Introduction Slides

- Good afternoon everyone, I'm Chandan B S D. I'll be presenting on my internship project titled "Tracking and Counting Vehicles using Machine learning"
- My internal guide throughout the internship work is Nagamani Ma'am.

AGENDA

- The presentation is organized in the following manner.
- We will begin with the details of the company and finally conclude with the results of the project

About the Company

- As part of my internship I worked for a period of 1 months from April to May at Chira Information Technology Services.

- Chira IT services is a software service company concentrated on cloud computing, embedded systems and Application software.

- They frequently undertake corporate training and workshops for students in emerging technologies.

About Supervisor

- My external guide was Mr. Madhusudhana A S

- Sir has 7 years of experience in embedded systems, IOT and software development

- He is also the main team lead at Chira IT services.

Objectives

- The main objective of the internship project has been to leverage computer vision to achieve 3 major tasks:
    - First is to implement a system to detect cars on the highway

    - Second is To Track the vehicle throughout its journey within the frame of the camera.

    - Finally to count the total number of vehicles once it crosses an arbitrary line on the line

Details of Internship Project

- The project tackles three machine learning problems:
  - First is in video preprocessing that is conversion of a raw footage into a form ready for feature extraction
  - Second is object detection. Here it is the detection of cars on the highway
  - Third is Object Tracking. Here it is following the car throughout its journey on the road

- The input is an raw video feed from a single lane or dual lane highway

- The output is the processed video containing visualizations showing the detected objects and associated information.

Tasks Performed

- Since it was a guidened project work I was able to work on all modules of the system.
- The idea was to use the process of elimination. A common technique used in machine learning where we start with large volume of data and continuously remove noise to finally obtain usable information
- Finally I made use of ready-to-use functionality provided by the open source computer vision library to implement the project.

Tools / Technologies Used

- The internship required the use of python to build the live project.

- Along with python I made use of there major libraries - openCV, numpy and pandas
  - Opencv deals with computer vision operations
  - Whereas pandas and numpy was used to manipulate numerical data

- The development environment I used was Visual Studio Code for its simplicity and low system requirements

Architecture Diagram

The following image depicts the architecture block diagram of the proposed system.

- **Preprocessing Module:** The system begins with the preprocessing module. Here we accept the highway video and apply filters to transform the video into a format suitable for further processing

- **Car Detection Module:** This is an object detection module which gets the filtered video from the preprocessing module. This module detects the cars on the road and draws contours around it and calculates the central point.

- **Car Tracking Module:** The car tracking module tracks the car by assigning a unique id to each car and constantly updating the coordinates of the object for each frame of video.

- **Car counting Module:** Finally when the car enters ROI or the region of interest we begin counting the cars and using 2 baselines determine the direction of travel.

- **Output Video Processor:** This uses the input video feed to generate an output feed containing information regarding the detected cars and its total count.

Implementation Details: Preprocessing Module

- Now that we have a blueprint showing how the system is supposed to work let's see how it actually works.

- The implementation begins with the preprocessing module. The top left corner of the screen depicts how we have read the input video feed.

    - We first set the upper limit values for frame_count, frames per second or fps, width and height of the video.
    - We do this because computer vision operations are not performed on the raw video but on each frame of the video. Therefore video is treated as a large collection of images. Thus the complexity greatly decreases.

- In the next step we initialize all major information we aim to extract such as frame number, the total number of cars that have moved up or down the highway among others.

- We also decide the parameters for how the output video feed is to be saved.

Implementation Details: Preprocessing Module Contd.

Finally the actually preprocessing begins:

- First step is converting the color image frame into grayscale image. This is a standard preprocessing operation that helps in improving detection accuracy.

    a. Once this is done we then use a foreground mask, this is nothing but the removal of background from the object in focus. OpenCV provides the method fgbg.apply() to perform this operation.

- In the second step we apply morphological transformation. :-
    a. Particularly we apply 3 types of transformations which are Closing, Opening and Dilation.

    b. To apply each of these operations we use a structural unit called kernel.

    c. Firstly we perform closing operation
        - So why do we need closing? - This is because when the background of the image is removed from foreground i.e, when the road is removed from the car there are times that parts of the cars are accidentally removed. Closing just filled these holes in our object image.

    d. Secondly we apply a filter known as Opening - this just removes the overall noise in the image frame.

    e. Finally we perform dilation. After applying fgmask, closing and opening - our object would have become distorted and small. Dilation just enlarges the image using binary bits.

- Finally we decide on the size of a typical car in the video feed. We do this by defining a maximum area and minimum area for the contours. This is an important step that is used to differentiate a car from a tree, bike etc.

- Finally we draw two baselines to count and find direction of car movement.

Implementation Details: Object Detection Module

- This stage of implementation is where we actually draw the boundary or contours that I mentioned earlier.
    - We use the area parameters set before to determine if the object is a car or not.
    - We need to do this because we are using object detection, many entities in an image can be an object - a tree is also an object, traffic light is also an object and car is also an object
    - For the object to be a car its area must be within the minimum and maximum area for a typical car.

- If the object satisfies this condition then we draw contours around it and compute the centroid of that object. We display both the contour and centroid in the output feed.

Implementation Details: Car Tracking Module Contd.

- Detection of the object is not sufficient to achieve our objective.
- Now that we have detected the object we need to track it as long as it exists in the video frame. To do this we assign each car a unique id that stays with it permanently.
- The main reason to assign such id’s to to prevent duplication of data and ensure accurate count of cars
- While the video progresses at each frame the object moves, its coordinates and centroids are updated but its unique id remains the same.
- So the output of this module is moving car mapped with a unique identifier.

Implementation Details: Car Counting Module

- In this module we identify all the cars that are currently visible on the screen.

- Using the centroid and the car id’s we determine if they are near, currently crossing or have already crossed our baseline.

- Now in the earlier slides I mentioned that we have created two baselines. These baseline are essentially points in space that trigger the car counting module when an object crosses it.

- So why do we need 2 baselines?

    - Since one of the system objectives is to determine which direction the traffic is flowing we will require two baselines to do this.

    - Depending on the sequence in which the cars crossed the line we can determine the direction in which the traffic flows.

Implementation Details: Output Video Processor

Finally all the information that we identified we will annotate it on the output feed.
These details are

- **Cars in Area** is The number of cars within the two baselines
- **Cars crossed up** is the number of cars that touched baseline 2 first and then baseline 2
- **Cars crossed down** is the number of cars that touched baseline 1 first and then baseline 2
- **Total cars detected** is simply the number of contoured objects having the area between minimum car area and maximum car area
- **Frame number** is the image number in the video
- **Time** variable tells the amount of time elapsed in the video.

Finally we display the output.

Results: Unidirectional Traffic

Here we can see the results of our system on unidirectional traffic footage.

The left image shows all the data points and the information extracted.

The video to the right shows the application of one of the filters (foreground marks). We can see that only the cars are highlighted while the rest of the details are treated as noise.

This is just one filter that I have show in the screenshot. We applied 3 more transformations on this to make the object and the final environment is much more smooth.

Results: Bi-Directional Traffic

This screenshot shows the system output for bidirectional traffic in two separate lanes. Here we an see the count of cars moving down lane 1 and moving up lane 2

Application

Let’s look at a few practical applications that can be used in our project.

- The system can be used to correlate traffic frequency and time of the data. This can help with road planning and maintenance activities

- Secondly, connecting the system to traffic lights in smart cities can help with automating the flow of traffic.

Reflection

- Firstly I understood the use of libraries such as openCV, numPy and Pandas. The extent to which they simplify complex computation tasks made it very easy to get started with solving real work computer vision problems.

- I gained a better understanding of the fundamentals of computer vision. The most interesting part was the number of parameters and operations that are done in the preprocessing stage itself. Even before the actual objectives are implemented.

- Finally got a better sense of why python is useful for machine learning. The lack of type safety may be detrimental to traditional software development but in machine learning tasks where data is transformed from one form to another at each stage.