-desktop Logs[14590]: #012time elapsed:19.022157 fps: 21.246940
Aug  7 21:29:21 mohit-desktop Logs[14590]: #012time elapsed:20.010412 fps: 21.253144
Aug  7 21:29:22 mohit-desktop Logs[14590]: #012time elapsed:21.045424 fps: 21.259146
```

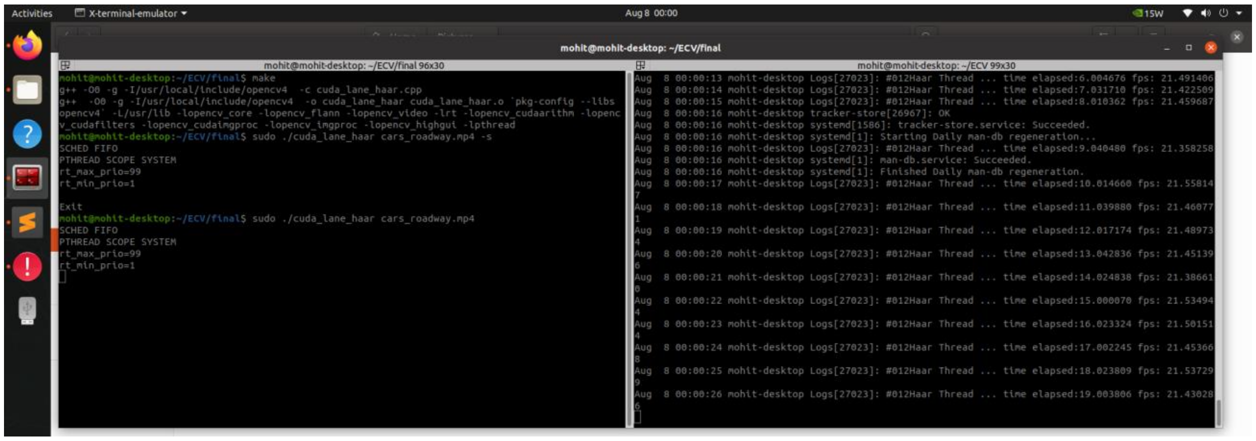
Figure 11 : Haar Cascade fps without imshow



```
mohit@mohit-desktop:~/ECV/final$ make
g++ -O0 -g -I/usr/local/include/opencv4  -c cuda_lane_haar.cpp
g++  -O0 -g -I/usr/local/include/opencv4  -o cuda_lane_haar cuda_lane_haar.o `pkg-config --libs
opencv4` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video -lrt -lopencv_cudaarithm -lopenc
v_cudafilters -lopencv_cudaimgproc -lopencv_imgproc -lopencv_highgui -lpthread
mohit@mohit-desktop:~/ECV/final$ sudo ./cuda_lane_haar cars_roadway.mp4 -s
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
```

```
Aug  7 23:59:28 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:5.003918 fps: 21.643767
Aug  7 23:59:29 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:6.022827 fps: 21.593124
Aug  7 23:59:30 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:7.041133 fps: 21.605754
Aug  7 23:59:31 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:8.013561 fps: 21.597019
Aug  7 23:59:32 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:9.031542 fps: 21.613075
Aug  7 23:59:33 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:10.005513 fps: 21.56274
5
Aug  7 23:59:34 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:11.025976 fps: 21.56016
0
Aug  7 23:59:35 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:12.044295 fps: 21.60571
7
Aug  7 23:59:36 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:13.016444 fps: 21.60299
6
Aug  7 23:59:37 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:14.033981 fps: 21.62235
6
Aug  7 23:59:38 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:15.005346 fps: 21.62054
5
Aug  7 23:59:39 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:16.026214 fps: 21.55175
4
Aug  7 23:59:40 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:17.043557 fps: 21.62633
6
Aug  7 23:59:41 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:18.017653 fps: 21.55984
2
Aug  7 23:59:42 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:19.034562 fps: 21.63556
4
Aug  7 23:59:43 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:20.005659 fps: 21.62641
0
Aug  7 23:59:44 mohit-desktop Logs[26939]: #012Haar Thread ... time elapsed:21.024002 fps: 21.60515
1
```

Figure 12 : CUDA Haar Cascade imshow and fps



```
mohit@mohit-desktop:~/ECV/final$ make
g++ -O0 -g -I/usr/local/include/opencv4  -c cuda_lane_haar.cpp
g++  -O0 -g -I/usr/local/include/opencv4  -o cuda_lane_haar cuda_lane_haar.o `pkg-config --libs
opencv4` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video -lrt -lopencv_cudaarithm -lopenc
v_cudafilters -lopencv_cudaimgproc -lopencv_imgproc -lopencv_highgui -lpthread
mohit@mohit-desktop:~/ECV/final$ sudo ./cuda_lane_haar cars_roadway.mp4 -s
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1

Exit
mohit@mohit-desktop:~/ECV/final$ sudo ./cuda_lane_haar cars_roadway.mp4
SCHED FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
```

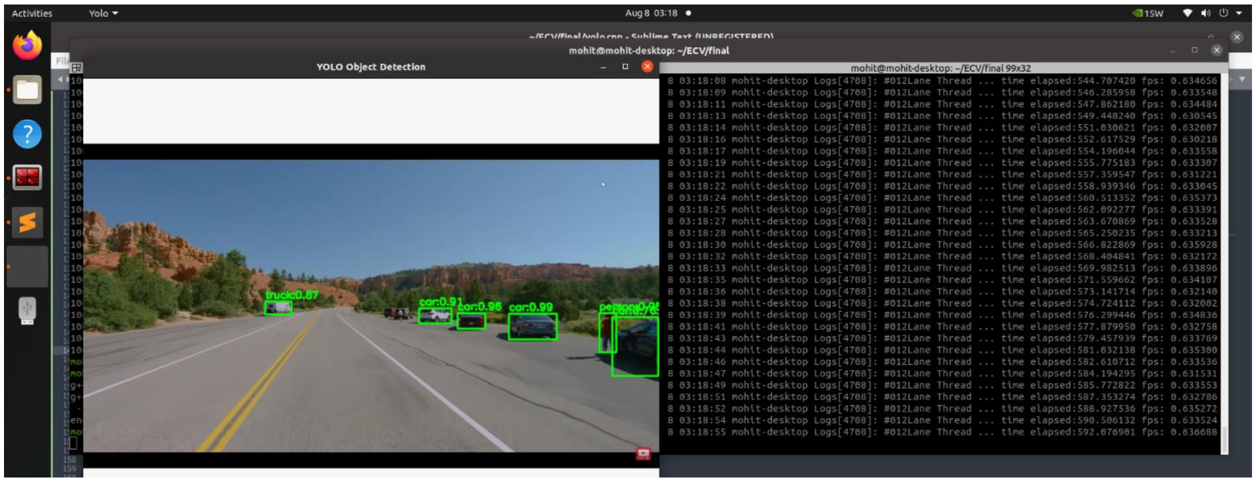
```
Aug  8 00:00:13 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:6.004676 fps: 21.491406
Aug  8 00:00:14 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:7.031710 fps: 21.422509
Aug  8 00:00:15 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:8.010362 fps: 21.459687
Aug  8 00:00:16 mohit-desktop tracker-store[26967]: OK
Aug  8 00:00:16 mohit-desktop systemd[1586]: tracker-store.service: Succeeded.
Aug  8 00:00:16 mohit-desktop systemd[1]: Starting Daily man-db regeneration...
Aug  8 00:00:16 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:9.040480 fps: 21.358258
Aug  8 00:00:16 mohit-desktop systemd[1]: man-db.service: Succeeded.
Aug  8 00:00:16 mohit-desktop systemd[1]: Finished Daily man-db regeneration.
Aug  8 00:00:17 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:10.014660 fps: 21.55814
7
Aug  8 00:00:18 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:11.039880 fps: 21.46077
1
Aug  8 00:00:19 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:12.017174 fps: 21.48973
4
Aug  8 00:00:20 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:13.042836 fps: 21.45139
6
Aug  8 00:00:21 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:14.024838 fps: 21.38661
0
Aug  8 00:00:22 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:15.000070 fps: 21.53494
4
Aug  8 00:00:23 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:16.023324 fps: 21.50151
4
Aug  8 00:00:24 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:17.002245 fps: 21.45366
8
Aug  8 00:00:25 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:18.023809 fps: 21.53729
9
Aug  8 00:00:26 mohit-desktop Logs[27023]: #012Haar Thread ... time elapsed:19.003806 fps: 21.43028
6
```

Figure 13 : CUDA Haar Cascade fps without imshow

	OpenCV (approx. fps)	**OpenCV's CUDA (approx. fps)**
1. Lane Detection	25.9	25.6
2. Haar Cascade	21.2	21.5

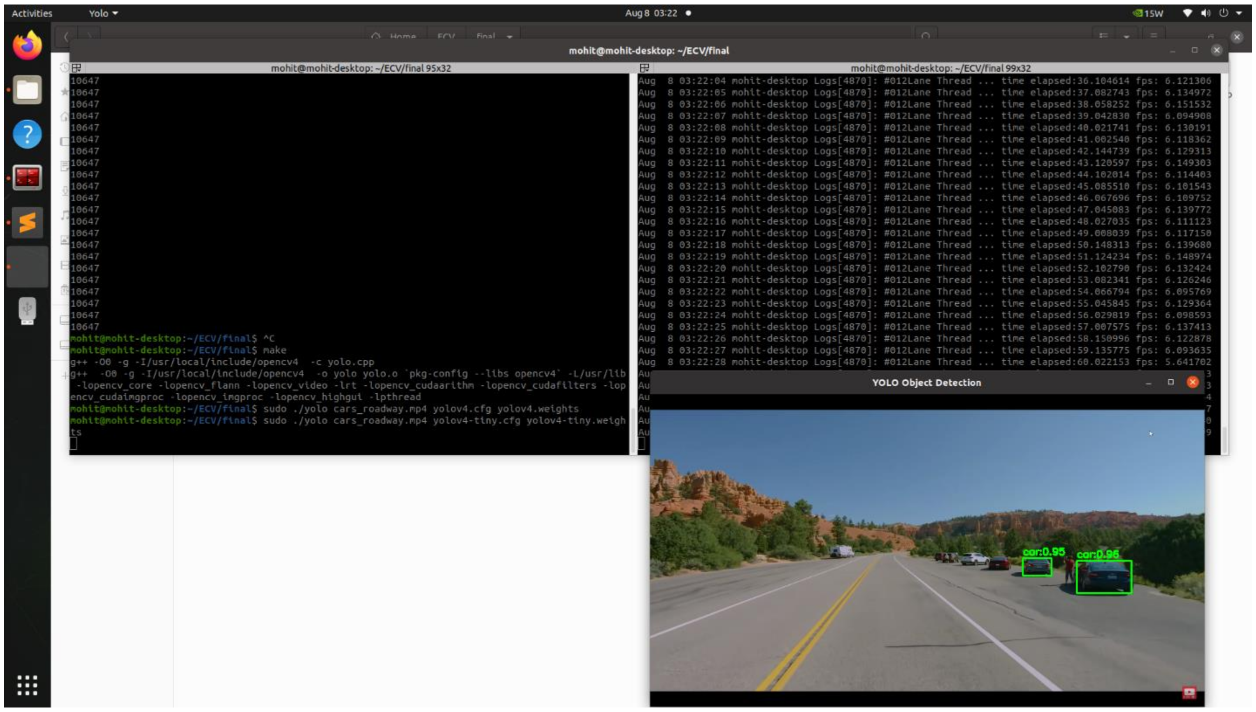
As we can see from the results above OpenCV's CUDA has similar processing speed when it comes to traditional algorithms like Canny edge detection, Hough-Line Probabilistic, BGR to grayscale conversion, Haar Cascade, etc. And is actually a bit slower in some cases, this might be because of couple of reasons, the limited parallelism nature of the traditional algorithms itself or because of slow uploads/downloads to the GPU device.

Yolo:


Activities
Yolo
Aug 8 03:18
15W
~/ECV/final/yolo.cpp - Sublime Text (UNREGISTERED)
mohit@mohit-desktop: ~/ECV/final
YOLO Object Detection
mohit@mohit-desktop: ~/ECV/final 99x32
truck:0.87
car:0.91
car:0.96
car:0.99

```
8 03:18:08 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:544.707420 fps: 0.634656
8 03:18:09 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:546.285958 fps: 0.633548
8 03:18:11 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:547.862180 fps: 0.634484
8 03:18:13 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:549.448240 fps: 0.630545
8 03:18:14 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:551.030621 fps: 0.632007
8 03:18:16 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:552.617529 fps: 0.630218
8 03:18:17 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:554.196044 fps: 0.633558
8 03:18:19 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:555.775183 fps: 0.633307
8 03:18:21 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:557.359547 fps: 0.631221
8 03:18:22 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:558.939346 fps: 0.633045
8 03:18:24 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:560.513352 fps: 0.635373
8 03:18:25 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:562.092277 fps: 0.633391
8 03:18:27 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:563.670869 fps: 0.633528
8 03:18:28 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:565.250235 fps: 0.633213
8 03:18:30 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:566.822869 fps: 0.635928
8 03:18:32 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:568.404841 fps: 0.632172
8 03:18:33 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:569.982513 fps: 0.633896
8 03:18:35 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:571.559662 fps: 0.634107
8 03:18:36 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:573.141714 fps: 0.632140
8 03:18:38 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:574.724112 fps: 0.632002
8 03:18:39 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:576.299446 fps: 0.634836
8 03:18:41 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:577.879950 fps: 0.632758
8 03:18:43 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:579.457939 fps: 0.633769
8 03:18:44 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:581.032138 fps: 0.635300
8 03:18:46 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:582.610712 fps: 0.633536
8 03:18:47 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:584.194295 fps: 0.631531
8 03:18:49 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:585.772822 fps: 0.633553
8 03:18:51 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:587.353274 fps: 0.632786
8 03:18:52 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:588.927536 fps: 0.635272
8 03:18:54 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:590.506132 fps: 0.633524
8 03:18:55 mohit-desktop Logs[4708]: #012Lane Thread ... time elapsed:592.076901 fps: 0.636688
```

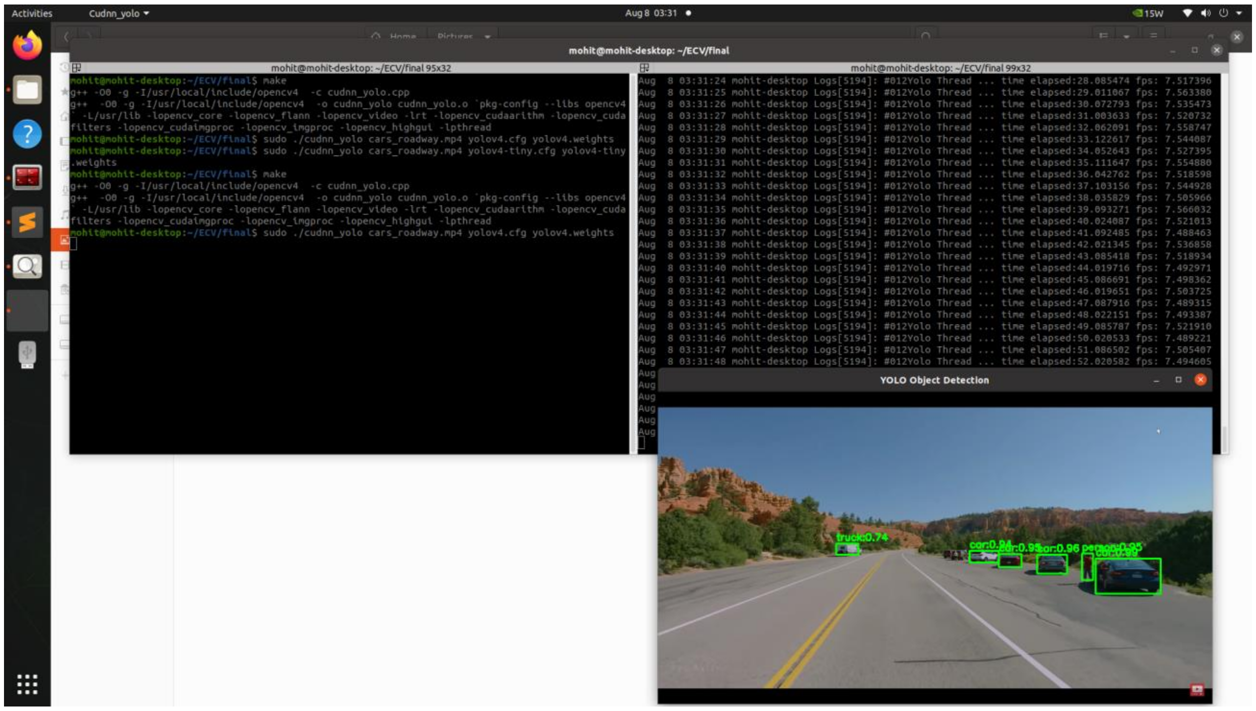
Figure 14 : Yolo v4 fps


Activities
Yolo
Aug 8 03:22
15W
mohit@mohit-desktop: ~/ECV/final
mohit@mohit-desktop: ~/ECV/final 95x32
mohit@mohit-desktop: ~/ECV/final 99x32
YOLO Object Detection
car:0.95
car:0.96

```
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
10647
mohit@mohit-desktop:~/ECV/final$ ^C
mohit@mohit-desktop:~/ECV/final$ make
g++ -O0 -g -I/usr/local/include/opencv4  -c yolo.cpp
g++  -O0 -g -I/usr/local/include/opencv4  -o yolo yolo.o `pkg-config --libs opencv4` -L/usr/lib
 -lopencv_core -lopencv_flann -lopencv_video -lrt -lopencv_cudaarithm -lopencv_cudafilters -lop
encv_cudaimgproc -lopencv_imgproc -lopencv_highgui -lpthread
mohit@mohit-desktop:~/ECV/final$ sudo ./yolo cars_roadway.mp4 yolov4.cfg yolov4.weights
mohit@mohit-desktop:~/ECV/final$ sudo ./yolo cars_roadway.mp4 yolov4-tiny.cfg yolov4-tiny.weigh
ts
```

```
Aug  8 03:22:04 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:36.104614 fps: 6.121306
Aug  8 03:22:05 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:37.082743 fps: 6.134972
Aug  8 03:22:06 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:38.058252 fps: 6.151532
Aug  8 03:22:07 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:39.042830 fps: 6.094908
Aug  8 03:22:08 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:40.021741 fps: 6.130191
Aug  8 03:22:09 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:41.002540 fps: 6.118362
Aug  8 03:22:10 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:42.144739 fps: 6.129313
Aug  8 03:22:11 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:43.120597 fps: 6.149303
Aug  8 03:22:12 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:44.102014 fps: 6.114403
Aug  8 03:22:13 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:45.085510 fps: 6.101543
Aug  8 03:22:14 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:46.067696 fps: 6.109752
Aug  8 03:22:15 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:47.045083 fps: 6.139772
Aug  8 03:22:16 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:48.027035 fps: 6.111123
Aug  8 03:22:17 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:49.008039 fps: 6.117150
Aug  8 03:22:18 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:50.148313 fps: 6.139680
Aug  8 03:22:19 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:51.124234 fps: 6.148974
Aug  8 03:22:20 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:52.102790 fps: 6.132424
Aug  8 03:22:21 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:53.082341 fps: 6.126246
Aug  8 03:22:22 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:54.066794 fps: 6.095769
Aug  8 03:22:23 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:55.045845 fps: 6.129364
Aug  8 03:22:24 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:56.029819 fps: 6.098593
Aug  8 03:22:25 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:57.007575 fps: 6.137413
Aug  8 03:22:26 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:58.150996 fps: 6.122878
Aug  8 03:22:27 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:59.135775 fps: 6.093635
Aug  8 03:22:28 mohit-desktop Logs[4870]: #012Lane Thread ... time elapsed:60.022153 fps: 5.641702
```

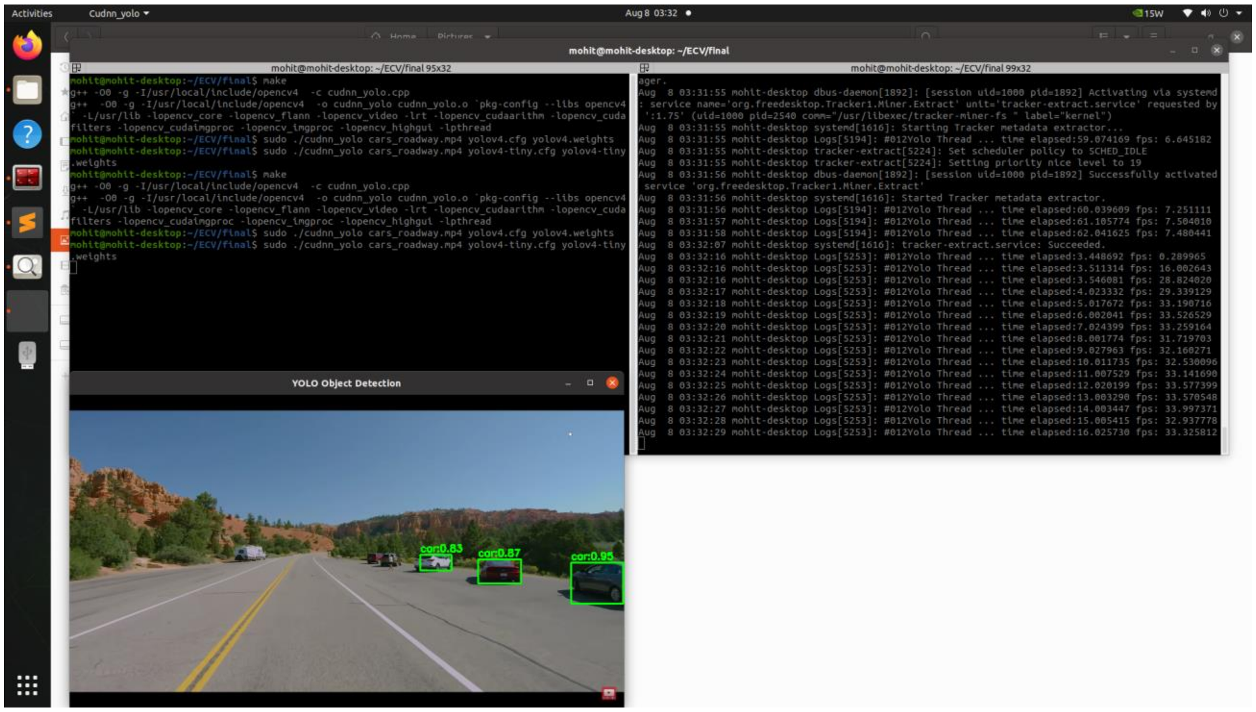
Figure 15 : yolov4-tiny fps



```
mohit@mohit-desktop: ~/ECV/final 95x32
mohit@mohit-desktop:~/ECV/final$ make
g++ -O0 -g -I/usr/local/include/opencv4  -c cudnn_yolo.cpp
g++  -O0 -g -I/usr/local/include/opencv4  -o cudnn_yolo cudnn_yolo.o `pkg-config --libs opencv4
` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video -lrt -lopencv_cudaarithm -lopencv_cuda
filters -lopencv_cudaimgproc -lopencv_imgproc -lopencv_highgui -lpthread
mohit@mohit-desktop:~/ECV/final$ sudo ./cudnn_yolo cars_roadway.mp4 yolov4.cfg yolov4.weights
mohit@mohit-desktop:~/ECV/final$ sudo ./cudnn_yolo cars_roadway.mp4 yolov4-tiny.cfg yolov4-tiny
.weights
mohit@mohit-desktop:~/ECV/final$ make
g++ -O0 -g -I/usr/local/include/opencv4  -c cudnn_yolo.cpp
g++  -O0 -g -I/usr/local/include/opencv4  -o cudnn_yolo cudnn_yolo.o `pkg-config --libs opencv4
` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video -lrt -lopencv_cudaarithm -lopencv_cuda
filters -lopencv_cudaimgproc -lopencv_imgproc -lopencv_highgui -lpthread
mohit@mohit-desktop:~/ECV/final$ sudo ./cudnn_yolo cars_roadway.mp4 yolov4.cfg yolov4.weights
```

```
mohit@mohit-desktop: ~/ECV/final 99x32
Aug  8 03:31:24 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:28.085474 fps: 7.517396
Aug  8 03:31:25 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:29.011067 fps: 7.563380
Aug  8 03:31:26 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:30.072793 fps: 7.535473
Aug  8 03:31:27 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:31.003633 fps: 7.520732
Aug  8 03:31:28 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:32.062091 fps: 7.558747
Aug  8 03:31:29 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:33.122617 fps: 7.544087
Aug  8 03:31:30 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:34.052643 fps: 7.527395
Aug  8 03:31:31 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:35.111647 fps: 7.554880
Aug  8 03:31:32 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:36.042762 fps: 7.518598
Aug  8 03:31:33 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:37.103156 fps: 7.544928
Aug  8 03:31:34 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:38.035829 fps: 7.505966
Aug  8 03:31:35 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:39.093271 fps: 7.566032
Aug  8 03:31:36 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:40.024087 fps: 7.521013
Aug  8 03:31:37 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:41.092485 fps: 7.488463
Aug  8 03:31:38 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:42.021345 fps: 7.536858
Aug  8 03:31:39 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:43.085418 fps: 7.518934
Aug  8 03:31:40 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:44.019716 fps: 7.492971
Aug  8 03:31:41 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:45.086691 fps: 7.498362
Aug  8 03:31:42 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:46.019651 fps: 7.503725
Aug  8 03:31:43 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:47.087916 fps: 7.489315
Aug  8 03:31:44 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:48.022151 fps: 7.493387
Aug  8 03:31:45 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:49.085787 fps: 7.521910
Aug  8 03:31:46 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:50.020533 fps: 7.489221
Aug  8 03:31:47 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:51.086502 fps: 7.505407
Aug  8 03:31:48 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:52.020582 fps: 7.494605
Aug
Aug
Aug
Aug
Aug
Aug
```

Figure 16 : CUDNN yolov4 fps



```
mohit@mohit-desktop: ~/ECV/final 95x32
mohit@mohit-desktop:~/ECV/final$ make
g++ -O0 -g -I/usr/local/include/opencv4  -c cudnn_yolo.cpp
g++  -O0 -g -I/usr/local/include/opencv4  -o cudnn_yolo cudnn_yolo.o `pkg-config --libs opencv4
` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video -lrt -lopencv_cudaarithm -lopencv_cuda
filters -lopencv_cudaimgproc -lopencv_imgproc -lopencv_highgui -lpthread
mohit@mohit-desktop:~/ECV/final$ sudo ./cudnn_yolo cars_roadway.mp4 yolov4.cfg yolov4.weights
mohit@mohit-desktop:~/ECV/final$ sudo ./cudnn_yolo cars_roadway.mp4 yolov4-tiny.cfg yolov4-tiny
.weights
mohit@mohit-desktop:~/ECV/final$ make
g++ -O0 -g -I/usr/local/include/opencv4  -c cudnn_yolo.cpp
g++  -O0 -g -I/usr/local/include/opencv4  -o cudnn_yolo cudnn_yolo.o `pkg-config --libs opencv4
` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video -lrt -lopencv_cudaarithm -lopencv_cuda
filters -lopencv_cudaimgproc -lopencv_imgproc -lopencv_highgui -lpthread
mohit@mohit-desktop:~/ECV/final$ sudo ./cudnn_yolo cars_roadway.mp4 yolov4.cfg yolov4.weights
mohit@mohit-desktop:~/ECV/final$ sudo ./cudnn_yolo cars_roadway.mp4 yolov4-tiny.cfg yolov4-tiny
.weights
```

```
mohit@mohit-desktop: ~/ECV/final 99x32
ager.
Aug  8 03:31:55 mohit-desktop dbus-daemon[1892]: [session uid=1000 pid=1892] Activating via systemd
: service name='org.freedesktop.Tracker1.Miner.Extract' unit='tracker-extract.service' requested by
 ':1.75' (uid=1000 pid=2540 comm="/usr/libexec/tracker-miner-fs " label="kernel")
Aug  8 03:31:55 mohit-desktop systemd[1616]: Starting Tracker metadata extractor...
Aug  8 03:31:55 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:59.074169 fps: 6.645182
Aug  8 03:31:55 mohit-desktop tracker-extract[5224]: Set scheduler policy to SCHED_IDLE
Aug  8 03:31:55 mohit-desktop tracker-extract[5224]: Setting priority nice level to 19
Aug  8 03:31:56 mohit-desktop dbus-daemon[1892]: [session uid=1000 pid=1892] Successfully activated
 service 'org.freedesktop.Tracker1.Miner.Extract'
Aug  8 03:31:56 mohit-desktop systemd[1616]: Started Tracker metadata extractor.
Aug  8 03:31:56 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:60.039609 fps: 7.251111
Aug  8 03:31:57 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:61.105774 fps: 7.504010
Aug  8 03:31:58 mohit-desktop Logs[5194]: #012Yolo Thread ... time elapsed:62.041625 fps: 7.480441
Aug  8 03:32:07 mohit-desktop systemd[1616]: tracker-extract.service: Succeeded.
Aug  8 03:32:16 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:3.448692 fps: 0.289965
Aug  8 03:32:16 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:3.511314 fps: 16.002643
Aug  8 03:32:16 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:3.546081 fps: 28.824020
Aug  8 03:32:17 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:4.023332 fps: 29.339129
Aug  8 03:32:18 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:5.017672 fps: 33.190716
Aug  8 03:32:19 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:6.002041 fps: 33.526529
Aug  8 03:32:20 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:7.024399 fps: 33.259164
Aug  8 03:32:21 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:8.001774 fps: 31.719703
Aug  8 03:32:22 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:9.027963 fps: 32.160271
Aug  8 03:32:23 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:10.011735 fps: 32.530096
Aug  8 03:32:24 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:11.007529 fps: 33.141690
Aug  8 03:32:25 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:12.020199 fps: 33.577399
Aug  8 03:32:26 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:13.003290 fps: 33.570548
Aug  8 03:32:27 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:14.003447 fps: 33.997371
Aug  8 03:32:28 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:15.005415 fps: 32.937778
Aug  8 03:32:29 mohit-desktop Logs[5253]: #012Yolo Thread ... time elapsed:16.025730 fps: 33.325812
```

Figure 17 : CUDNN yolov4-tiny fps

	Yolov4 (approx. fps)	Yolov4-tiny (approx. fps)
1. CPU Backend	0.63	6.1
2. GPU Backend (CUDNN)	7.5	33.5

As we can see from the results above, in case of yolo we get much better performance when using GPU in the backend. This is also more accurate when compared to Haar-Cascade which had a high false positive rate. The only draw back is that yolov4-tiny has limited capability while detecting vehicles which are far away.

Combined Versions Comparisons:

	Lane Detection (approx Fps)	**Car Detection (approx. fps)**
1. Lane detection & Haar Cascade	**19.9**	**19.9**
2. CUDA lane detect & CUDA Haar	**24.9**	**19.9**
3. Lane detection & CUDNN yolov4-tiny	**24.9**	**24.9**

As we can see from the implementation, the third program which is combination of CPU based lane detection and GPU based car detection gives the best results when it comes to performance. Also, Haar-cascade pretrained models had around 5% false positive rate for this dataset and more tuning and preprocessing might be required for better results.

Haar Cascade:
True Positive rate: 96%
False Positive rate: 5.3%

Yolov4-Tiny:
True Positive rate: 92%
False Positive rate: 0%

Note that the false and true positive rates are approximate and were calculated by seeing the frames manually.

Thus, Yolov4-tiny with lane detection on CPU was used which gave the best performance, which was highly accurate and reliable.

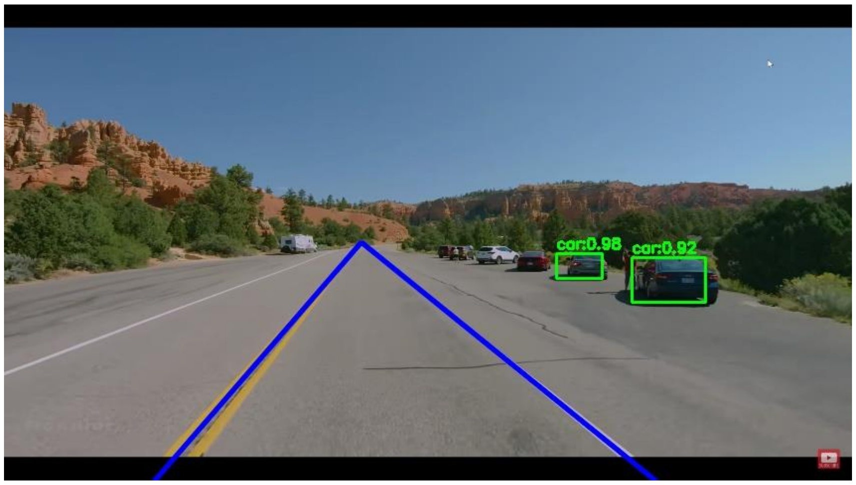


Figure 18 : Lane detection & CUDNN Yolov4-tiny Output

Using this program further steering and vehicle warnings were developed. Note that the data set selected didn't have cars coming right in front of the car. So to demonstrate that the warnings are working, once a car/truck comes within some range, "vehicle Nearby" warning pops up on the left top of the window. This feature was developed by adding threshold to the width of the object detected as the width will be directly proportional to the distance of the object.
The Turn Value is proportional to the steering angle, negative values signify left direction and positive value signify right direction, which can be seen on the right top of the window. This feature was developed by averaging the left and right lines and turn value is basically tilt of this new line. With this the optimal objective was successfully achieved.

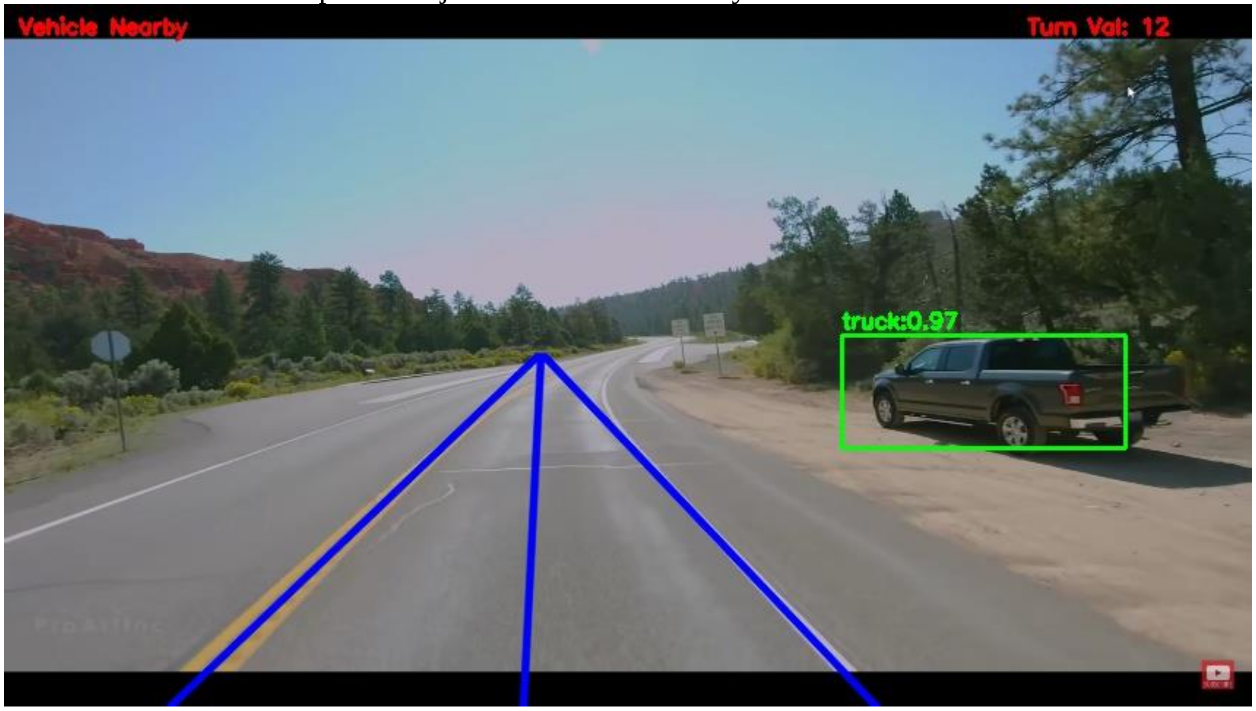


Figure 19 : Final Output

Conclusion

This project successfully developed an Advanced Driver Assistance System (ADAS) by progressing through three stages: minimum, target, and optimal. Beginning with basic lane detection using Canny edge detection and the Probabilistic Hough Transform, the system achieved reliable real-time processing, establishing a solid foundation.
Through this project, an exhaustive analysis of different algorithms provided by OpenCV and their comparison which OpenCV's CUDA extension has been provided. The developed lane detection application is highly reliable and can be developed further for higher performance. Overall, the study offered insightful information on real-time processing and system integration while showcasing how embedded computer vision may improve vehicle safety and further autonomous driving.

References

1. **https://github.com/siewertsmooc/ECV-ECEE-5763/tree/main/computer_vision_cv4_tested**
2. https://docs.opencv.org/4.5.4/index.html
3. https://o365coloradoedu-my.sharepoint.com/personal/siewerts_colorado_edu/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fsiewerts%5Fcolorado%5Fedu%2FDocuments%2FESEE%2DWeb%2DResources%2FECV%2D5763%2DCU%2Fcode
4. https://github.com/siewertsmooc/ECV-ECEE-5763/tree/main/simple-capture
5. https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html
6. https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html
7. https://learnopencv.com/object-detection-using-yolov5-and-opencv-dnn-in-c-and-python/
8. https://gist.github.com/YashasSamaga/e2b19a6807a13046e399f4bc3cca3a49
9. https://opencv.org/platforms/cuda/
10. https://www.youtube.com/watch?v=ZOZOqbK86t0
11. https://github.com/AlexeyAB/darknet
12. https://www.geeksforgeeks.org/opencv-real-time-road-lane-detection/

Appendices

In the outputs folder, all the three implementations:

1. Lane detection and Haar cascade: lane_haar.mp4
2. CUDA lane detection and CUDA Haar Cascade: cuda_lane_haar.mp4
3. Lane detection and CUDNN yolov4 tiny: cudnn_yolo_lane.mp4

In the Screenshots folder, the ffmpeg command screenshot, output of all the programs is present.
The input video has been uploaded to google drive.
Link: https://drive.google.com/file/d/1JLknG1IgN9M-hGOQ4gJ0QyYbNvsk9rDy/view?usp=sharing

The instructions for running the programs can be found in the readme.txt file in the codes folder