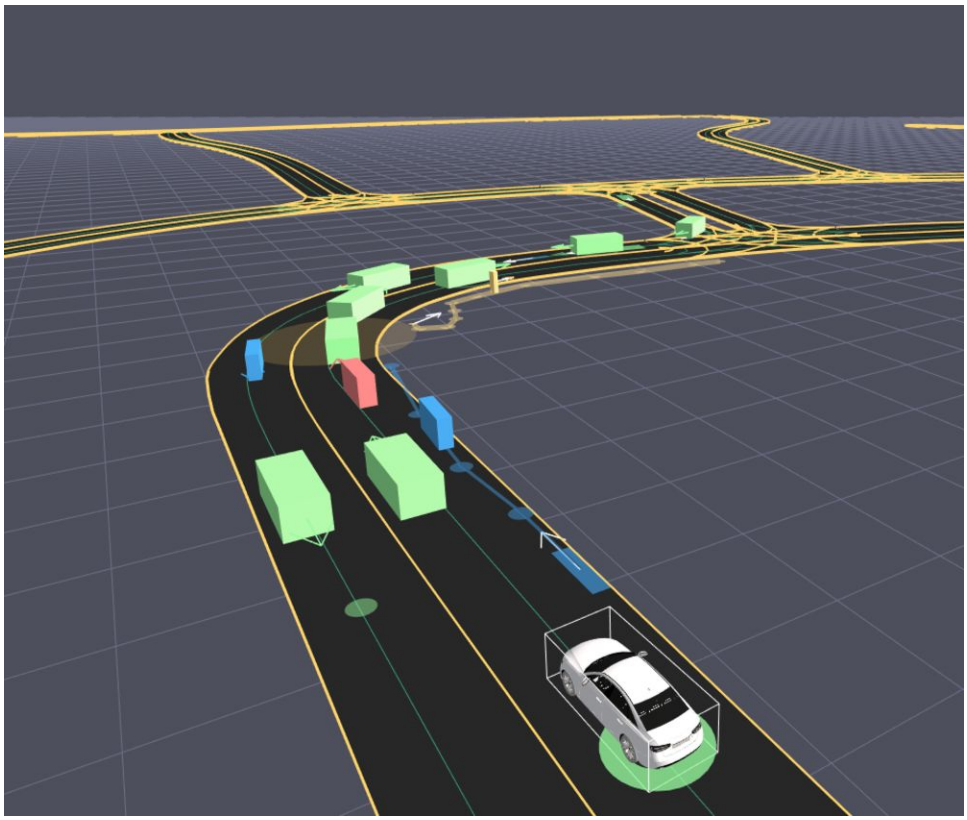
Self Driving Car In A Simulation.

Aug 17, 2021



Self-driving cars will continue to become increasingly popular and will form the future of personal transportation in everyday life. Companies like Tesla, Google and Waymo are already testing and developing this technology, making it an extremely new and unknown field of technology and research. This enticed me to approach this area of emerging technology as my project, but with a twist by using video games to act as a platform to test and develop parts of self-driving technology.

Overview

Using a video game (in this case Forza horizon 3) as a test environment. This is place where we can test and adapt the self-driving car model virtually, it also allows us to change specific parameters to evaluate how the self-driving car model adapts and behaves in different conditions, such as with a different car, different location (city, highway, offroad) and different weather conditions.

This research project will explore both the method and the real-world potential of a self-driving car model used in a video game simulation designed to also work in the real world. This project aims to assess the viability of a self-driving car model designed to function in the real world in a simulated environment. To assess this, a prerecorded video of a car being driven in a simulated environment(Video game) will be processed and put through several algorithms designed to extract useful and potentially relevant information that would assist a future full self driving car model to make accurate decisions based on these extractions.

Aim of this project

Is to accurately extract and identify features, from a prerecorded video in a simulated environment. These features should include active tracking of **lane markings, other vehicles, pedestrians, traffic lights and speed signs.**

System requirements

Operating system: latest version of linux ubuntu:20.04 LTS

CPU: Intel Core i5, preferred i7-7700

Ram: 8gb preferred 16gb

GPU: Preferred GTX 1060 (install GPU support tensorflow/gpu.com)

Programs and Packages installed

For this project we will be installing lots of programs many of switch we install onto our system through the command line (terminal) ensure you have python3.6 `sudo apt-get install python3.6`

- Opencv - Image processing and image manipulation library, opensource
    - Install by `pip install opencv-python`
- Tensorflow - Machine learning, building neural networks library developed by google
    - Install by `pip install tensorflow`
- Keras - Machine learning model saving and loading library
    - Install by `pip install keras`
- MatPlotLib - Math display library, to display graph data used to display machine learning training accuracy results
    - Install by `pip install matplotlib`
- Jupyter notebook - python run environment, makes it easy to debug and understand python code by displaying it in a notebook style
    - Install by `pip install jupyterlab`
    - `pip install notebook`
    - Run by `jupyter notebook`
- Numpy - adds support for large, multi-dimensional arrays and matrices within python for high performance mathematics
    - Install by `pip install numpy`
- Scikit learn- machine learning library for the Python It features various classification, regression and clustering algorithms
    - Install by `pip install -U scikit-learn`
- Pickle- used for saving and loading a trained machine learning model into tensorflow when used for testing
    - Install by `pip install pickle`
- imutils- a library for a series of convenience functions to make basic image processing functions easier
    - Install by `pip install imutils`

These are the biggest packages that are used in my project, to install all the packages needed to run lane detection and YOLO detection download and run the file found on my github: https://github.com/jimhoggey/SelfdrivingcarForza/blob/main/installall.sh

In terminal run `bash installall.sh` and type in your password to authenticate install.

Capabilities To Be Tested

1. **Read Lane Markings:** Using hog features, no sliding window searches for lines disrespective of other lane lines already found. (Basic line detection)

2. **Advanced Lane Detection and Prediction:** The use of a sliding window with multiple filters the car will draw both the left and right lane onto the video, use a birds eye view to identify the lanes and be more stable. Find the best search ratio

3. **Calculate and display steering wheel movements to turn left or right:** By detecting the lane marking we can calculate the position of the car in the lane, therefore work out what streaming inputs are needed to bring the car into the middle of the lane.

4. **Recognize vehicles on the road:** Use the YOLO pipeline to identify and categorise cars and assign a probability weight with them, of how accurate the model is. (99% high probability, <40% low probability)

5. **Recognize Pedestrians important to the car (people close to the road) (no action taken):** At this stage the car will recognize people but not yet make decisions or change its behavior. To make the car as safe as possible we need to know and categorize objects and people, that might affect both the safety of the car and the safety of the perdestains.

6. **Recognize traffic signs Traffic control (traffic lights):** (but not perform any action on this data yet) a similar algorithm to YOLO but used to detect and Recognize traffic signs that may be present in the view of the car. Stop sign, Speed limit sign, traffic lights
7. **Car to drive in between the lane markings:** This involves two steps at the end a feedback loop should be in action
    a. Get Forza Horizon working in a web browser or a virtual environment (win 10)
    b. Remap the controls of the xbox controller to keyboard buttons
    c. Integrate a screen recording function as an input for the lane code, and display the output live
    d. Integrate into the lane code to press left right key to turn the wheel

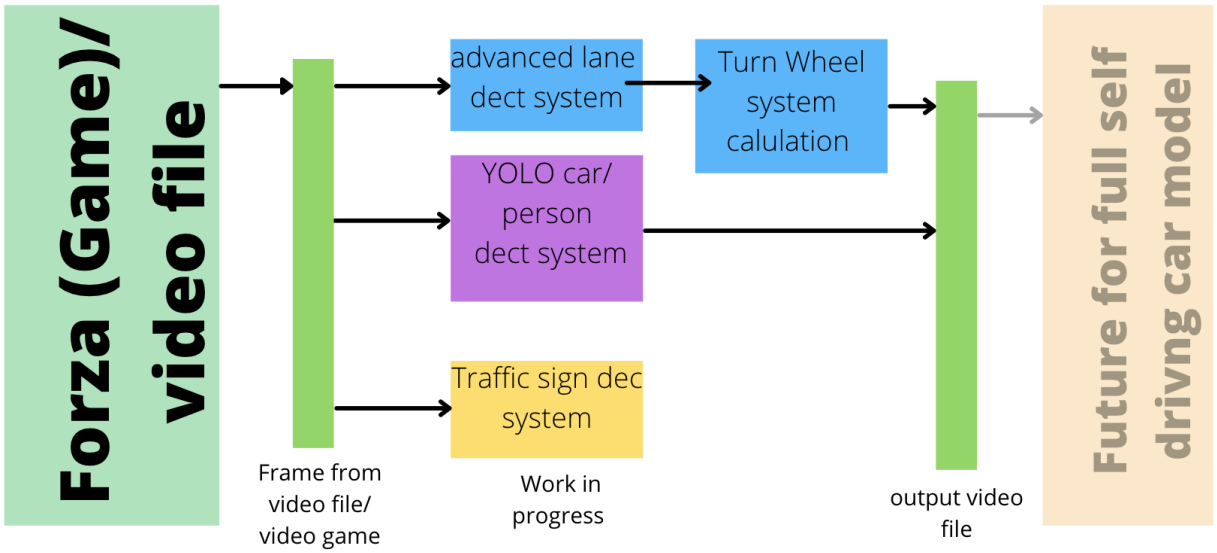
Overview of system

As detailed as possible without getting into code and machine learning. I will be using 3 different methods and machine learning approaches to extract each category of features.

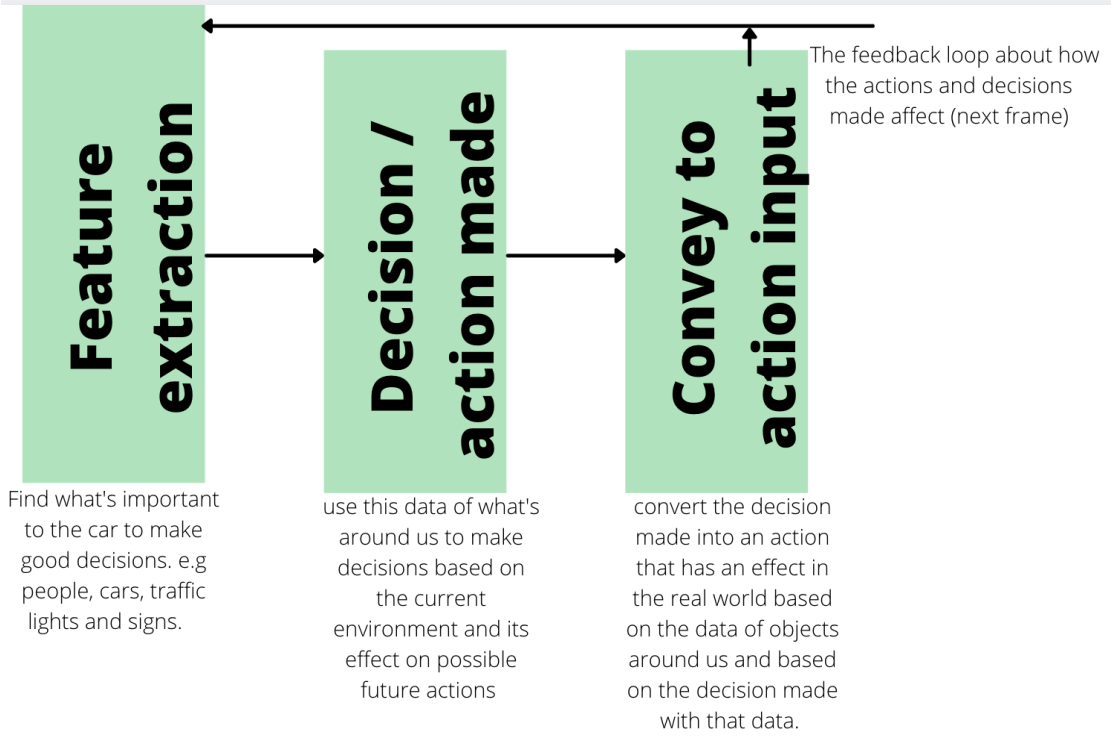
Extract Road Markings. **sliding window approach**	Extract other vehicles and people and traffic light. **YOLO**	Extract road signs. **HOG, SVM**
With the use of opencv2 and a range of image filters, such as a magnitude threshold filter, sobel x/y filter and a colour threshold. We can highlight the high contrast change in the lane line on the black road. A sliding window search function to narrow down the search window and searches where theres a high probability where the next line might be. To minimise confusion and false positive results.	we use YOLO (you only look once) with a huge training set that was pickled so its easy for us to use. YOLO uses a a convolutional neural network, that searches the frame for possible matches, applies bounding boxes and categorizes them. Anything lower then 50% will be disregarded.	We use hog features that creates a profile for how a road sign would look like and scans over the frame to find a similar match. This approach is overall a bit slower and less accurate. Hopefully to upgrade this method with a YOLO approach.

All these methods use opencv, imutils, matplotlib,numpy, and more.

A visual representation of how my model will extract features from the video file and display them.



Simplified the 3 sections of a self driving car, my project is focused on the first part (feature extraction of the environment.



Progress Report

I will meet with the teacher in the last 20min of our Thursday lesson to discuss progress and goals for the following week. I will also aim to keep a logbook of my daily progress to document errors and problems I run into in order to keep a record of what I have done to support me in case I have to backtrack what I have done.

Problem solving

As I progressed with this project I encountered and had to overcome many many errors and things blocking my path. In these situations I used problem solving skills to overcome these, such as asking the question on an online forum or trying a different approach.

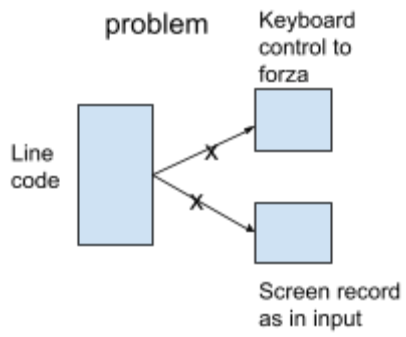
NONE ERROR

In the first generation of my line detection when I changed the original program to process a video instead of a signal frame I came across an error “TypeError: 'NoneType' object is not iterable Crash” that would process a few frames then crash. I worked out that this was some sort of python logic error. I tackled this error in the early stages of my project and at the time this was the only way to step forward so I posted a question on stack overflow. The answer came back and I worked out that I needed to test a variable to see if it's none before I could test something against this variable. View video: https://youtu.be/u32axlrp8uU stack overflow question asked: https://stackoverflow.com/questions/69087811/typeerror-nonetype-object-is-not-iterable-crash

Trying to get input to Forza Horizon

Nearing the end of my project I wanted to connect the line dect system to the forza horizon game, and use the video stream from the game as an input and therefore import a python package that could press the left and right key to control the car in forza, creating a feedback loop. Before getting into this, Identified two paths that needed to be completed in order for the system to interact with the game 1. Screen record forza as an input for the line detection pipeline and 2. keyboard input to control the game. I tried to see if it was even possible to control the game through the ‘keyboard’ a python package that can press keys on demand. I

loaded up forza though xbox's xcloud which is a game streaming platform but ran into problems because the xcloud version of forza did not support keyboard input as the windows 10 version of forza had, and I cant install forza natively because the computer was running on ubuntu. So first I tried to use an emulator of an xbox controller to trick xcloud into reading the inputs. No success. Next was a long shot I downloaded virtual box and installed a windows 10 iso image that I then ran and installed the windows 10 version of forza on it which did have keyboard support. As Forza was installing I wondered why there was no GPU option in windows. I realized that virtual box does not support gpu pass though to windows, which meant forza would be running poorly on the cpu, the cpu where the line detection system would also be running. Which meant the thing i was trying to prove wouldt work, because the line system needed lots of cpu power and forza being a graphically demanding game also needed as much power as possible. I didn't end up solving this problem but still looked into the bigger problem of how to set a screen recording as an input for my line detection system. Logbook video: https://youtu.be/gbRIlkofcc0 stack overflow question: https://stackoverflow.com/questions/69087811/typeerror-nonetype-object-is-not-iterable-crash



Risk Management

The potential risks of this project, are manageable. If I come across software or courses that require payment I have verbal confirmation that I would be given the correct support to gain access to this software or courses. If our school goes into lockdown, there are a couple of options such as open a remote desktop server on the computer at school and open a client on my personal computer and gain access to the computer this way to continue my project, though far from ideal as this will have performance and productivity impacts. At this point, I don't have reason to believe it would be hard to take the computer home and connect it to my workspace at home, this way I can continue to work on my project with no negative impacts.

I have divided the project into 3 main parts for the purpose of risk management, the first beginning where the project is getting started no main goals have been completed, the halfway

point where the project is well on its way with the halfway point in the goals reached, and finally the end where we have completed most of the goals.

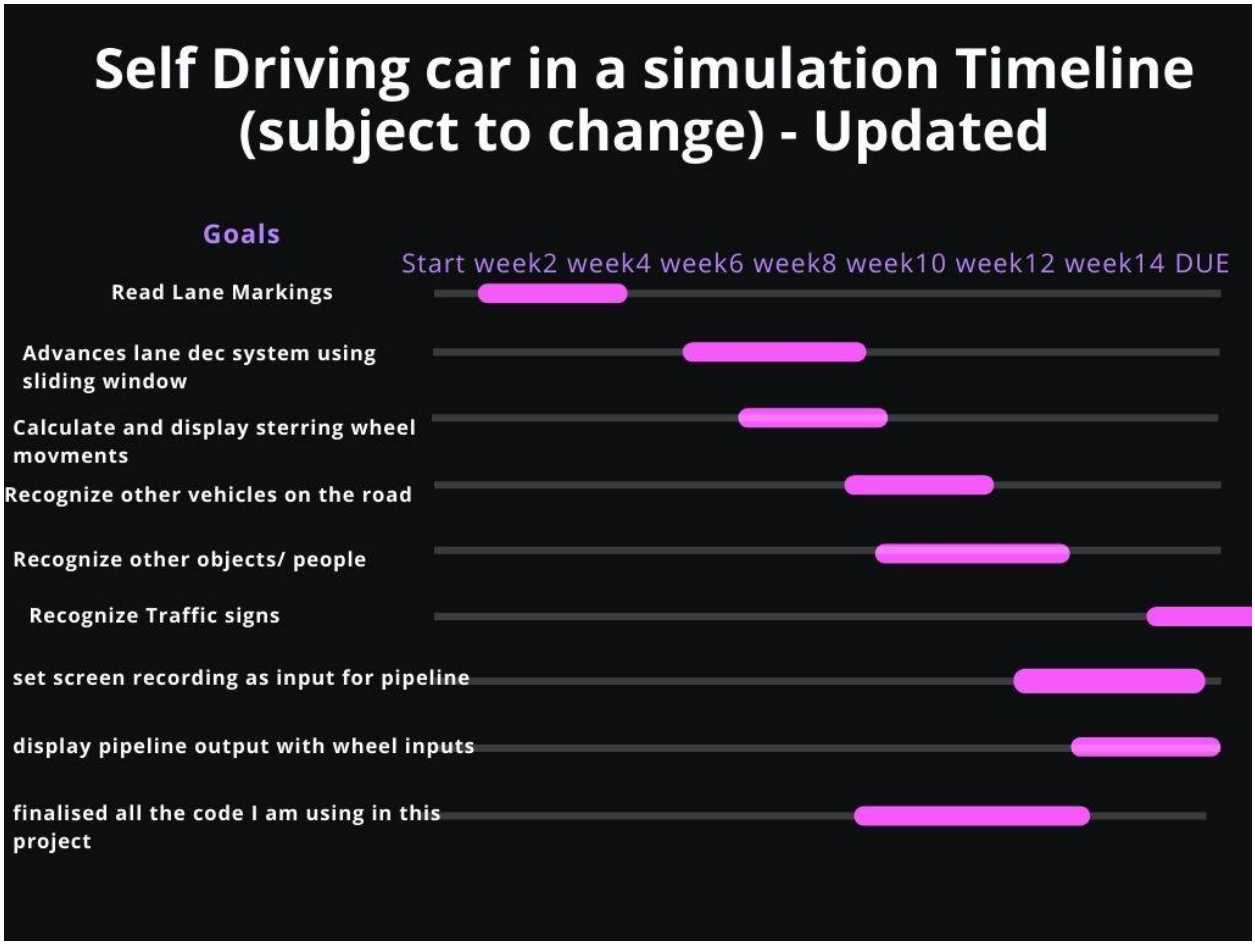
- **I come across content that requires a payment**
    - This can happen in any of the 3 parts of my project but mostly at the halfway point when my project is well on its way.
        - I have received verbal confirmation that I will be supported appropriately.

- **Unable to come to school for an extended period**
    - This can also happen in any point in my project but it will have a bigger effect on my project the closer to the halfway point or the end the project this risk happens. If this happened near the beginning I will have enough time to move my a new location to still have enough time to finish.
        - Open a remote desktop client and gain access to the computer though that
        - Take the computer home and connect it to my home set up with no negative impacts

- **Run out of time**
    - This is high risk near the end of the project as when my time runs out, there is little room for extension. To prevent this from happening I will keep a daily log of what I am doing, as well as track my goals and tick them off when I have completed them.
        - Complete project at the good level until next goal, keep detailed logs of what you have completed
        - Follow the timeline created to stay on track

Progress Timeline

My timeline and an approximation as how long it should take from starting to work on that goal to the end of when I have completed that goal.



How To get Started

Download the github repo: https://github.com/uppala75/CarND-Advanced-Lane-Lines

Download and run the install.sh file:
https://github.com/jimhoggey/SelfdrivingcarForza/blob/main/installall.sh

Open jupyter notebook and run the notebook in CarND or to transform a video file download advancedlinedectsys.py and edit the input and output video. Follow instructions
https://github.com/jimhoggey/SelfdrivingcarForza

Final Product

I will aim to produce two pieces as a result of completing this project.

- **The finished project will be the system successfully extracting important information from the prerecorded video clip.**
    - The system will recognize and detect other cars, people, traffic lights, and hold with confidence road lane marketings, and display wheel movements to keep the car in the middle of the lane when both lane lines are detected.

- **Produce a short clip demonstrating the extraction process**
    - A short 4min video that displays the active tracking of the lane lines, the detection of people and cars. Annotated with titles
    - A longer clip explaining more depth and understanding of the project and the systems used.

- **Produce easy installation of programs**
    - Make it easy for anyone to download the content and install all necessary packages, as well as run the code easily.

Video documentation:

Long video explaining in depth: https://youtu.be/KOZXrtPuaR8

Shorter video project video goal and output: https://youtu.be/Jl_7BrnvueU

None error video: https://youtu.be/u32axlrp8uU

Connecting to forza problem logbook: https://youtu.be/gbRIlkofcc0

Github project open source:
https://github.com/jimhoggey/SelfdrivingcarForza/blob/main/README.md

Biography

https://www.udemy.com/course/autonomous-cars-deep-learning-and-computer-vision-in-python/ (udemy course) for **first gen line detection system** and to learn about the basics (very good to get started)

https://github.com/uppala75/CarND-Advanced-Lane-Lines (**Advanced Lane Line System**) Really good write up, clear and works effectively

https://github.com/Sigil-Wen/YOLO (**YOLO**) simple, easy to use just have to download the weights, for car and people detection

Can not list all websites and forums accessed but the ones that directly helped me solve an error or the ones I used are.

https://github.com/tensorflow/tensorflow/issues/18538#issuecomment-403211069 (tensorflow install gpu error)

https://stackoverflow.com/questions/30861493/how-to-fix-python-valueerrorbad-marshal-data ValueError:bad marshal data error

https://www.codegrepper.com/code-examples/python/python+capture+desktop+as+video+source python video creation

https://stackoverflow.com/questions/59474533/modulenotfounderror-no-module-named-numpy-testing-nosetester numpy error not found

https://python-mss.readthedocs.io/examples.html#opencv-numpy

Thanks, Fynn

Glossary

- **FPS** - Frames Per Second (how fast and smooth a video is, is determined by the FPS)
- **Sobel** - Sobel filter, is used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasising edges.
- **Convolution** - The term convolution refers to both the result function and to the process of computing it.
- **Neural Network** - A computer system modelled after the human brain with the use of neurons and weights.
- **Hog Features** - HOG, or Histogram of Oriented Gradients, is a feature descriptor that is often used to extract features from image data, it creates a template of what an image needs to look like, for our classifier.
- **SVM -** Support-Vector Machines are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.