Introduction to Artificial Intelligence
CPSC 8100
Project Report: Phase 2

Vehicle detection and Pose Estimation for Autonomous driving

Bc. Libor Novák

Team Members:
Kushal Hebbar
Reetayan Das

Introduction

The scenario of driverless vehicles brought popularity in the field of public sector as an on-demand service. People's expectation increases regularly, and the autonomous driving technique is found to be very reliable. For an example, an autonomous vehicle has no dependency on human for its service, where in case drivers need to get rest after a certain time. The main objective of the autonomous driving needs the object detection technique. When we drive, we constantly pay attention to our environment by avoiding obstacles for our safety. Like introducing sensors, the most important criteria are to detect obstacles. To a seasoned observer, their readings are more specific and profound whereas for autonomous driving they are the representation of their data gets from their navigation and obstacle detector.

Extensive analysis of the architecture

The object detection method introduced in this paper is bound to locate an object within a box. This method can identify many classes of objects at a time. The solution of this system lies in Deep-Learning methods. In this model, a dense box is used as the primary architecture and applied Deep neural networks like MSCNN (Multiscale convolutional neural network) to specify the object proposal subnetwork and an object detection sub network. The unified network can be trained all together. The regions proposal methods are actually very useful to generate candidate windows. One can detect here the discriminative one from the samples and confirms which needs to confirmation or which can be ignored. It also introduces the region proposal method where the model has the capability to specify a domain which has interest points.

Region Proposal method

This method is necessary for image classification by confirming that the image window has highest priority. The algorithm runs the classifier thousand times per image and its main objective is to reduce the number of the evaluated windows. The region based convolutional neural network (RCNN) is used to combine with SPP Net as its training is expensive for both space and time. RCNN has an image from which it finds out the Region of Interest (ROI). A wrapped image region is then created for each of the region of interest and then forwarded to the convolutional neural network.

Applied Method for specification: DenseBox

Its main objective is to detect cars from all possible viewpoints, like in every weather condition sunny, warm, light and dark. The detector here is efficient, fast and able to provide the 3d and 2d regions selected for classification. The DenseBox makes an estimation to find a probabilistic map of object placed center in the image. For an arbitrary image size, it's easy to use a convolutional neural network.

Bounding box detection

The artificial neural network is introduced here to detect bounding boxes of the car taken from the images which are captured by a monocular camera. A fully convolutional neural network is introduced here which takes the output from the DenseBox as an input. The bounding boxes are the sliding window determined from the probability which is confirmed by the response maps. The output of the response maps resides at the center of the window.

Input layer and model

3 channel RGB features are used as an input. The input for our model can directly be a combination of images and labels but what differentiates our model with other models is the direct streaming input into the network rather than just using defined, trained images and labels. The model identifies the object during stream and learns from them. The longer the model is run the better it gets by gaining knowledge. For the bounding box we use width w and height h in the program. The streaming in our case is actual live gameplay of Grand Theft Auto 5. The game uses Directx11 and runs on the windows platform and requires high GPU powers. Our model should be preferably run on the Nvidia Geforce Titan X as it has been tested on it.

This model is not particular to GTA 5 and can be overlaid on any other game which consists of a decent feature/object definition capability. The reason we decided to run our model on GTA 5 is because of the sandbox feel to it and hence making it very customizable. The game comes with a lot of MODs that allow you to simulate weather, traffic, saturation features. We use OpenCV to grab the screen and that is directly given as the input to the model.

Another scenario for our model is generating and collecting data. This method can be used if in case the data needs to be used in a system where the game cannot be installed. For this model, it is better to collect the data using OpenCV and preferably riding in the game on a scooter or a motorcycle because this gives a 180-degree view and hence the lanes and surroundings are clearly visible. The motorcycle might be too fast for the frames to get generated clearly and hence using a scooter is a much better option.

Since I trained and tested the model on my laptop it has just a CPU and hence set the resolution of the game to 800x600 in windowed mode but if the model is trained and tested on a GPU then the game can be played in its highest resolution. Note that this might generate a large training dataset file due to how rich the frames are. Also note the FPS of the game and decide on the resolution.

Modules of the project

- **Self-driving AI:**

The initial idea of the project was to build a self-driving AI in GTA 5 but as I started building the model and tried giving the keys as input to the game I realized that dirextx11 does not support pyautogui. I then found a solution to the pyautogui issue here:
(https://stackoverflow.com/questions/14489013/simulate-python-keypresses-for-controlling-a-game)

One of the improvements for the phase 3 will be the self-driving AI in the GTA San Andreas game which supports pyautogui and the model will then be run on CARLA by Intel which has clear definitions of road lanes. A comparison analysis will be performed between the AI in the game and CARLA.

- **Object detection:**

Object detection in images, videos and live video streams through webcams have been around for a long time now and there are multiple deep learning packages that are available. Our project involves object detection with live stream gameplay of GTA 5 using OpenCV and we use Tensorflow in order to perform the object detection:
(https://github.com/tensorflow/models/tree/master/research/object_detection)
Tensorflow provides a robust model for object detection which performs image classification when it is trained on image or video data.

- **Datasets used:**

Our model was initially tested on the *COCO dataset* and upon testing showed high accuracy. Objects such as signals, human beings, cars, trucks, motorcycles, boats, bridges and street lights were recognized with high accuracy. However, it had some inaccuracies where it recognized billboards as a TV and the main menu as a microwave, but it was the most accurate compared to all other datasets that we used.
We then tested our model on the *Pascal dataset* and the results were poor. It recognized most basic objects such as cars and people, but the dataset resulted in a N/A for all other objects.
The third dataset and this test is what we were most optimistic about which was the *Kitti dataset*.
This dataset is produced by Karlsruhe Institute of Technology. These are video sequences from the streets of Karlsruhe. This dataset is good for 3d images where the bounding box and its coordinate axis is converted to a world reference frame by rotation around the y-axis and translation. Kitti datasets are real world first person view footage of scenes and objects around a city. The dataset was able to identify objects such as cars as it had previous exposure to cars, but no other object was identified as I tested it for almost 45 minutes. The only possible conclusion that I came up with was the difference in pixels between the game and the actual footage and the inaccuracy because of the distortion in the game as there was no GPU.
The following link was used to generate the various datasets: https://github.com/fvisin/dataset_loaders

- **Collision detection:**

This is something I came up with because of an issue I have had while playing the game. What this module of our project does is it checks proximity to other vehicles and objects and sends out warning signals to the user when he is 70% away from the object or car and sends out crash critical signals when he is 30% away from the object or car. The proximity calculations within the game is normally done using pixels and that was my initial inference but once I implemented the module to deal with pixels, it did not work exactly as I had thought. After further research on YouTube I found that the game works with percentages. This module uses the score that is derived from the tensorflow model. If the score is higher than the threshold (defined by us) only then will it activate the collision detection. The collision is measured by using the bounding boxes that are drawn for object detection. The distance between the mid-points of the boundary boxes is what is used as the measure to either return the warning signal or the crash critical signal. The collision detection module can be incorporated along with the self-driving module of the project. This enhances the performance of the self-driving

AI as it can not only detect lanes but can also detect other objects in its way and hence reduces collisions and going off track.

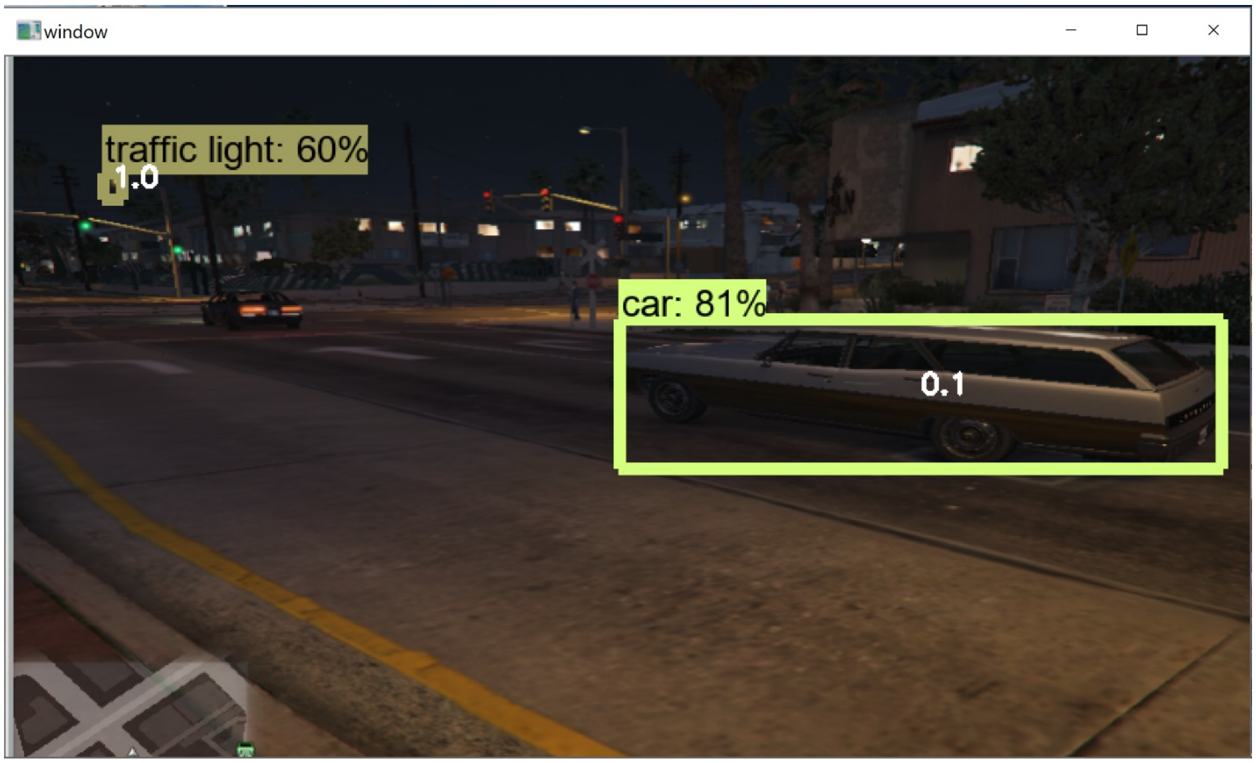
- **Finding the user transportation:**

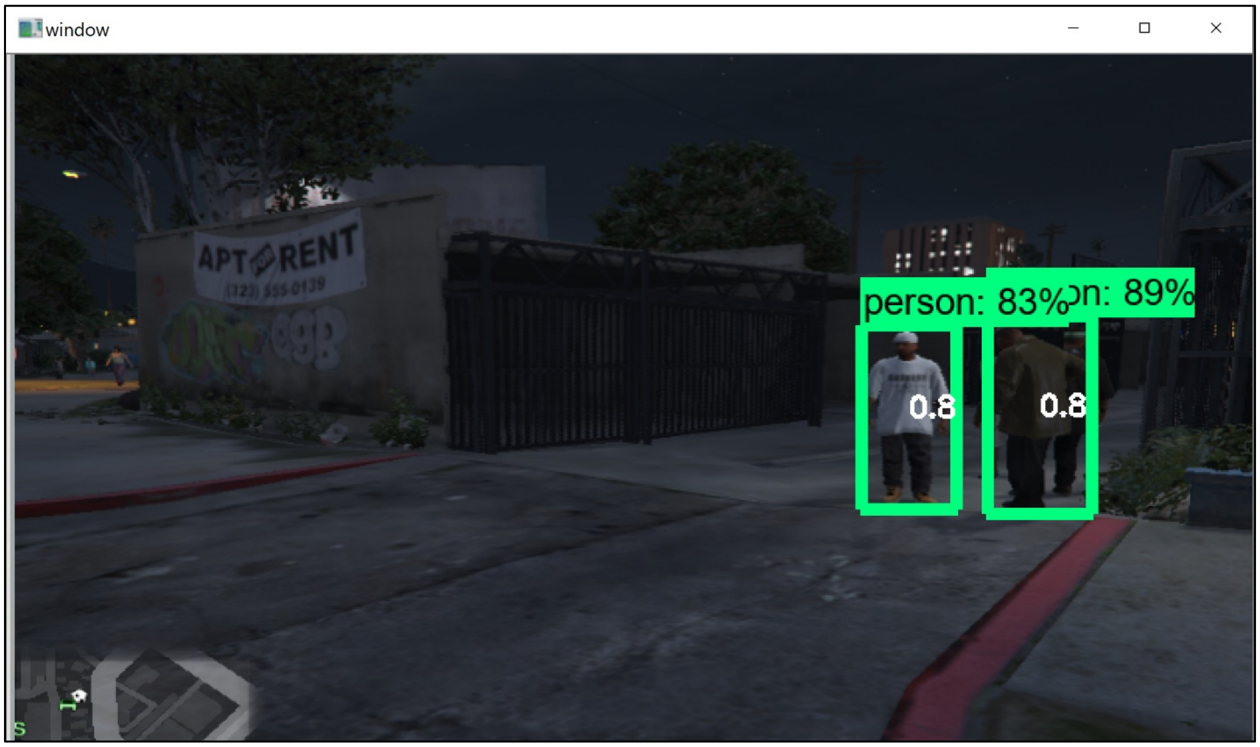
This module makes use of the object detection labels from the various datasets to detect if there is a transport in the user proximity and naturally walks towards the transport and steals it. The current module does not involve the self-driving AI and hence this module only steals the car but must be driven manually by the user and will be implemented along with the self-driving AI for the third phase of the project. The longer the game plays on its own using these modules the better it gets but ideally this would involve using a high-end GPU and a large memory or just an isolated system that runs the game on a loop non-stop.

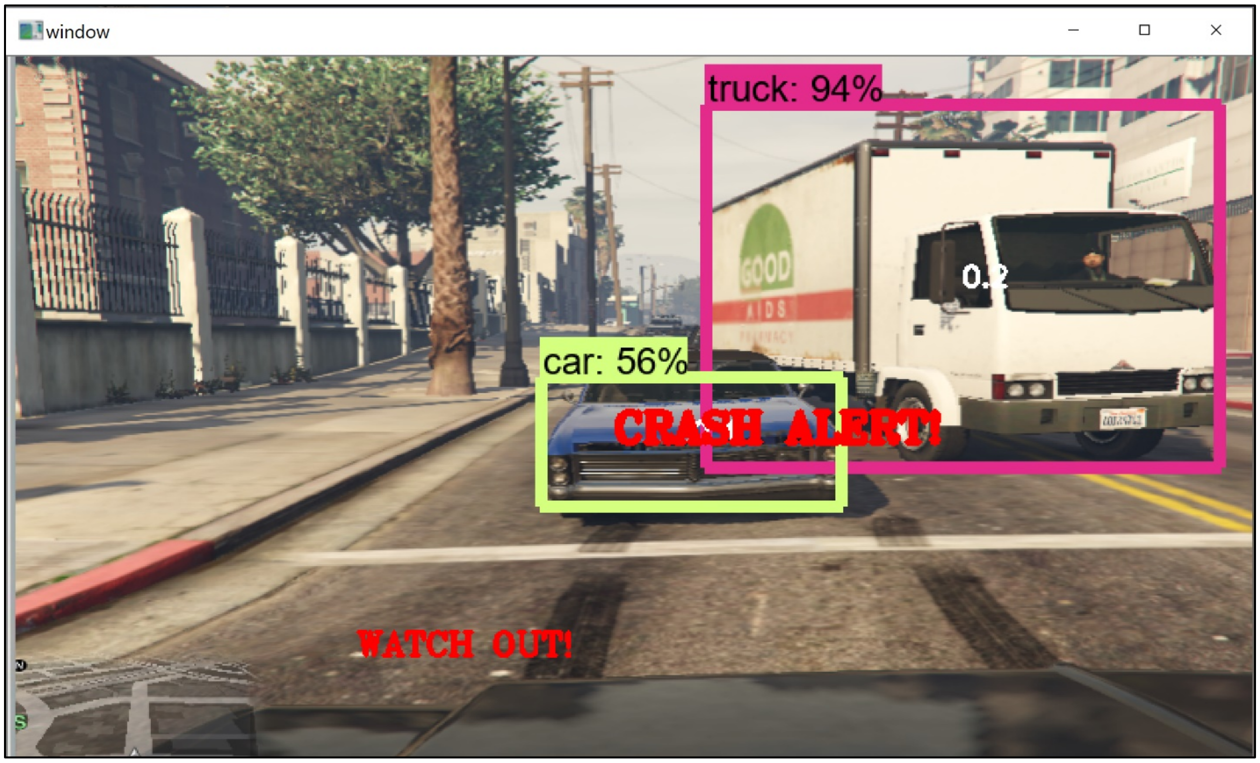
For the self-driving AI in the game, higher the number of frames (~600,000-700,000), higher the accuracy of the model and therefor higher the chances of the vehicle staying in between the lanes.

The self-driving AI in our project will also be tested on CARLA and the result will be in the form of a comparison between the two interfaces and which the module adapts better to. Currently our model uses pre-existing datasets and labels such as Kitti, COCO and Pascal.

Results




window
APT RENT
person: 83%
on: 89%
0.8
0.8


window
truck: 94%
car: 56%
0.2
CRASH ALERT!
WATCH OUT!

Please find the following link to the screen recording of the gameplay. There is a frame rate drop and lag because the game takes a lot of processing power and requires a GPU which is not present on my Laptop.
https://drive.google.com/open?id=1Ff-uIBz2JKYBejie_DrUj1dMUBVofQWE
(Link can only be accessed by using a g.clemson.edu email)

Future work

- Analyze the performance of our model on alternate games such as Watchdogs, CounterStrike 1.6 and GTA San Andreas
- Perform a comparative analysis between the performance of the model when it plays using data from GTA 5 and when it plays on CARLA
- Incorporate the self-driving AI which makes use of key presses and the object detection module implemented in this phase of the project
- Gender classification within the game (if the game differentiates between gender by default)
- Identify car make models and walk towards only certain car makes (preferred by the user)
- Incorporate Alexnet to perform the self-driving AI
- Make use of Tensorboard to graphical represent the model performance

Conclusion

Object detection being one of the major breakthrough using Neural Networks has grown and taken many forms. New methods are invented regularly and is ever growing. This project attempts to combine the power of object detection with other features within the game such as collision detection and self-driving modules. GTA 5 being a developer friendly game with its sandbox environment made it an obvious choice. Being able to compare and contrast the various datasets used resulted in COCO being the most efficient image dataset. Collision detection between vehicles in the game results in warning signals based on the distance between the bounded boxes. Future work for this project involves building a self-driving AI and also comparing the performance of the model on CARLA.

References

[1] Vehicle Detection and Pose Estimation for Autonomous Driving, Bc. Libor Novák, May 2017

[2] http://www.gamespp.com/directx/directInputKeyboardScanCodes.html

[3] Playing for Data: Ground Truth from Computer Games, Stephan R. Richter, Vibhav, Vineet, Stefan Roth, Vladlen Koltun, arXiv:1608.02192, Aug 2016 (https://www.youtube.com/watch?v=QkqNzrsaxYc)

[4] https://pythonprogramming.net/

[5] https://www.youtube.com/

[6] DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving, Chenyi Chen, Ari Seff, Alain Kornhauser, Jianxiong Xiao

[7] End to End Learning for Self-Driving Cars, Mariusz Bojarski, Davide Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, Karol Zieba, April 2016

[8] https://devblogs.nvidia.com/deep-learning-self-driving-cars/#disqus_thread

[9] Deep enforcement learning: an overview, Yuxi Li, arXiv:1701.07274v5, September 2017

[11] https://download.visinf.tu-darmstadt.de/data/from_games/

[12] https://github.com/fvisin/dataset_loaders

[13] https://stackoverflow.com/questions/14489013/simulate-python-keypresses-for-controlling-a-game

[14] https://github.com/tensorflow/models

[15] https://twitter.com/daniel_kukiela

[16] https://github.com/libornovax/master_thesis_code