A Major Project Report On

**DETECTION OF ACCIDENTS UNDER LOW LIGHT CCTV MONITORING CONDITIONS IN TUNNELS**

Submitted in partial fulfillment of the Requirements for the award of the degree

## BACHELOR OF TECHNOLOGY IN

**INFORMATION TECHNOLOGY**

Submitted By

## Karrolla Rohith Mithra (19H61A1229) Musku Raja Reddy (19H61A1238)

**Silvani Nitya Sree (19H16A1250)**

Under the guidance of

**Mr. G. Sekhar Reddy**

Assistant Professor



**Department of Information Technology**

# ANURAG GROUP OF INSTITUTIONS

## (An Autonomous Institution)

**(Affiliated to JNTU-HYD, Approved by AICTE and NBA Accredited) Venkatapur (V), Ghatkesar (M), Medchal district, Hyderabad, Telangana,500088**

# 2019-2023

**ANURAG GROUP OF INSTITUTIONS**

## (An Autonomous Institution)

**(Affiliated to JNTU-HYD, Approved by AICTE and NBA Accredited) Venkatapur (V), Ghatkesar (M), Medchal district, Hyderabad, Telangana,500088 Department of Information Technology**



# CERTIFICATE

This is to certify that the project report entitled “ **DETECTION OF ACCIDENTS UNDER LOW LIGHT CCTV MONITORING CONDITIONS IN TUNNELS”**is a Bonafide work done and submitted by **Karrolla Rohith Mithra (19H61A1229), Musku Raja Reddy (19H61A1238) and Silvani Nitya Sree (19H61A1250)** in partial fulfillment of the requirements for the award of the degree of B.Tech in **Information Technology from Anurag Group of Institutions (An Autonomous Institution),** Affiliated to Jawaharlal Nehru Technological University, Hyderabad during the academic year 2022-2023 and the Bonafide work has not been submitted elsewhere for the award of any other degree.

### Internal Guide HOD

**Mr. G. Sekhar Reddy Dr. K. S. Reddy**

**Assistant Professor**  **Professor**

### Department of IT Department of IT

**External Examiner**

# ACKNOWLEDGEMENT

It is our privilege and pleasure to express my profound sense of respect, gratitude and indebtedness to our guide **Mr. G. Sekhar Reddy**, **Assistant Professor,** Department of Information Technology, Anurag Group of Institutions, Ghatkesar, for her indefatigable inspiration, guidance, cogent, discussion, encouragement and valuable advice throughout the dissertation work.

We would like to express our sincere thanks to **Dr. K. Sudheer Reddy,** Dean, Academic and Planning, Professor and Head of the Department of Information Technology, Anurag Group of Institutions, Ghatkesar, whose motivation in the field of software development has made us to overcome all hardships during the course of study and successful completion of the project.

We extend our sincere thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, Venkatapur(V), Ghatkesar(M), Medchal Dist., for their encouragement and constant help.

We would like to thank all the faculty of the IT department, for their support and encouragement.

Finally, we would like to express our heartfelt thanks to our parents who were very supportive both financially and mentally and for their encouragement to achieve our set goals.

**Karrolla Rohith Mithra (19H61A1229) Musku Raja Reddy (19H61A1238)**

**Silvani Nitya Sree (19H61A1250)**

# DECLARATION

This is to Certify that the project work entitled **“DETECTION OF ACCIDENTS UNDER LOW LIGHT CCTV MONITORING CONDITIONS IN TUNNELS”** submitted to Anurag Group of Institutions in partial fulfillment of the requirement for the award of the Degree of Bachelor of Technology (B-Tech), is an original work carried out By **Karrolla Rohith Mithra (19H61A1229), Musku Raja Reddy (19H61A1238), Silvani Nitya Sree (19H61A1250)** under the guidance of **Mr. G. Sekhar Reddy,** Assistant Professor in the Department of Information Technology. This matter embodied in this project is a genuine work, done by the students and has not been submitted whether the university or to any other University/Institute for the fulfillment of the requirement of any course of study.

**Karrolla Rohith Mithra (19H61A1229)**

**Musku Raja Reddy (19H61A1238)**

**Silvani Nitya Sree (19H61A1250)**

## ABSTRACT

The project focuses on Object Detection and Tracking System (ODTS) in combination with a well-known deep learning network - Faster Regional Convolution Neural Network (Faster R-CNN), for Object Detection and Conventional Object Tracking algorithm will be introduced and applied for automatic detection and monitoring of unexpected events on CCTVs in tunnels, which are likely to be (1) Wrong-Way Driving (WWD), (2) Stop, (3) Person out of vehicle in tunnel (4) Fire.

ODTS accepts a video frame in time as an input to obtain Bounding Box (BBox) results by Object Detection and compares the BBoxes of the current and previous video frames to assign a unique ID number to each moving and detected object. This system makes it possible to track a moving object in time, which is not usual to be achieved in conventional object detection frameworks.

A deep learning model in ODTS was trained with a dataset of images in tunnels to achieve Average Precision (AP) values of 0.8479, 0.7161 and 0.9085 for target objects: Car, Person, and Fire, respectively. Then, based on trained deep learning model, the ODTS based Tunnel CCTV Accident Detection System was tested using four accident videos.

As a result, the system can detect all accidents within 10 seconds. The objective achieved is that the detection capacity of ODTS could be enhanced automatically without any changes in the program codes as the training dataset becomes rich.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PG.NO** |
| **1** | **INTRODUCTION**  1.1 AIM, PURPOSE AND PROBLEM STATEMENT  1.2 BACKGROUND OF PROJECT  1.3 SCOPE OF PROJECT | **02**  02  03  04 |
| **2** | **SYSTEM ANALYSIS**  2.1 INTRODUCTION  2.2 HARDWARE AND SOFTWARE REQUIREMENTS  2.3 SOFTWARE REQUIREMENTS SPECIFICATION | **05**  05  07  08 |
| **3** | **TECHNOLOGIES USED**  3.1 DEEP LEARNING (YOLOv3)  3.2 PYTHON | **12**  12  16 |
| **4** | **SYSTEM DESIGN**  4.1 SOFTWARE DESIGN  4.2 ARCHITECTURE  4.3 UML DIAGRAMS | **18**  18  19  20 |
| **5** | **METHODOLOGY**  5.1 MODULES DESCRIPTION  5.2 PROCESS/ALGORITHM | **26**  26  27 |
| **6** | **IMPLEMENTATION**  6.1 CODE  6.2 OUTPUT SCREEN-SHOTS | **34**  34  44 |
| **7** | **TESTING**  7.1 TESTING STRATEGY  7.2 TEST CASES | **53**  53  55 |
| **8** | **CONCLUSION & FUTURE SCOPE** | **57** |
| **9** | **REFERENCES** | **58** |

**INTRODUCTION**

**1.1 AIM, PURPOSE AND PROBLEM STATEMENT**

**AIM:**

Our objective is to enhance the detection capacity of ODTS automatically without any changes in the program codes as the training dataset becomes rich.

Object Detection and Tracking System (ODTS) in combination with a well-known deep learning network, Faster R-CNN, is introduced and applied for Object Detection and Conventional Object Tracking algorithm for automatic detection and monitoring of unexpected events.

**PURPOSE:**

Object tracking is another area in image processing to be achieved by unique identification and tracking the positions of identified objects over time. However, to track objects, it is necessary to define object class and position first in a firstly given static image by object detection. Therefore, it can be said that the results of object tracking should be deeply dependent on the performance of the object detection involved.

**PROBLEM STATEMENT**

Unexpected events on CCTVs in tunnels, are likely to be (1) Wrong-Way Driving (WWD), (2) Stop, (3) Person out of vehicle in tunnel and (4) Fire.

Generally Surveillance in Tunnel is done through the CCTV monitoring but CCTV monitoring might work in general highway but this method might not be useful in tunnels because:

In the tunnel video has low luminance, so the video was greatly influenced by the tail light of the driving vehicle or the warning light of the car in operation.

The tone of the tunnel video was a dark colour. In other words, it has a different colour compared to the road of the tunnel outside.

**1.2 BACKGROUND OF PROJECT (LITERATURE SURVEY)**

## 1) Bird’s eye view localization of surrounding vehicles : Longitudinal and lateral distance estimation with partial appearance

## AUTHORS: E. S. Lee, W. Choi, D. Kum

On-road vehicle detection is essential for perceiving driving settings, and localizing the detected vehicle helps drivers predict possible risks and avoid collisions. However, there are limited works on vehicle detection with partial appearance, and the method for partially visible vehicle localization has not been explored. In this paper, a novel framework for vehicle detection and localization with partial appearance is proposed using stereo vision and geometry. First, the original images from the stereo camera are processed to form a v-disparity map. After object detection using v-disparity, vehicle candidates are generated with prior knowledge of possible vehicle locations on the image. Deep learning-based verification completes vehicle detection. For each detected vehicle, partially visible vehicle tracking algorithm is newly introduced. To track partially visible vehicles, this algorithm detects the vehicle edge on the ground, defined as the grounded edge, and then selects a reference point for Kalman filter tracking. Finally, a rectangular box is drawn on the bird’s eye view to represent vehicle’s longitudinal and lateral location. For testing the localization performance, the datasets in a highway and an urban setting are used and provide less than 1.5 m longitudinal error and 0.4 m lateral error in standard deviation.

# 2) Robust vehicle detection by combining deep features with exemplar classification

**AUTHORS:**  **L. Cao, Q. Jiang, M. Cheng, C. Wang**

Very recently, vehicle detection in satellite images has become an emerging research topic with various applications ranging from military to commercial systems. However, it retains as an open problem, mainly due to the complex variations in imaging conditions, object intra-class changes, as well as due to its low-resolution. Coming with the rapid advances in deep learning for feature representation, in this paper we investigate the possibility to exploit deep neural features towards robust vehicle detection.

In addition, along with the rapid growth in the data volume, new classification methodology is also demanded to explicitly handle the intra-class variations.

In this paper, we propose a vehicle detection framework, which combines Deep Convolutional Neural Network (DNN) based feature learning with Exemplar-SVMs (E-SVMS) based, robust instance classifier to achieve robust vehicle detection in satellite images. In particular, we adopt DNN to learn discriminative image features, which has a high learning capacity. In our practice, the leverage of DNN has achieve significant performance boost by comparing to a serial of handcraft designed features. In addition, we adopt E-SVMs based robust classifier to further improve the classification robustness, which can be considered as an instance-specific metric learning scheme. By conducting extensive experiments with comparisons to a serial of state-of-the-art and alternative works, we further show that the combination of both schemes can benefit from each other to jointly improve the detection accuracy and effectiveness.

# 3) Detection and classification of vehicles for traffic video analytics

**AUTHORS** **: A. Arinaldi, J. A. Pradana, A. A. Gurusinga**

We present a traffic video analysis system based on computer vision techniques. The system is designed to automatically gather important statistics for policy makers and regulators in an automated fashion. These statistics include vehicle counting, vehicle type classification, estimation of vehicle speed from video and lane usage monitoring. The core of such system is the detection and classification of vehicles in traffic videos. We implement two models for this purpose, first is a MoG + SVM system and the second is based on Faster RCNN, a recently popular deep learning architecture for detection of objects in images. We show in our experiments that Faster RCNN outperforms MoG in detection of vehicles that are static, overlapping or in night time conditions. Faster RCNN also outperforms SVM for the task of classifying vehicle types based on appearances.

**1.3 SCOPE OF PROJECT**

Object Detection and Tracking System (ODTS) in combination with a well-known deep learning network, Faster Regional Convolution Neural Network (Faster R-CNN), for Object Detection and Conventional Object Tracking algorithm will be introduced and applied for automatic detection and monitoring of unexpected events on CCTVs in tunnels, which are likely to be (1) Wrong-Way Driving (WWD), (2) Stop, (3) Person out of vehicle in tunnel (4) Fire. ODTS accepts a video frame in time as an input to obtain Bounding Box (BBox) results by Object Detection and compares the BBoxes of the current and previous video frames to assign a unique ID number to each moving and detected object

This system adds CADA that discriminates every cycle based on dynamic information of the car objects. As a result of experimenting with the image containing each accident, it was possible to detect the accidents within 10 seconds. On the other hand, training of deep learning secured the object detection performance of a reliable Car object, and Person showed relatively low object detection performance.

**SYSTEM ANALYSIS**

**2.1 INTRODUCTION:**

Object detection technology has been successfully applied to find the size and position of target objects appearing on images or videos. Several applications have appeared mainly in self-driving of vehicles, CCTV monitoring and security system, cancer detection, etc. Object tracking is another area in image processing to be achieved by unique identification and tracking the positions of identified objects over time. However, to track objects, it is necessary to define object class and position first in a firstly given static image by object detection. Therefore, it can be said that the results of object tracking should be deeply dependent on the performance of the object detection involved. This object tracking technology has been successfully utilized for tracing of targeted pedestrian and the moving vehicle, accident monitoring in traffic camera, criminal and security monitoring in the certain local area of concern, etc. In the traffic control field, a case study on analysis and control of traffic conditions by automatic object detection has carried out in this paper. The summaries are given as follows. According to, an on-road vehicle detection system for the self-driving car was developed. This system detects vehicle object and classifies the type of vehicle by Convolutional Neural Network (CNN). The vehicle object tracking algorithm tracks the vehicle object by changing the tracking center point according to the position of the recognized vehicle object on the image. Then, the monitor shows a localized image like a bird’s viewpoint with the visualized vehicle objects, and the system calculates the distance between the driving car and the visualized vehicle objects. This process of the system enables to objectively view the position of the vehicle object so that it can help assistance of the self-driving system. As a result, it can localize the vehicle object in vertical 1.5m, horizontal 0.4m tolerance at the camera. In, another deep learning-based detection system in combination with CNN and Support Vector Machine (SVM) was developed to monitor moving vehicles on urban roads or highways by satellite. This system extracts the feature from the satellite image through CNN using the satellite image as an input value and performs the binary classification with SVM to detect the vehicle BBox. Besides, Arinaldi, Pradana, and Gurusinga developed a system to estimate the speed of the vehicle, classify vehicle type, and analyze traffic volume. This system utilizes BBox obtained by object detection based on videos or images. The algorithm applied to the system was compared with the Gaussian Mixture Model SVM and faster RCNN. Then it appears that faster R-CNN was able to detect the position and type of vehicle more accurately. In other words, it could be said that the deep learning-based object detection approach is superior to the algorithm based object detection system. As a conclusion, all of the development cases in this paper deal with object information, showing outstanding performance with deep learning.

**EXISTING SYSTEM:**

In the existing system detects vehicle object and classifies the type of vehicle by Convolutional Neural Network (CNN). The vehicle object tracking algorithm tracks the vehicle object by changing the tracking center point according to the position of the recognized vehicle object on the image. Then, the monitor shows a localized image like a bird’s viewpoint with the visualized vehicle objects, and the system calculates the distance between the driving car and the visualized vehicle objects.

**DISADVANTAGES OF EXISTING SYSTEM:**

* This system extracts the feature from the satellite image through CNN using the satellite image as an input value and performs the binary classification with SVM to detect the vehicle BBox.
* This system utilizes BBox obtained by object detection based on videos or images. The algorithm applied to the system was compared with the Gaussian Mixture Model.
* **Algorithm**: Support Vector Machine (SVM),Convolutional Neural Network(CNN),Gaussian Mixture Model

**PROPOSED SYSTEM:**

In the proposed system we attempt is made for generate an object detection & tracking system (ODTS) with yolo, that can obtain moving information of target objects with names by combining object tracking algorithm with the deep learning-based object detection process. It is assumed that ODTS has been trained enough to perform object detection properly on a given image frame. ODTS receives selected frames of video at specified time interval c and gains sets of coordinates, n BBoxes are detected. BBoxT of objects on the given image frame at the time T, from the trained object detection system. The corresponding type or class ClassT of each detected object BBoxT is simultaneously classified by the object detection module.

**ADVANTAGES OF PROPOSED SYSTEM:**

* A deep learning model of R-CNN was used for training with yolo object detection model.
* This object tracking module was composed by introducing an object tracking model called yolo.

**Algorithm**: Faster R-CNN (Regional Convolution Neural Network),YOLO Model

****

**2.2 HARDWARE AND SOFTWARE REQUIREMENTS**

**HARDWARE REQUIREMENTS**

* **System :** Intel i3, i5, i7 Processor.
* **Hard Disk :** 500 GB.
* **Floppy Drive :** 1.44 Mb.
* **Monitor** : 14’ Colour Monitor.
* **Mouse :** Optical Mouse.
* **Ram :** 4 Gb.

**SOFTWARE REQUIREMENTS**

* **Operating system :** Windows 7,8,10. (Preferred: Windows 10 64 Bit OS)
* **Front-End :** Python.
* **Coding Language :** Python 3.7.4.
* **Software Environment :** Python IDLE or Command Prompt.

**2.3 SOFTWARE REQUIREMENTS SPECIFICATION**

**SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

* **ECONOMICAL FEASIBILITY**
* **TECHNICAL FEASIBILITY**
* **SOCIAL FEASIBILITY**

**A) ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### B) TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**C) SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**INPUT AND OUTPUT DESIGN**

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

* 1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
  2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
  3. 3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

**INPUT STAGES:**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**INPUT TYPES:**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**OUTPUT TYPES:**

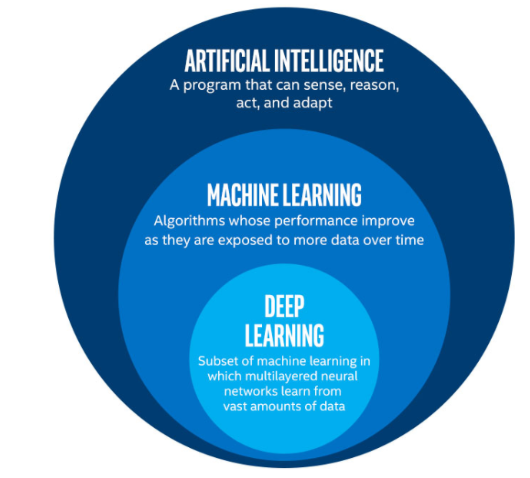
The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**TECHNOLOGIES USED**

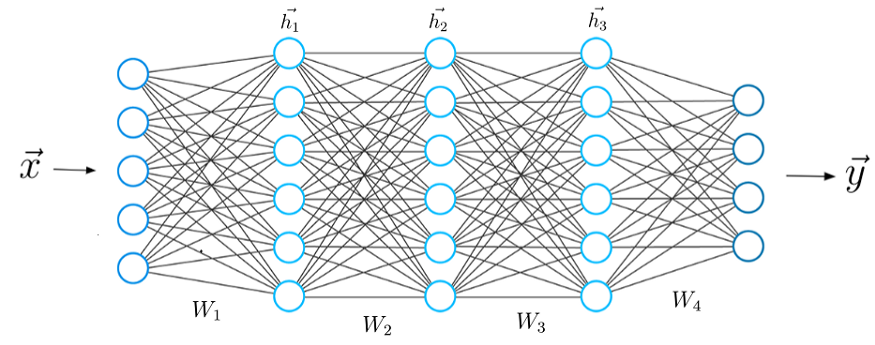
**3.1 Deep Learning**

Deep Learning is a subset of Machine Learning, which on the other hand is a subset of Artificial Intelligence. Artificial Intelligence is a general term that refers to techniques that enable computers to mimic human behavior. Machine Learning represents a set of algorithms trained on data that make all of this possible.

****

Deep Learning, on the other hand, is just a type of Machine Learning, inspired by the structure of a human brain. Deep learning algorithms attempt to draw similar conclusions as humans would by continually analyzing data with a given logical structure. To achieve this, deep learning uses a multi-layered structure of algorithms called neural networks.

The design of the neural network is based on the structure of the human brain. Just as we use our brains to identify patterns and classify different types of information, neural networks can be taught to perform the same tasks on data. The individual layers of neural networks can also be thought of as a sort of filter that works from gross to subtle, increasing the likelihood of detecting and outputting a correct result.



The human brain works similarly. Whenever we receive new information, the brain tries to compare it with known objects. The same concept is also used by deep neural networks.

**Neural networks enable us to perform many tasks, such as clustering, classification or regression.**With neural networks, we can group or sort unlabeled data according to similarities among the samples in this data. Or in the case of classification, we can train the network on a labeled dataset in order to classify the samples in this dataset into different categories.

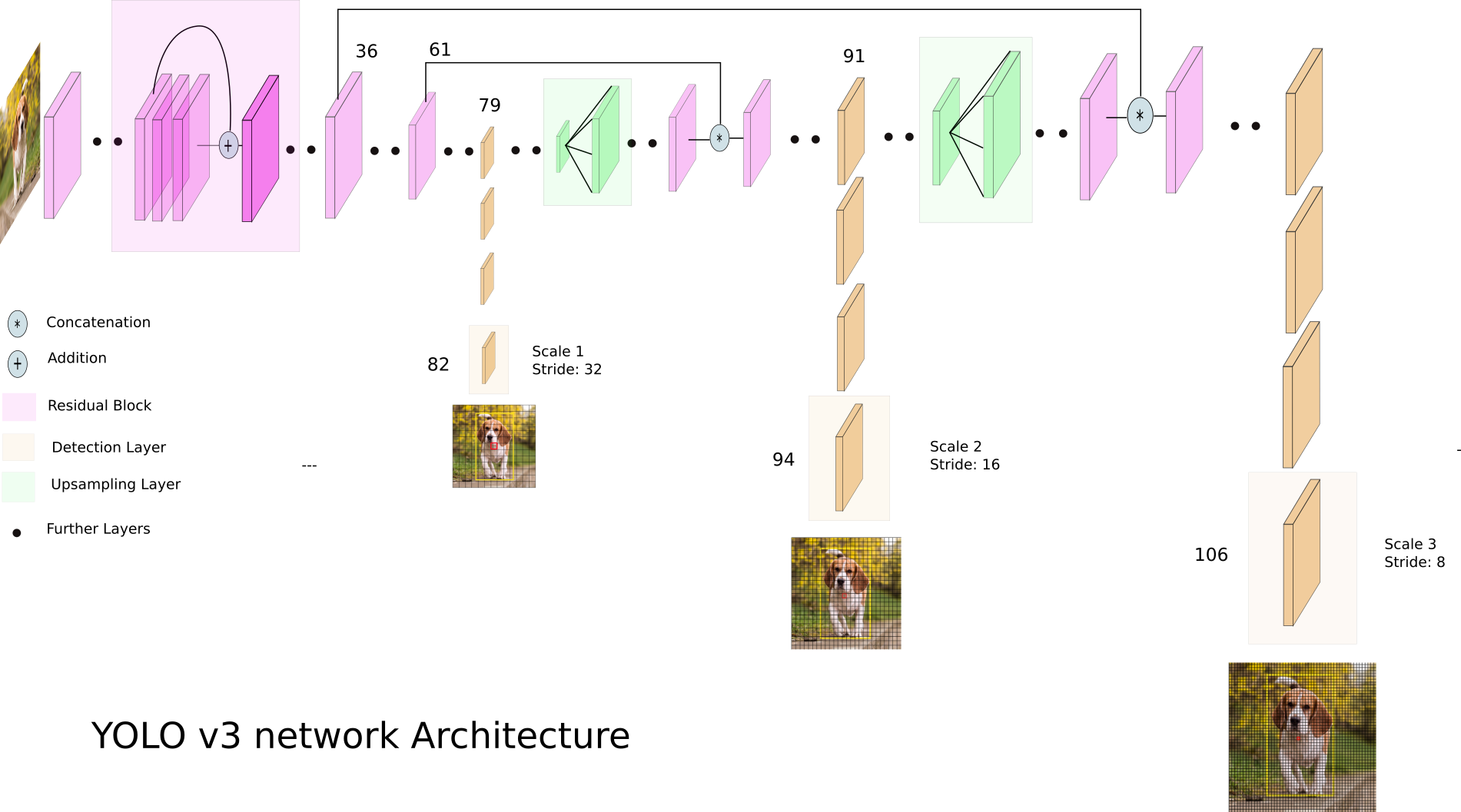
In general, neural networks can perform the same tasks as classical algorithms of machine learning. However, it is not the other way around. Artificial neural networks have unique capabilities that enable deep learning models to solve tasks that machine learning models can never solve.

**YOLOv3**

YOLO v2 used a custom deep architecture darknet-19, an originally 19-layer network supplemented with 11 more layers for object detection. With a 30-layer architecture, YOLO v2 often struggled with small object detections. This was attributed to loss of fine-grained features as the layers downsampled the input. To remedy this, YOLO v2 used an identity mapping, concatenating feature maps from from a previous layer to capture low level features.

However, YOLO v2’s architecture was still lacking some of the most important elements that are now staple in most of state-of-the art algorithms. No residual blocks, no skip connections and no upsampling. YOLO v3 incorporates all of these.

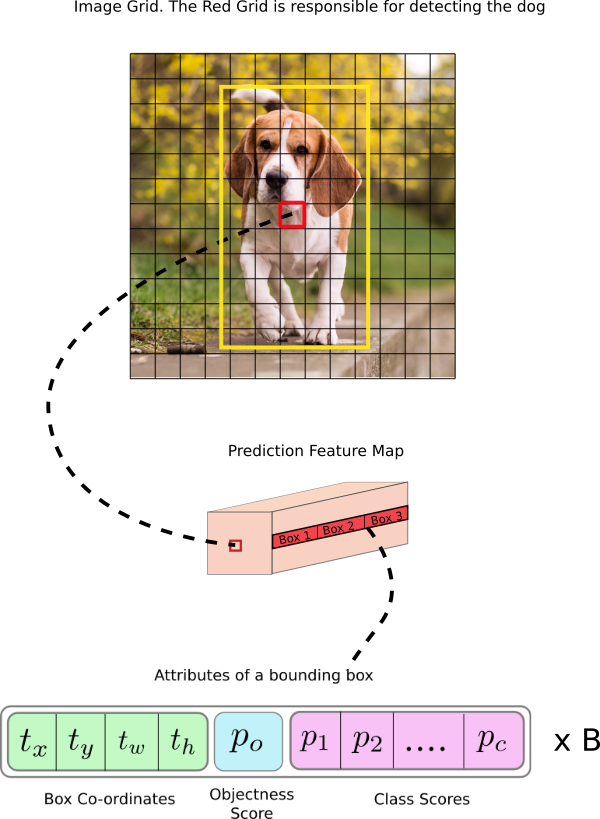
First, YOLO v3 uses a variant of Darknet, which originally has 53 layer network trained on Imagenet. For the task of detection, 53 more layers are stacked onto it, giving us a **106 layer fully convolutional underlying architecture for YOLO v3**. This is the reason behind the slowness of YOLO v3 compared to YOLO v2. Here is how the architecture of YOLO now looks like.

****

# Detection at three Scales

The newer architecture boasts of residual skip connections, and upsampling. **The most salient feature of v3 is that it makes detections at three different scales.**YOLO is a fully convolutional network and its eventual output is generated by applying a 1 x 1 kernel on a feature map. In YOLO v3, **the detection is done by applying 1 x 1 detection kernels on feature maps of three different sizes at three different places in the network.**

The shape of the detection kernel is **1 x 1 x (B x (5 + C) ).**Here B is the number of bounding boxes a cell on the feature map can predict, “5” is for the 4 bounding box attributes and one object confidence, and C is the number of classes. In YOLO v3 trained on COCO, B = 3 and C = 80, so the kernel size is 1 x 1 x 255. The feature map produced by this kernel has identical height and width of the previous feature map, and has detection attributes along the depth.



**3.2 Python**

**IMPLIMENTATION ON (PYTHON)**

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python.  This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

**Scripts are reusable**

Basically, a script is a text file containing the statements that comprise a Python program.  Once you have created the script, you can execute it over and over without having to retype it each time.

**Scripts are editable**

Perhaps, more importantly, you can make  different versions of the script by modifying the statements from one file to the next using a text editor.  Then you can execute each of the individual versions.  In this way, it is easy to create different programs with a minimum amount of typing.

**Python**

what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

Python concepts

If your not interested in the the hows and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

• Open source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-ish to interface with C++ (via SWIG)

• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features**

Python's features include −

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

**SYSTEM DESIGN**

**4.1 SOFTWARE DESIGN**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

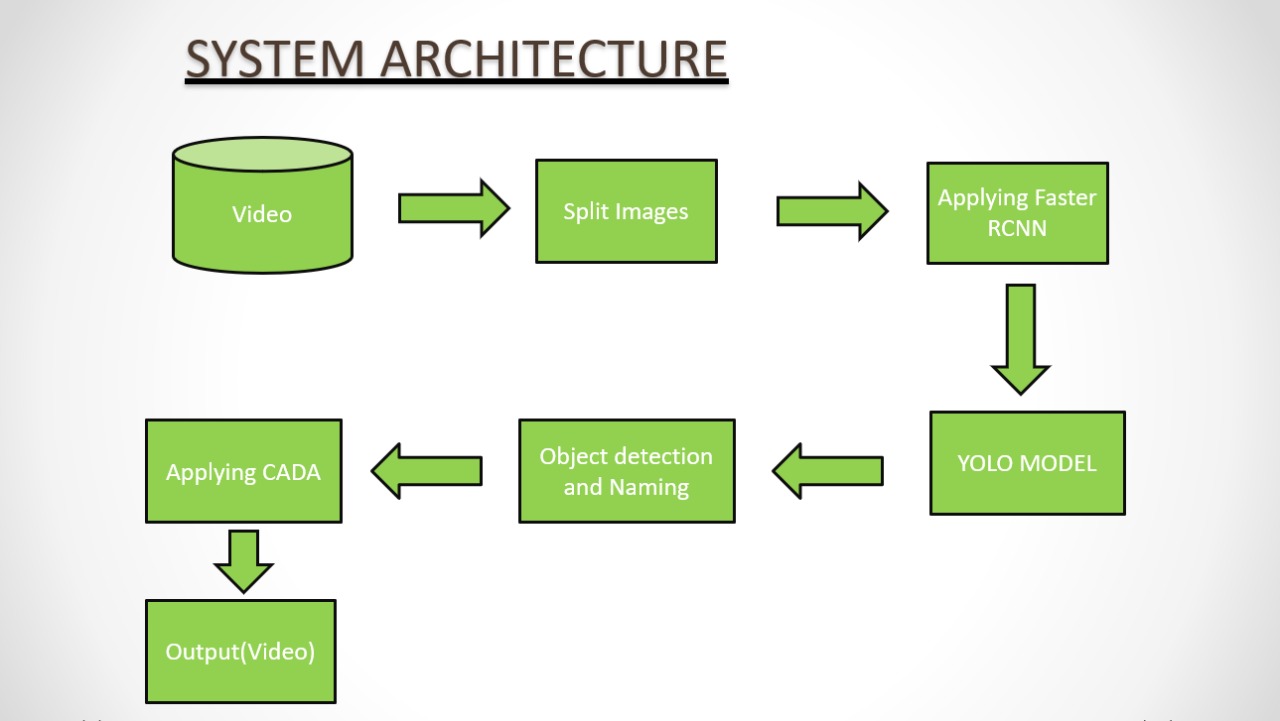
The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.

During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.

Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

**4.2 ARCHITECTURE**

****

**4.3 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

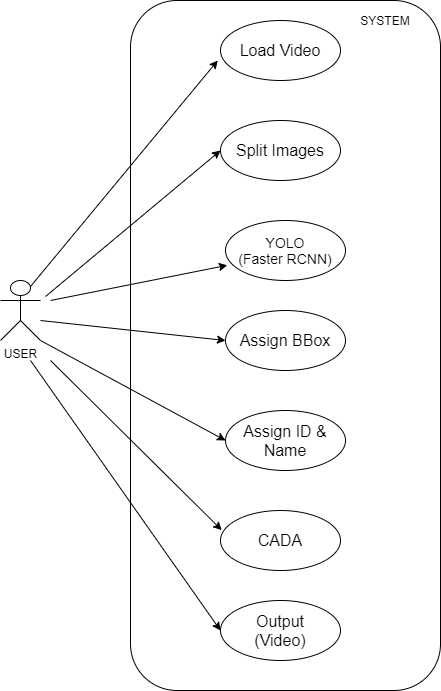
**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.

**Use Case Diagram**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

****

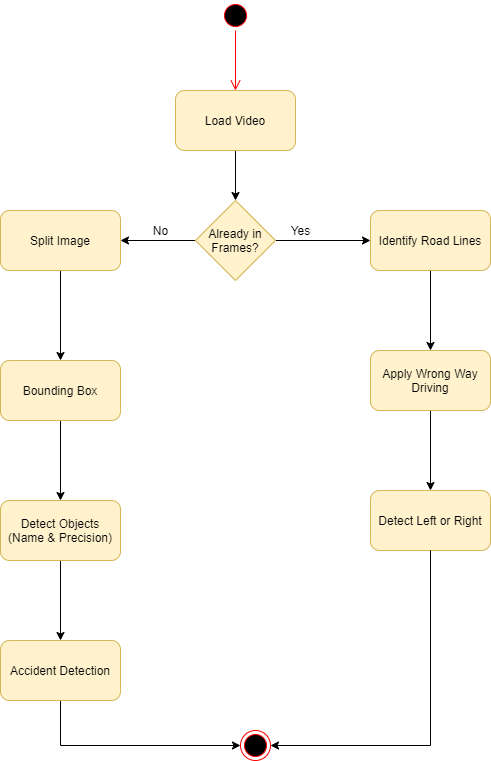
**Class diagram**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**State Chart Diagram**

Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered.

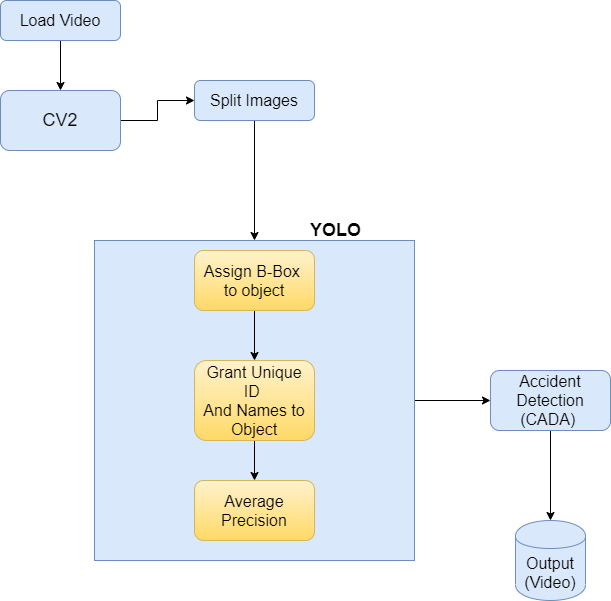
****

**Sequence diagram:**

A **sequence diagram** in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.



**FLOW DIAGRAM**

****

**METHODOLOGY**

**5.1 MODULES DESCRIPTION**

* **User**
* **Object Detection and Tracking**
* **RCNN**
* **Average Precision**

**MODULES DESCRIPTION:**

**User:**

User can load the cctv videos. To start the project user has to give –input (Video file path).The open cv class VideoCapture(0) means primary camera of the system, VideoCapture(1) means secondary camera of the system. VideoCapture(Videfile path) means with out camera we can load the pre recorded ideo file to the system. After that user has to load the yolo object detection system which is implemented on RCNN concepts. This yolo module is used for identify the objects from each frame and name that. It can be idenfied humans things fire etc…

**Object Detection and Tracking:**

Prior detection systems repurpose classifiers or localizers to perform detection. They apply the model to an image at multiple locations and scales. High scoring regions of the bounding box of the image are considered detections. We apply a Regional Convolution neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. Our model has several advantages over classifier-based systems. It looks at the whole image at test time so its predictions are informed by global context in the image.

**RCNN (Regional Convolution Neural Network):**

R-CNN models first select several proposed regions from an image (for example, anchor boxes are one type of selection method) and then label their categories and bounding boxes (e.g., offsets). Then, they use a CNN to perform forward computation to extract features from each proposed area. Afterwards, we use the features of each proposed region to predict their categories and bounding boxes.Then, based on the detected object information, a dependent object tracking module is initiated to assign the unique ID number to each of the detected objects, IDt and predict the next position of each of the objects, BBOX. The number of tracking BBox u is different from n. But If past tracked BBox is 0, the number of tracking BBox equals to the number of the detected objects.

**Average Precision:**

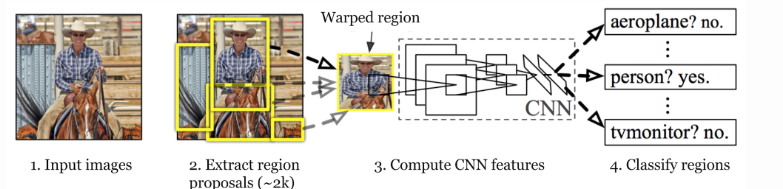
AP values for the target objects to be detected, in the training dataset, the number of Car objects is the largest object and very high AP value was obtained for the Car object in comparison with other classes. That is, the object detection performance of deep running of the Car in the video was expected to be highly reliable. On the

other hand, AP for Person object results in relatively low value because Person object exists long, tiny shape in small size. The AP of Fire object was high, but false detection for the object might be highly possible as the number of the objects available for training was very small, Nonetheless, training about deep learning, including No Fire objects, could reduce the false detection of Fire object. However, to detect the Fire in the tunnel control center, it was necessary to collect and involve more images of a Fire event in training.

**5.2 ALGORITHMS USED:**

**1. R-CNN | REGION BASED CNNS**

Since Convolution Neural Network (CNN) with a fully connected layer is not able to deal with the frequency of occurrence and multi objects. So, one way could be that we use a sliding window brute force search to select a region and apply the CNN model on that, but the problem of this approach is that the same object can be represented in an image with different sizes and different aspect ratio. While considering these factors we have a lot of region proposals and if we apply deep learning (CNN) on all those regions that would computationally very expensive.

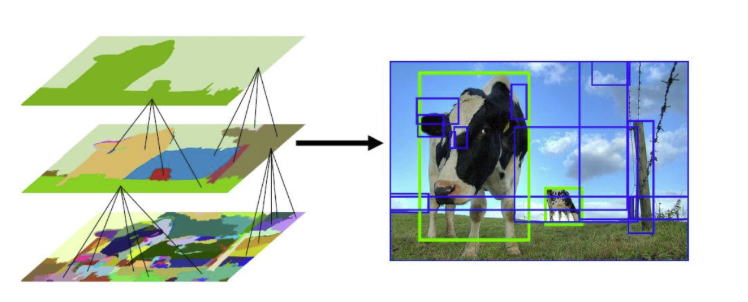
[](https://media.geeksforgeeks.org/wp-content/uploads/20200219161502/RCNN1.png)

**R-CNN architecture**

*Ross Girshick et al.in 2013 proposed an architecture called R-CNN (Region-based CNN) to deal with this challenge of object detection. This R-CNN architecture uses the selective search algorithm that generates approximately 2000 region proposals. These 2000 region proposals are then provided to CNN architecture that computes CNN features. These features are then passed in an SVM model to classify the object present in the region proposal. An extra step is to perform a bounding box regressor to localize the objects present in the image more precisely.*

**Region Proposals:**

Region proposals are simply the smaller regions of the image that possibly contains the objects we are searching for in the input image. To reduce the region proposals in the R-CNN uses a greedy algorithm called selective search.

[](https://media.geeksforgeeks.org/wp-content/uploads/20200219133430/seelctive-search.png)Region Proposals generation using selective search

**Selective Search:**  
Selective search is a greedy algorithm that combines smaller segmented regions to generate region proposal. This algorithm takes an image as input and output generate region proposals on it. This algorithm has the advantage over random proposal generation is that it limits the number of proposals to approximately *2000* and these region proposals have a high recall.

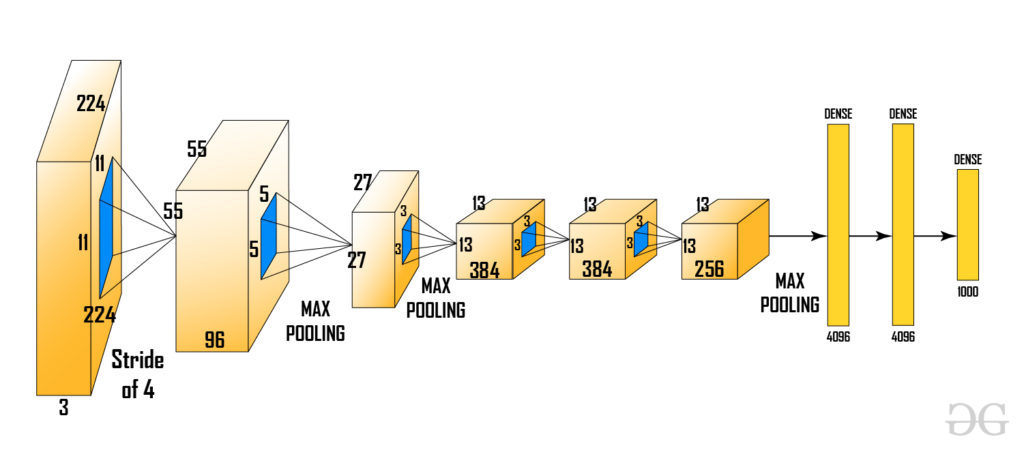
**Algorithm:**

1. Generate initial sub-segmentation of input image.
2. Combine similar bounding boxes into larger ones recursively.
3. Use these larger boxes to generate region proposals for object detection.

In Step 2 similarities are considered based on color similarity, texture similarity, region size, etc.

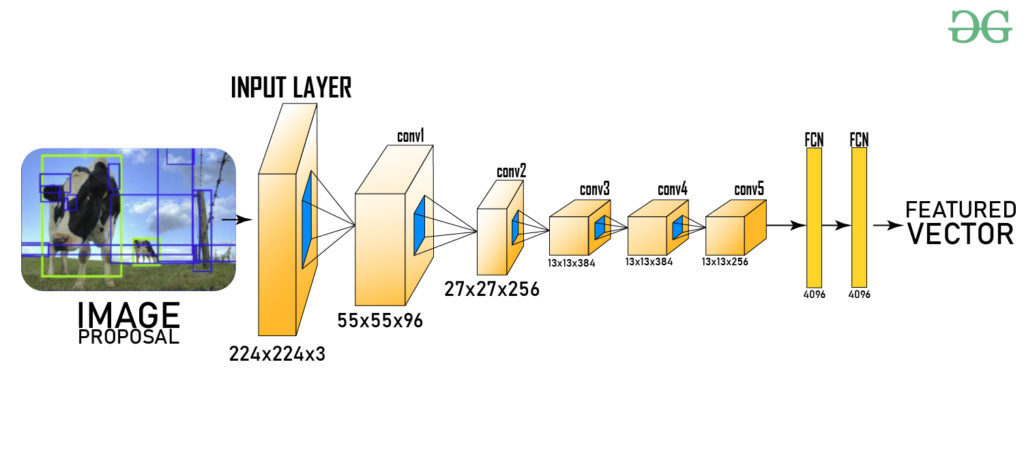
**2. CNN architecture of R-CNN:**

After that these regions are warped into the single square of regions of dimension as required by the CNN model. The CNN model that we used here is a pre-trained AlexNet model, which is the state of the art CNN model at that time for image classification.

[](https://media.geeksforgeeks.org/wp-content/uploads/20200217183955/new6.jpg)

Here the input of AlexNet is (227, 227, 3). So, if the region proposals are small and large then we need to resize that region proposal to given dimensions.

From the above architecture, we remove the last softmax layer to get (1, 4096) feature vector. We pass this feature vector into SVM and bounding box regressor.

[](https://media.geeksforgeeks.org/wp-content/uploads/20200219162206/feature1.jpg)

**3. SVM (Support Vector Machine):**

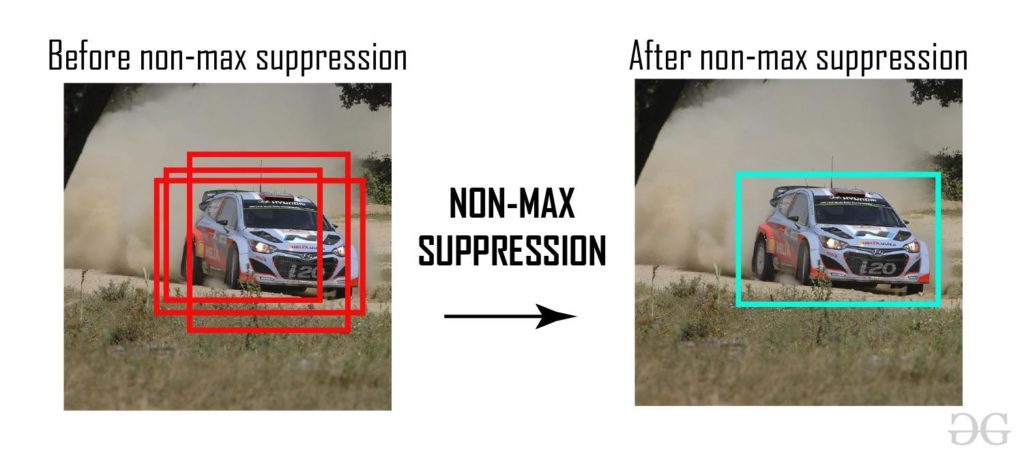
The feature vector generated by CNN is then consumed by the binary SVM which is trained on each class independently. This SVM model takes feature vector generated in previous CNN architecture and outputs a confidence score of the presence of an object in that region. However, there is an issue for training with SVM is that we required AlexNet feature vectors for training SVM class. So, we could not train AlexNet and SVM independently in paralleled manner. This challenge is resolved in future versions of R-CNN (Fast RCNN and Faster RCNN)

**Bounding Box Regressor:**  
In order to precisely locate the bounding box in the image., we used a scale-invariant linear regression model called bounding box regressor. For training this model we take as predicted and Ground truth pairs of four dimensions of localization. These dimensions are *(x, y, w, h)* where *x* and *y* are the pixel coordinates of center of bounding box respectively. w and h represents the width and height of bounding boxes. This method increases Mean Average precision (mAP) of the result by *3-4%*.

[[](https://media.geeksforgeeks.org/wp-content/uploads/20200219162610/2020-02-171.jpg)](https://media.geeksforgeeks.org/wp-content/uploads/20200219162610/2020-02-171.jpg)

**Output:**  
Now we have region proposals that are classified for every class label. In order to deal with the extra bounding box generated by the above model into the image, we use an algorithm called **Non-Maximum Suppression.**  
It works in 3 steps:

* Discard those objects where the confidence score is less than a certain threshold value( say 0.5).
* Select the region which has the highest probability among candidates regions for object as predicted region.
* In the final step we discard those regions which has IoU (intersection Over Union) with predicted region over 0.5.

[](https://media.geeksforgeeks.org/wp-content/uploads/20200219163223/nonmaxsuppression.jpg)  
After that we can obtain output by plotting these bounding boxes on input image and labeling objects that are present in bounding boxes.

**Results:**   
The R-CNN gives Mean Average Precision (mAPs) of 53.7% on VOC 2010 dataset. On 200-class ILSVRC 2013 object detection dataset it gives mAP of 31.4% which is large improvement from previous best 24.3%. However, this architecture is very slow to train and takes ~ 49 sec to generate test results on a single image of VOC 2007 dataset.

**Challenges of R-CNN:**

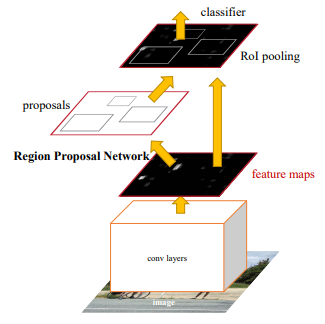
* Selective Search algorithm is very rigid and there is no learning happens in that. This sometimes leads to bad region proposals generation for object detection.
* Since there are approximately 2000 candidate proposals. It takes a lot of time to train the network. Also we need to train multiple steps separately (CNN architecture, SVM model, bounding box regressor). So, This makes it very slow to implement.
* R-CNN can not be used in real time because it takes approximately 50 sec to test an image with bounding box regressor.
* Since we need to save feature maps of all the region proposals. It also increases the amount of disk memory required during training.

## 3. Faster R-CNN:

It turns out that Fast R-CNN is still pretty slow, and that is mostly because the CNN is bottlenecked by the aforementioned region proposal algorithm, selective search. **Faster R-CNN**solves this by abandoning the traditional region proposal method, and relying on a fully deep learning approach. It consists of two modules: a CNN called **Region Proposal Network** (RPN), and the Fast R-CNN detector. The two modules are merged into a single network and trained end-to-end.

The authors of Faster R-CNN drew inspiration from the attention mechanism when they designed RPN to emphasize what is important in the input image. Creating region proposals is done by sliding a small network over the last shared convolution layer of the network. The small network requires a (n x n) window of the convolutional feature map as an input. Each sliding window is mapped to a lower-dimensional feature, so just like before, it is fed to two fully connected layers: a box-classification and box-regression layer.

It is important to mention that the bounding boxes are parametrized relative to hand-picked reference boxes called **anchors**. In other words, the RPN predicts the four correction coordinates to move and resize an anchor to the right position, instead of the coordinates on the image. Faster R-CNN is using 3 scales and 3 aspect ratios by default, resulting in 9 anchors at each sliding window.



Faster R-CNN framework

Faster R-CNN is considered state-of-the-art, and it is certainly one of the best options for object detection. However, it does not provide segmentation on the detected objects, i.e. it is not capable of locating the exact pixels of the object, rather just the bounding box around it. In many cases this is not needed, but when it is, Mask R-CNN should be the first one to come to mind.

**IMPLEMENTATION**

* 1. **CODE:**

**CCTV\_Video.py**

# USAGE

import numpy as np

import argparse

import imutils

import time

import cv2

import os

import subprocess

import os, shutil

folder = 'output'

for filename in os.listdir(folder):

file\_path = os.path.join(folder, filename)

try:

if os.path.isfile(file\_path) or os.path.islink(file\_path):

os.unlink(file\_path)

elif os.path.isdir(file\_path):

shutil.rmtree(file\_path)

except Exception as e:

print('Failed to delete %s. Reason: %s' % (file\_path, e))

# construct the argument parse and parse the arguments

ap = argparse.ArgumentParser()

ap.add\_argument("-i", "--input", required=True,

help="path to input video")

ap.add\_argument("-o", "--output", required=False,

help="path to output video")

ap.add\_argument("-y", "--yolo", required=True,

help="base path to YOLO directory")

ap.add\_argument("-c", "--confidence", type=float, default=0.5,

help="minimum probability to filter weak detections")

ap.add\_argument("-t", "--threshold", type=float, default=0.3,

help="threshold when applyong non-maxima suppression")

args = vars(ap.parse\_args())

# load the COCO class labels our YOLO model was trained on

labelsPath = os.path.sep.join([args["yolo"], "coco.names"])

LABELS = open(labelsPath).read().strip().split("\n") #Changes here

# initialize a list of colors to represent each possible class label

np.random.seed(42)

COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),

dtype="uint8")

weightsPath = os.path.sep.join([args["yolo"], "yolov3.weights"])

configPath = os.path.sep.join([args["yolo"], "yolov3.cfg"])

print("[INFO] loading YOLO from disk...")

net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

ln = net.getLayerNames() #Changes here

ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# initialize the video stream, pointer to output video file, and

# frame dimensions

vs = cv2.VideoCapture(args["input"])

writer = None

(W, H) = (None, None)

try:

prop = cv2.cv.CV\_CAP\_PROP\_FRAME\_COUNT if imutils.is\_cv2() \

else cv2.CAP\_PROP\_FRAME\_COUNT

total = int(vs.get(prop))

print("[INFO] {} total frames in video".format(total))

except:

print("[INFO] could not determine # of frames in video")

print("[INFO] no approx. completion time can be provided")

total = -1

cnt = 0

fno = 0

# ------------------FRAME PART-----------------------------------------

counter = 0

while True:

start1 = time.time()

(grabbed, frame) = vs.read()

if(cnt%2!=0):

cnt+=1

continue

fno+=1

print("Frame No:", fno)

if not grabbed:

break

# if the frame dimensions are empty, grab them

if W is None or H is None:

(H, W) = frame.shape[:2]

blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),

swapRB=True, crop=False)

net.setInput(blob)

start = time.time()

layerOutputs = net.forward(ln)

end = time.time()

boxes = []

confidences = []

classIDs = []

# loop over each of the layer outputs

for output in layerOutputs:

# loop over each of the detections

for detection in output:

# extract the class ID and confidence (i.e., probability)

# of the current object detection

scores = detection[5:]

classID = np.argmax(scores)

confidence = scores[classID]

# filter out weak predictions by ensuring the detected

# probability is greater than the minimum probability

if confidence > args["confidence"]:

box = detection[0:4] \* np.array([W, H, W, H])

(centerX, centerY, width, height) = box.astype("int")

x = int(centerX - (width / 2))

y = int(centerY - (height / 2))

# update our list of bounding box coordinates,

# confidences, and class IDs

boxes.append([x, y, int(width), int(height)])

confidences.append(float(confidence))

classIDs.append(classID)

# apply non-maxima suppression to suppress weak, overlapping

# bounding boxes

idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"],

args["threshold"])

# ensure at least one detection exists

if len(idxs) > 0:

idArray = []

for j in idxs.flatten():

if classIDs[j]==2 or classIDs[j]==0:

idArray.append(j)

flag = 0

for j in idArray:

for k in idArray:

if k==j:

continue

(x1, y1) = (boxes[j][0], boxes[j][1])

(w1, h1) = (boxes[j][2], boxes[j][3])

(x2, y2) = (x1+w1, abs(y1-h1))

(x3, y3) = (boxes[k][0], boxes[k][1])

(w2, h2) = (boxes[k][2], boxes[k][3])

(x4, y4) = (x3+w2, abs(y3-h2))

if(x1>x4 or x3>x2):

flag = 1

counter = 0

if(y1 < y4 or y3 < y2):

flag = 1

counter = 0

if flag == 0:

counter+=1

break

if flag == 0:

break

# loop over the indexes we are keeping

if(counter==4):

print("CRASH ALERT")

cv2.imwrite("output/Crash{}.jpg".format(fno), frame)

counter=0

for i in idxs.flatten():

# extract the bounding box coordinates

(x, y) = (boxes[i][0], boxes[i][1])

(w, h) = (boxes[i][2], boxes[i][3])

# draw a bounding box rectangle and label on the frame

color = [int(c) for c in COLORS[classIDs[i]]]

cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)

text = "{}: {:.4f}".format(LABELS[classIDs[i]],

confidences[i])

cv2.putText(frame, text, (x, y - 5),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2)

print(f"Box[{i}]: {x} {y} {w} {h} Labels[{i}]: {LABELS[classIDs[i]]} classIDs[{i}]: {classIDs[i]} AveragePrecision[{i}]: {confidences[i]}")

end1 = time.time()

cv2.imwrite("output/frame{}.jpg".format(fno), frame)

print("Complete time for algorithm", (end1-start1))

cnt+=1

print("[INFO] cleaning up...")

print("[Msg] Accidents frames analyised")

vs.release()

program = "wwd\_analysis.py "+args["input"]+" TRUE"

print("[INFO] Wrong Driving Analysiss Started")

subprocess.call("python "+program)

print("[INFO] Making All Frames into a video File")

subprocess.call("python playCrash.py")

print("[INFO] Video FIle Playing Started")

subprocess.call("python PlayVideo.py")

**wwdanalysis.py**

import sys

import numpy as np

import cv2

import math

import matplotlib.pyplot as plt

from numpy.linalg import inv

import time

import operator

import random

import shutil, os

from scipy import signal

######## Setting most important parameters #######

show\_images\_flag = False

if(len(sys.argv)<2):

sys.exit("please provide the name of video to be analyzed")

video = sys.argv[1]

if(len(sys.argv)>2):

show\_images\_flag = sys.argv[2]

####### To create an empty sub\_final folder everytime we run this script #######

if not os.path.exists('sub\_final'):

os.mkdir('sub\_final')

shutil.rmtree('sub\_final')

os.makedirs('sub\_final')

####### This code can be used to generate gray frames from given video #######

print("extracting frames from image")

cap = cv2.VideoCapture(video)

count = 0

while(cap.isOpened()):

ret, frame = cap.read()

if(ret):

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

else:

break

# cv2.imshow('frame',gray)

cv2.imwrite("sub\_final/frame%d.jpg" % count, gray)

count = count + 1

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

####### The above code to extract the frames from the images was given in the official documentation of opencv #######

####### Reference: https://docs.opencv.org/3.1.0/dd/d43/tutorial\_py\_video\_display.html #######

sub\_part\_of\_image1 = str(int(count\*0.65))

sub\_part\_of\_image2 = str(int(count\*0.65)+1)

path\_of\_image1 = 'sub\_final/frame'+ sub\_part\_of\_image1 + '.jpg'

path\_of\_image2 = 'sub\_final/frame'+ sub\_part\_of\_image2 + '.jpg'

####### Takes a matrix as input and converts range of matrix to 0 to 255 range #######

####### This function is usually used for printing purposes (better vizualization) #######

def change\_range(old\_array):

new\_array = np.zeros(old\_array.size).reshape(old\_array.shape[0],old\_array.shape[1])

max\_intensity = old\_array.max()

min\_intensity = old\_array.min()

for j in range(0, old\_array.shape[1]):

for i in range(0, old\_array.shape[0]):

new\_array[i][j] = ((old\_array[i][j]-min\_intensity)\*(255/(max\_intensity - min\_intensity)))

return new\_array

####### This function takes in an image matrix and smoothes its with k dimension kernal #######

def smooth\_image(img,k):

# creating a new image matrix in which smoothed values will be stored

a = img.copy()

#k = dimension of the filter

for i in range(math.floor(k/2), (img.shape[0]-math.ceil(k/2))):

for j in range(math.floor(k/2), (img.shape[1]-math.ceil(k/2))):

i\_lower = i-int(math.floor(k/2))

i\_upper = i+int(math.ceil(k/2))

j\_lower = j-int(math.floor(k/2))

j\_upper = j+int(math.ceil(k/2))

sub\_img\_flat = img[i\_lower:i\_upper,j\_lower:j\_upper].ravel().copy()

a[i][j] = np.sum(sub\_img\_flat)/(k\*k)

return a

####### This function used for reporducing edge strength and edge orientation matrix for given smoothed #######

####### matrix image. This function returns edge strenght matrix (mapped from 0 to 255) and edge #######

####### orientation matrix mapped from -90 to 90 #######

def edge\_detection(a):

img = a

# creating a new image matrix in which edge-y values will be stored

c = np.zeros(img.size).reshape(img.shape[0],img.shape[1])

c\_display = np.zeros(img.size).reshape(img.shape[0],img.shape[1])

# creating a new image matrix in which edge-x values will be stored

d = np.zeros(img.size).reshape(img.shape[0],img.shape[1])

d\_display = np.zeros(img.size).reshape(img.shape[0],img.shape[1])

# applying x and y edges on the image and storing the values in another matrix images

for j in range(1, img.shape[1]-1):

for i in range(1, img.shape[0]-1):

c[i][j] = ((-0.5)\*int(a[i-1][j]) + (0.5)\*int(a[i+1][j]) + (0)\*(int(a[i][j])))

d[i][j] = ((-0.5)\*int(a[i][j-1]) + (0.5)\*int(a[i][j+1]) + (0)\*(int(a[i][j])))

c\_display = change\_range(c)

# creating a new image matrix in whihc edge matrix values will be stored

e = np.zeros(img.size).reshape(img.shape[0],img.shape[1])

e\_display = np.zeros(img.size).reshape(img.shape[0],img.shape[1])

# finding edge magnitude values from edge x and edge y values

for j in range(1, img.shape[1]-1):

for i in range(1, img.shape[0]-1):

e[i][j] = math.sqrt((c[i][j]\*c[i][j]) + (d[i][j]\*d[i][j]))

e\_display = change\_range(e)

# creating a new image matrix in whihc edge orientation matrix values will be stored

f = np.zeros(img.size).reshape(img.shape[0],img.shape[1])

# just trying to avoid divide by zero

d[d == 0] = 0.0000000001

# finding edge orientation values from edge x and edge y values

for j in range(1, img.shape[1]-1):

for i in range(1, img.shape[0]-1):

f[i][j] = math.degrees(math.atan(c[i][j]/d[i][j]))

f\_display = change\_range(f)

return e\_display, f

####### This is a threshold function which takes in a matrix and threshold level number #######

####### and outputs all the values greater than threshold level, retains the values in matrix #######

def threshold(e\_display, level):

image = np.zeros((e\_display.shape[0], e\_display.shape[1]))

for i in range(0, e\_display.shape[0]):

for j in range(0, e\_display.shape[1]):

if(e\_display[i][j]<level):

image[i][j] = 0

else:

image[i][j] = e\_display[i][j]

return image

####### This is a histogram function used for display images only as it can sample for 255 bins only #######

####### Output is an array of size 255 which consists of number of pixels w.r.t. index of array #######

def histogram(img):

bins = 255

img\_flat = img.ravel()

max\_intensity = img\_flat.max()

min\_intensity = img\_flat.min()

hist = np.zeros(bins)

for element in img\_flat:

bin\_id = int(round((bins-1)\*((element - min\_intensity)/(max\_intensity - min\_intensity))))

hist[bin\_id] += 1

return hist

####### This is a median filter with kernal size of kXk #######

def medianfilter(img,k):

# creating a new image matrix in which smoothed values will be stored

a = img.copy()

#k is dimension of the filter

for i in range(math.floor(k/2), (img.shape[0]-math.ceil(k/2))):

for j in range(math.floor(k/2), (img.shape[1]-math.ceil(k/2))):

i\_lower = i-int(math.floor(k/2))

i\_upper = i+int(math.ceil(k/2))

j\_lower = j-int(math.floor(k/2))

j\_upper = j+int(math.ceil(k/2))

sub\_img\_flat = img[i\_lower:i\_upper,j\_lower:j\_upper].ravel().copy()

a[i][j] = np.max(sub\_img\_flat)

return a

####### custom made gaussian blur kernel which inputs image, kernel size of 1-D and sigma########

def GaussianBlur\_custom(image, size, sigma):

a = image.copy()

gaussian\_kernal = signal.get\_window(('gaussian',sigma),size\*size)

for i in range(math.floor(size/2), (image.shape[0]-math.ceil(size/2))):

for j in range(math.floor(size/2), (image.shape[1]-math.ceil(size/2))):

i\_lower = i-int(math.floor(size/2))

i\_upper = i+int(math.ceil(size/2))

j\_lower = j-int(math.floor(size/2))

j\_upper = j+int(math.ceil(size/2))

sub\_img\_flat = image[i\_lower:i\_upper,j\_lower:j\_upper].ravel().copy()

value = [a\*b for a,b in zip(sub\_img\_flat,gaussian\_kernal)]

a[i][j] = np.sum(value)/np.sum(gaussian\_kernal)

return a

# reading a first image file in black and white mode

print("Reading and resizing the first image")

img1 = cv2.imread(path\_of\_image1,0)

# resizing image to 200 X 400 size image

img1 = cv2.resize(img1, (400,200) )

# displaying all the image properties

print("Shape of First Image: ", img1.shape)

print("Minimum Intensity: ", img1.min())

print("Maximum Intensity: ", img1.max())

if(show\_images\_flag):

plt.imshow(img1)

plt.title("First Image")

plt.show()

# reading a second image file in black and white mode

print("Reading and resizing the second image")

img2 = cv2.imread(path\_of\_image2,0)

# resizing image to 200 X 400 size image

img2 = cv2.resize(img2, (400,200) )

# displaying all the image properties

print("Shape of Second Image: ", img2.shape)

print("Minimum Intensity: ", img2.min())

print("Maximum Intensity: ", img2.max())

if(show\_images\_flag):

plt.imshow(img2)

plt.title("Second Image")

plt.show()

####### Determining the degree of the vehicle on the left side of the vehicle #######

bins\_for\_histogram = 5

max\_x = 90 + bins\_for\_histogram

min\_x = -90 + bins\_for\_histogram

bin\_list = []

for i in range(min\_x, max\_x, 5):

bin\_list.append(i)

countprev = 0

value\_list = []

for i in range(len(bin\_list)):

count = 0

for j in range(len(data)):

if(data[j] < bin\_list[i]):

count += 1

value\_list.append(count - countprev)

countprev = count

# weighted\_sum\_list = [a\*b for a,b in zip(bin\_list,value\_list)]

# weighted\_angle\_value\_left = sum(weighted\_sum\_list)/sum(value\_list)

# max\_degree\_left = weighted\_angle\_value\_left

max\_index, max\_value = max(enumerate(value\_list), key=operator.itemgetter(1))

max\_degree\_left = bin\_list[max\_index]

####### Right side of the road analaysis #######

print("Analyzing the histogram of the UV directions of the right side of the road")

angles\_list = []

for i in range(0, img1.shape[0]):

for j in range(divider\_column, img1.shape[1]):

if(uv\_magr[i][j]> magnitude\_threshold):

angles\_list.append(uv\_dirr[i][j])

data = angles\_list

# fixed bin size

bins = np.arange(-200, 200, 5) # fixed bin size

if(show\_images\_flag):

plt.xlim([min(data)-5, max(data)+5])

plt.hist(data, bins=bins, alpha=0.5)

plt.title('UV orientation of right side of the bin (fixed bin size)')

plt.xlabel('UV orientation X (bin size = 5)')

plt.ylabel('count')

plt.show()

####### Determining the degree of the vehicle on the left side of the vehicle #######

bins\_for\_histogram = 1

max\_x = 90 + bins\_for\_histogram

min\_x = -90 + bins\_for\_histogram

bin\_list = []

for i in range(min\_x, max\_x, 5):

bin\_list.append(i)

countprev = 0

value\_list = []

for i in range(len(bin\_list)):

count = 0

for j in range(len(data)):

if(data[j] < bin\_list[i]):

count += 1

value\_list.append(count - countprev)

countprev = count

max\_index, max\_value = max(enumerate(value\_list), key=operator.itemgetter(1))

max\_degree\_right = bin\_list[max\_index]

print("FINAL RESULTS")

left\_correct = False

right\_correct = False

if(max\_degree\_left < 0):

print("The vehicle on the LEFT SIDE OF THE ROAD IS IN THE CORRECT DIRECTION")

left\_correct = True

else:

print("The vehicle on the LEFT SIDE OF THE ROAD IS IN THE WRONG DIRECTION")

if(max\_degree\_right > 0):

print("The vehicle on the RIGHT SIDE OF THE ROAD IS IN THE CORRECT DIRECTION")

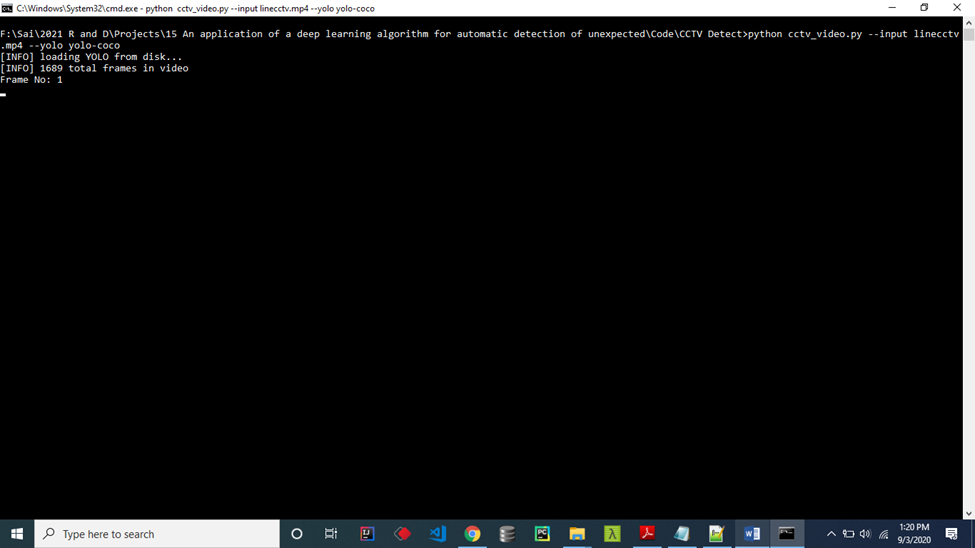
right\_correct = True

else:

print("The vehicle on the RIGHT SIDE OF THE ROAD IS IN THE WRONG DIRECTION")

**6.2 OUPUT SCREEN-SHOTS**

**Loading Training Model**

****

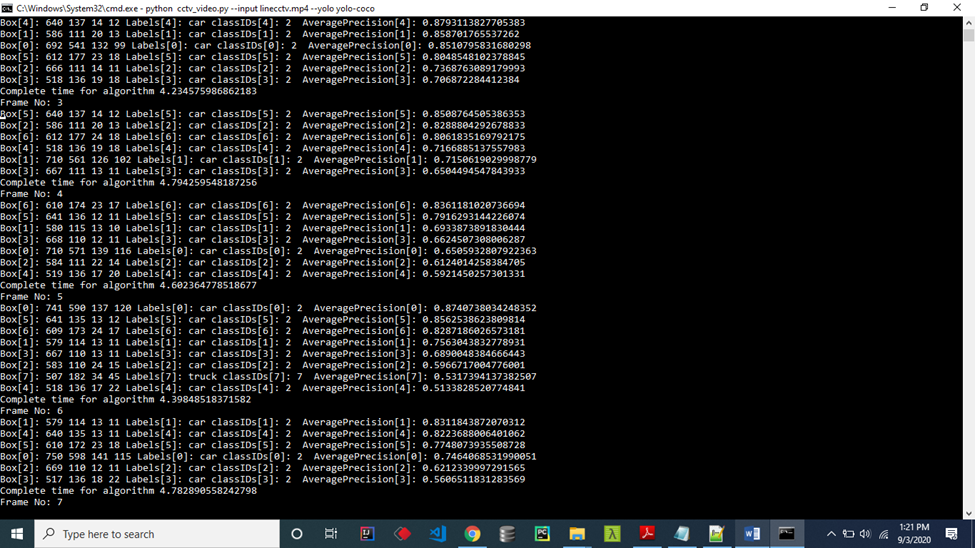
## Fig 1 : Loading Training Model

**Video Separated as Frames**

****

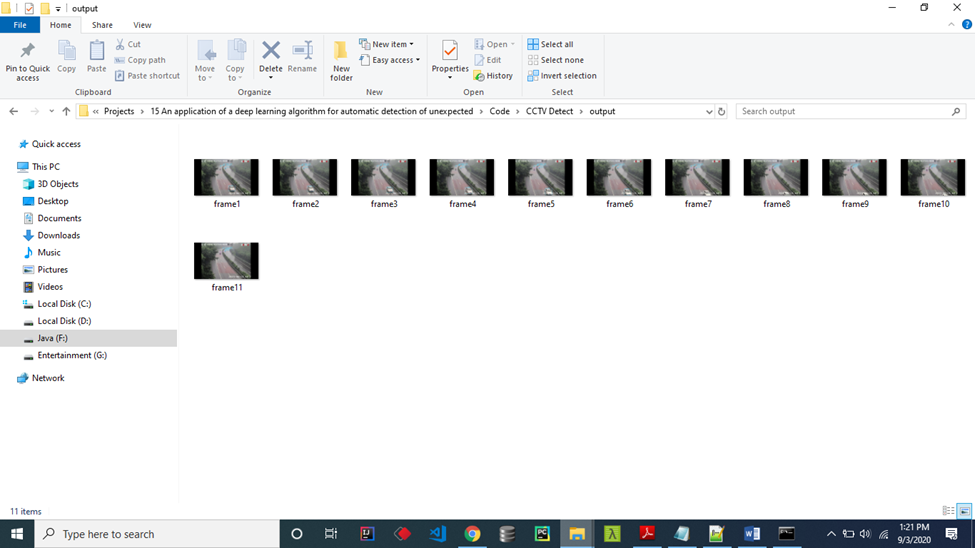
## Fig 2: Video Separated as Frames

**Analyzing Each Frame**

****

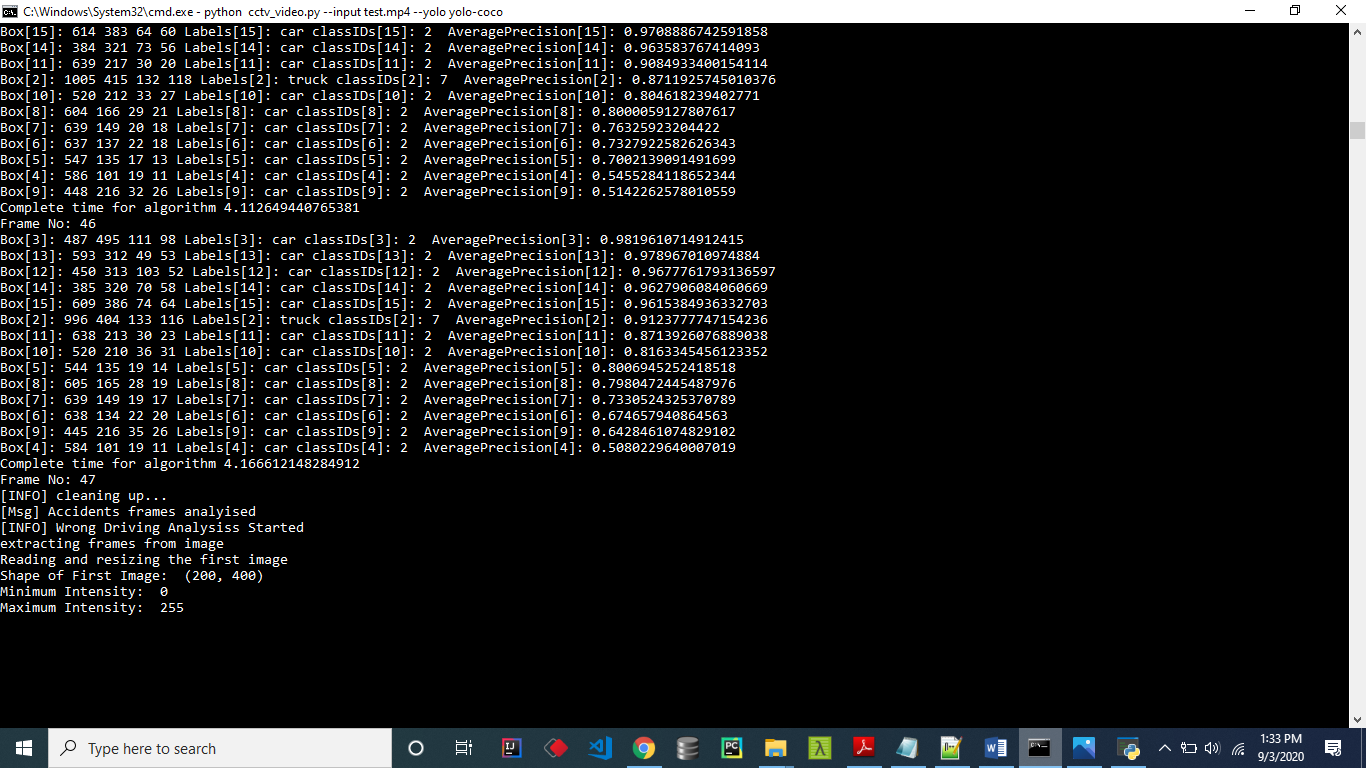
**Fig 3: Analyzing Each Frame**

**Output Frames:**

****

## Fig 4: Output Frames

**Preprocess done:**



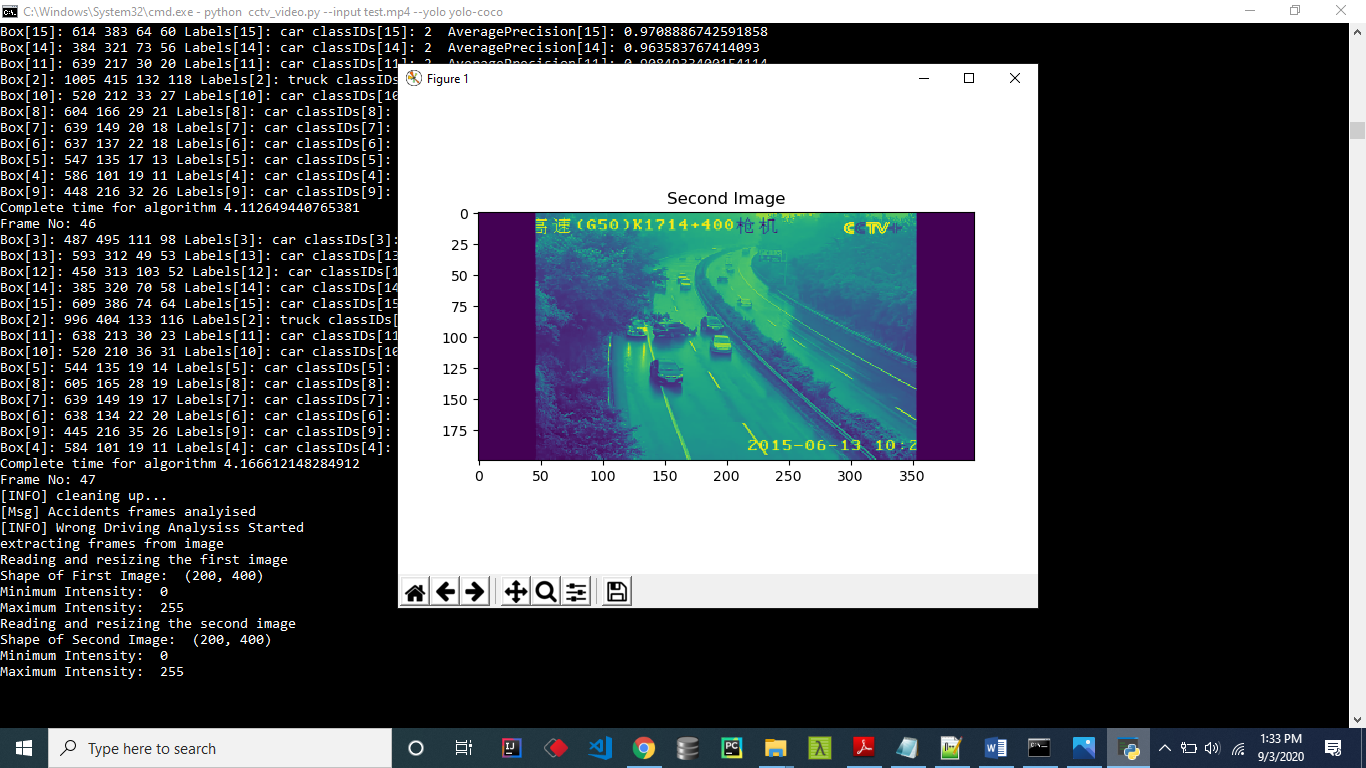
**Fig 5: Preprocess done**

**First Image Compare:**



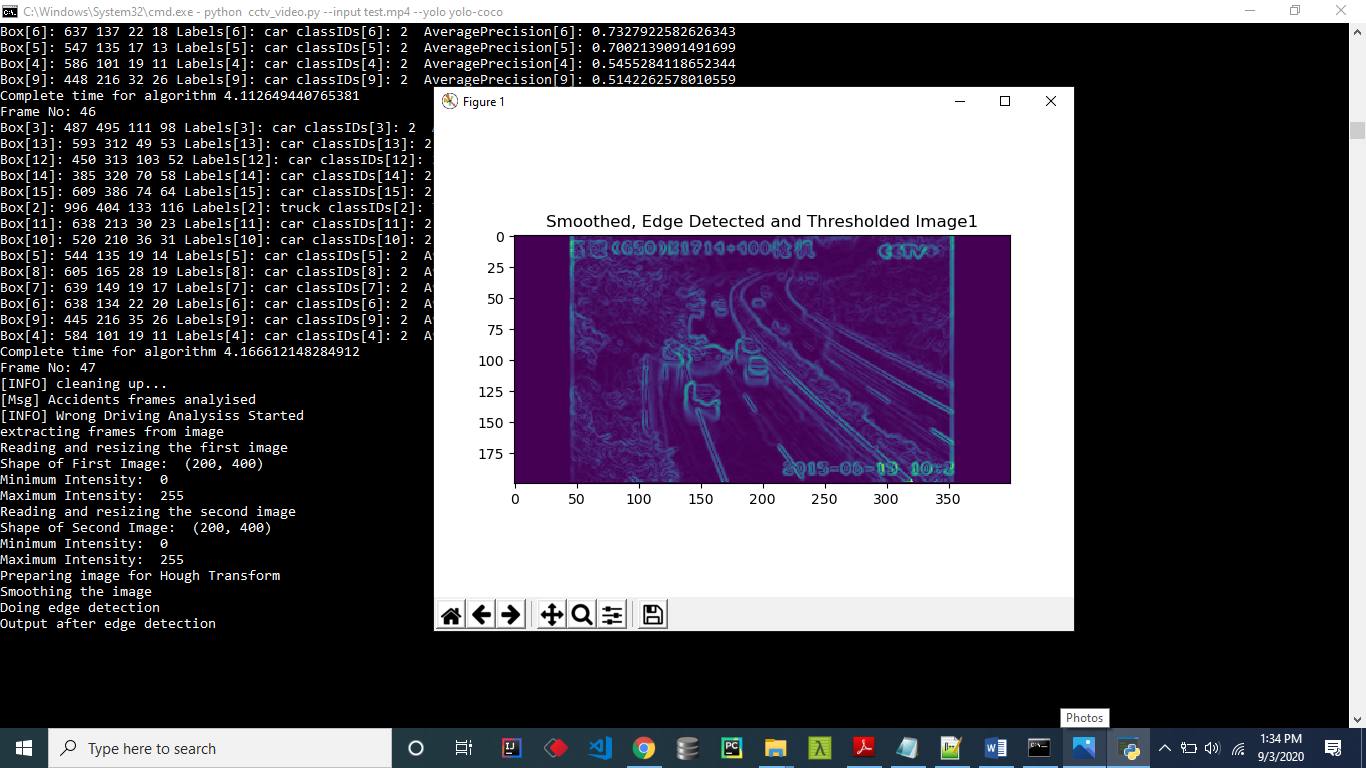
**Fig 6: First Image Compare**

**Second Image Compare:**



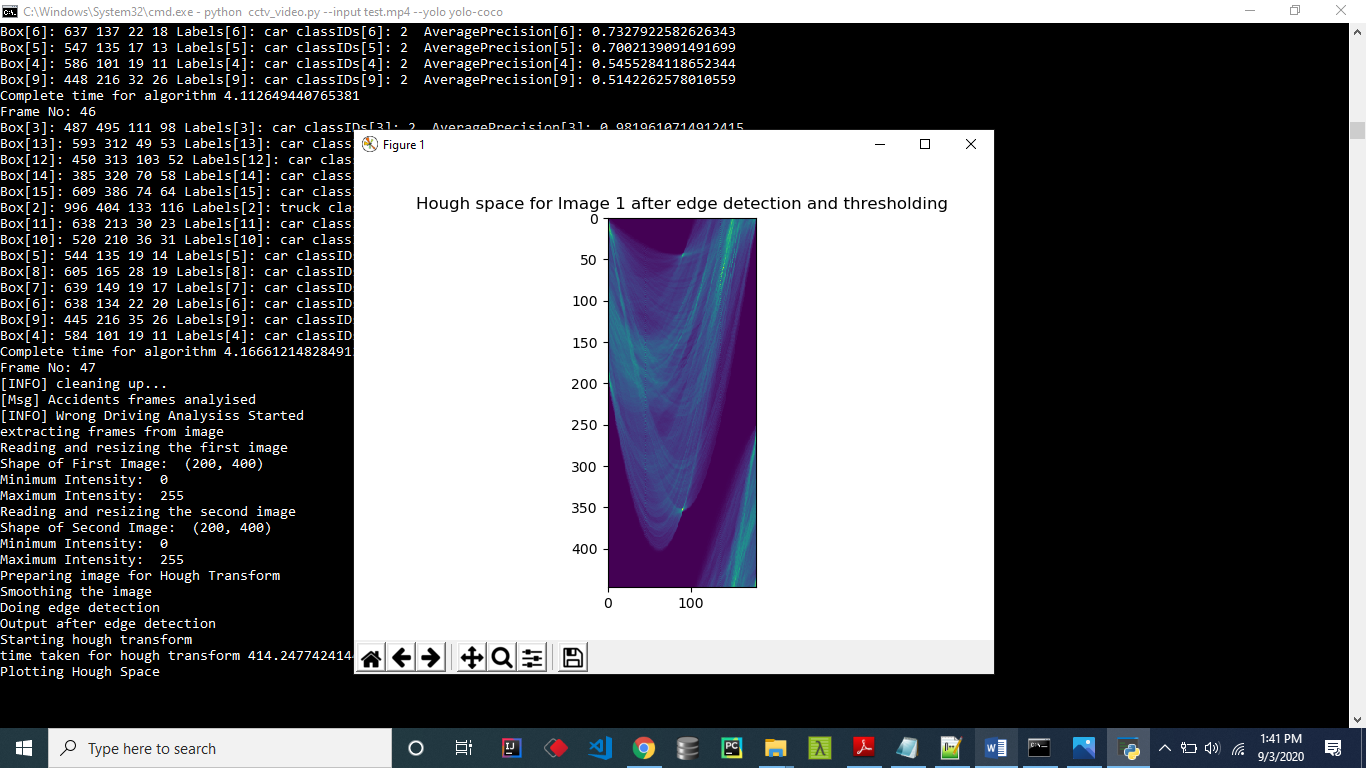
## Fig 7: Second Image Compare

**Smooth Detection:**



**Fig 8: Smooth Detection**

**Space Count:**



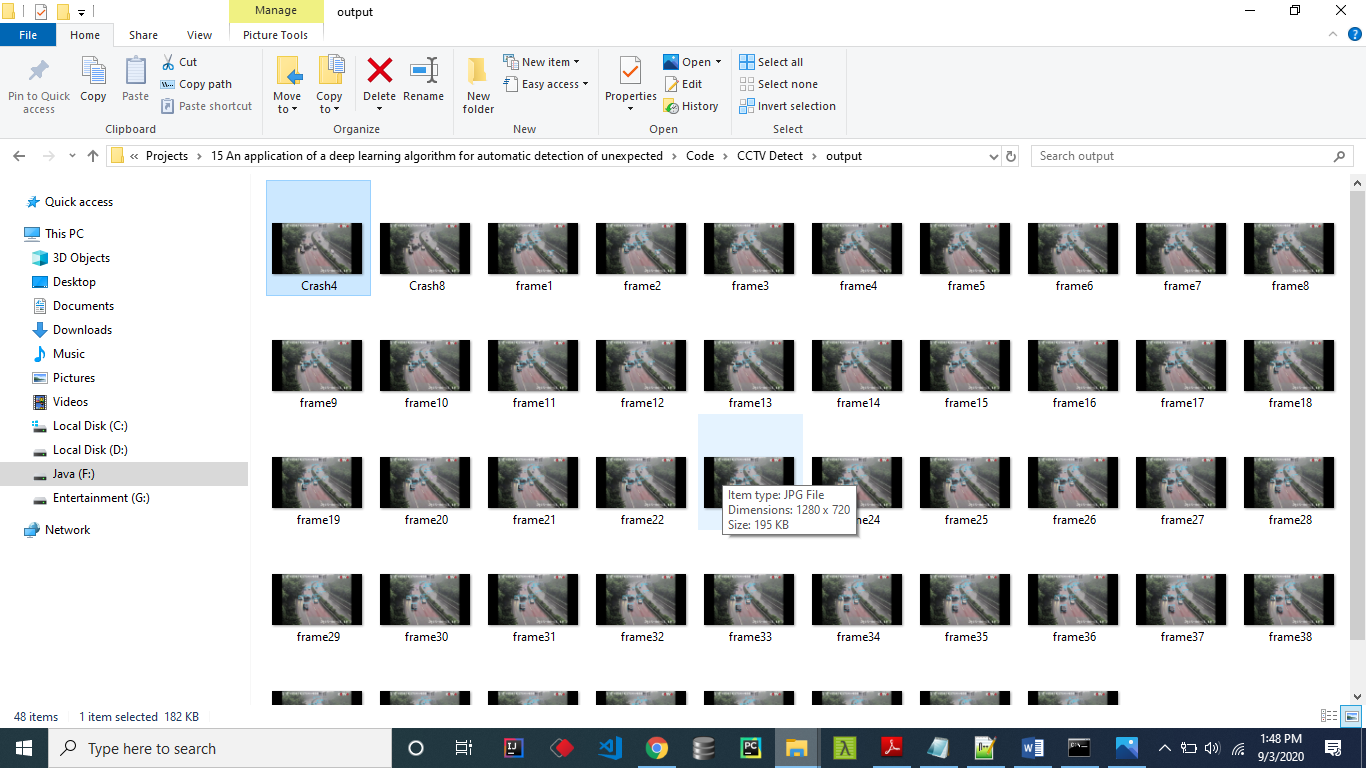
**Fig 9: Space Count**

**Making Video:**



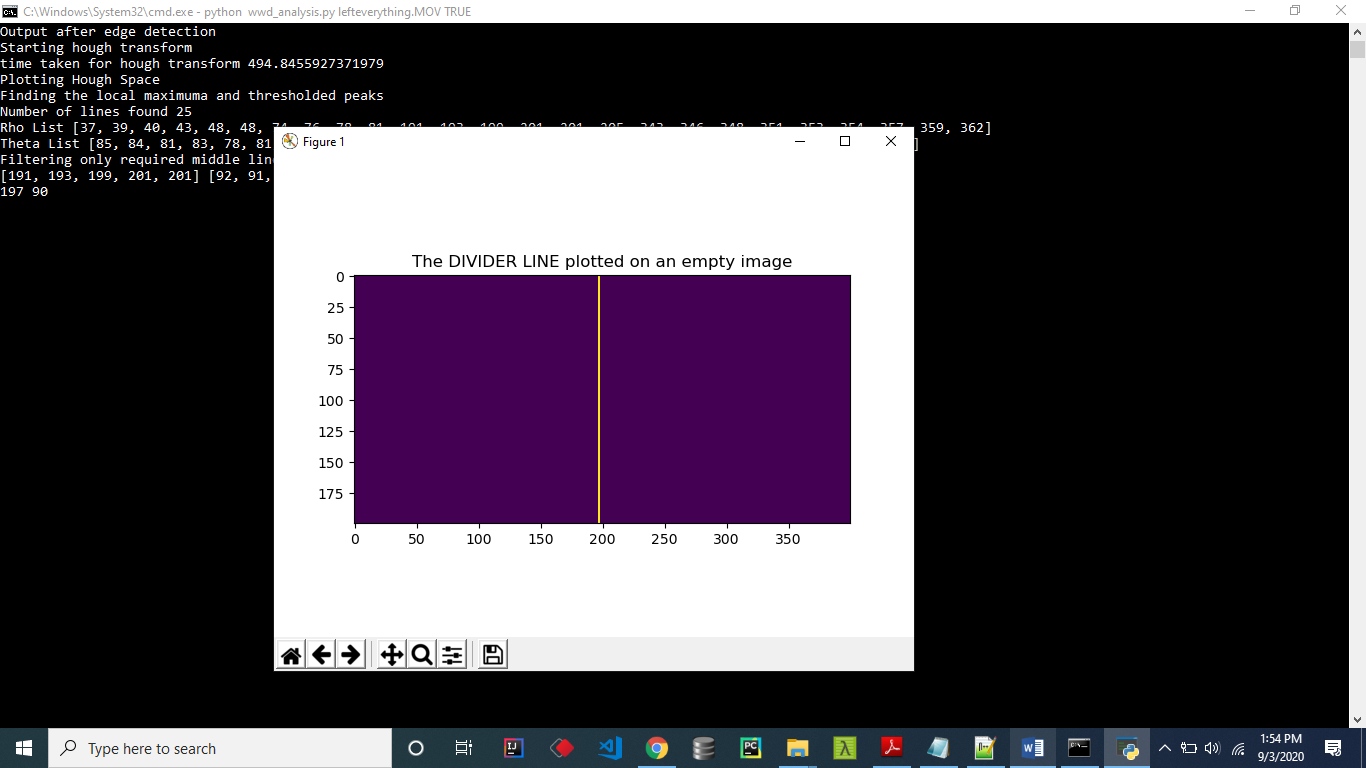
**Fig 10: Making Video**

**Crash Detected:**



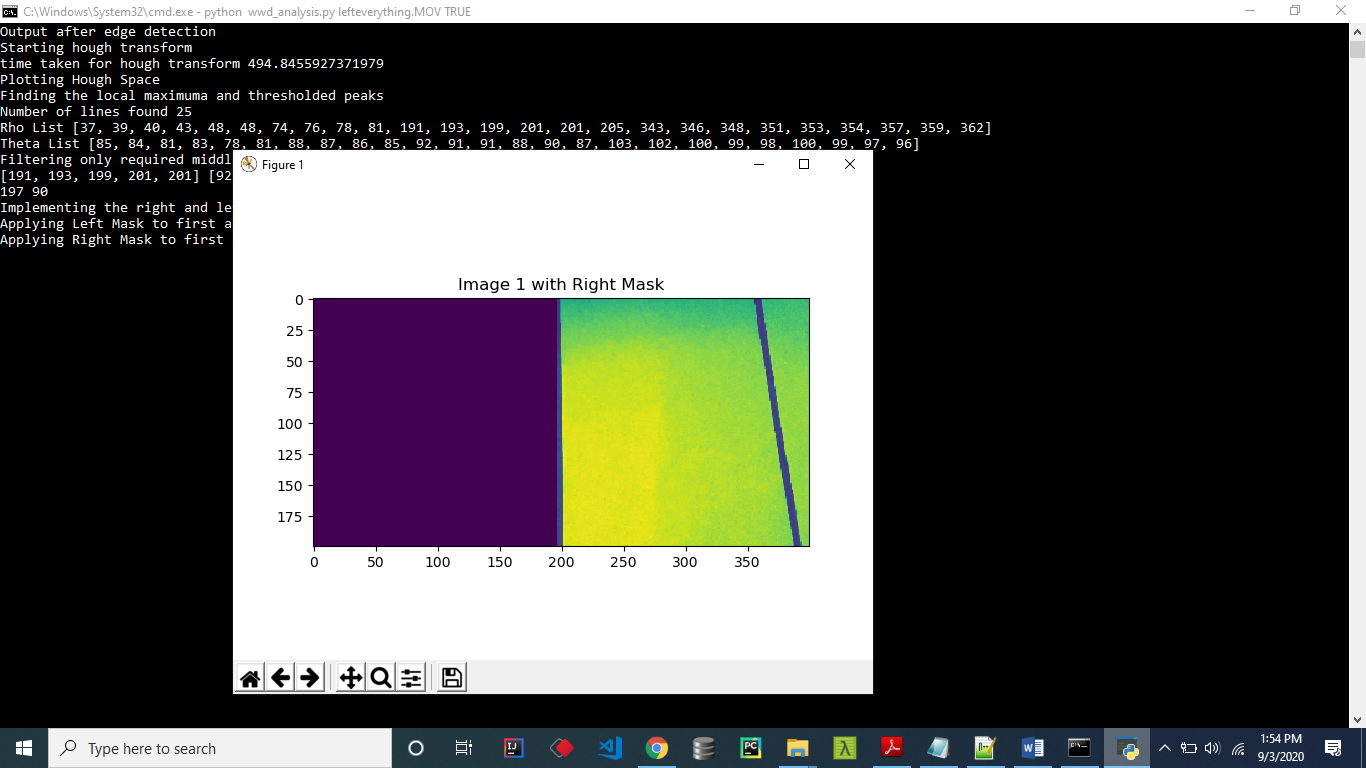
## Fig 11: Crash Detected

**Identifying driving line:**



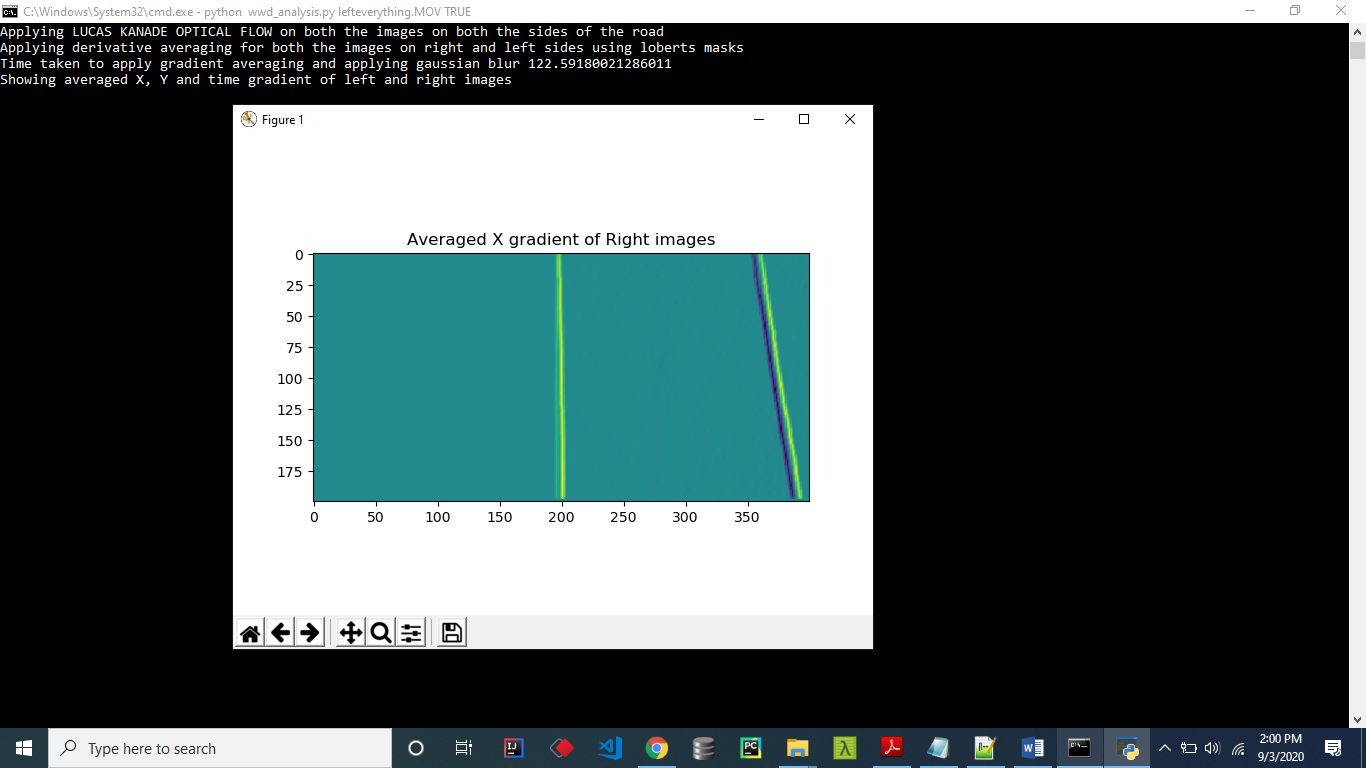
## Fig 12: Identifying driving line

**Image with left mask:**



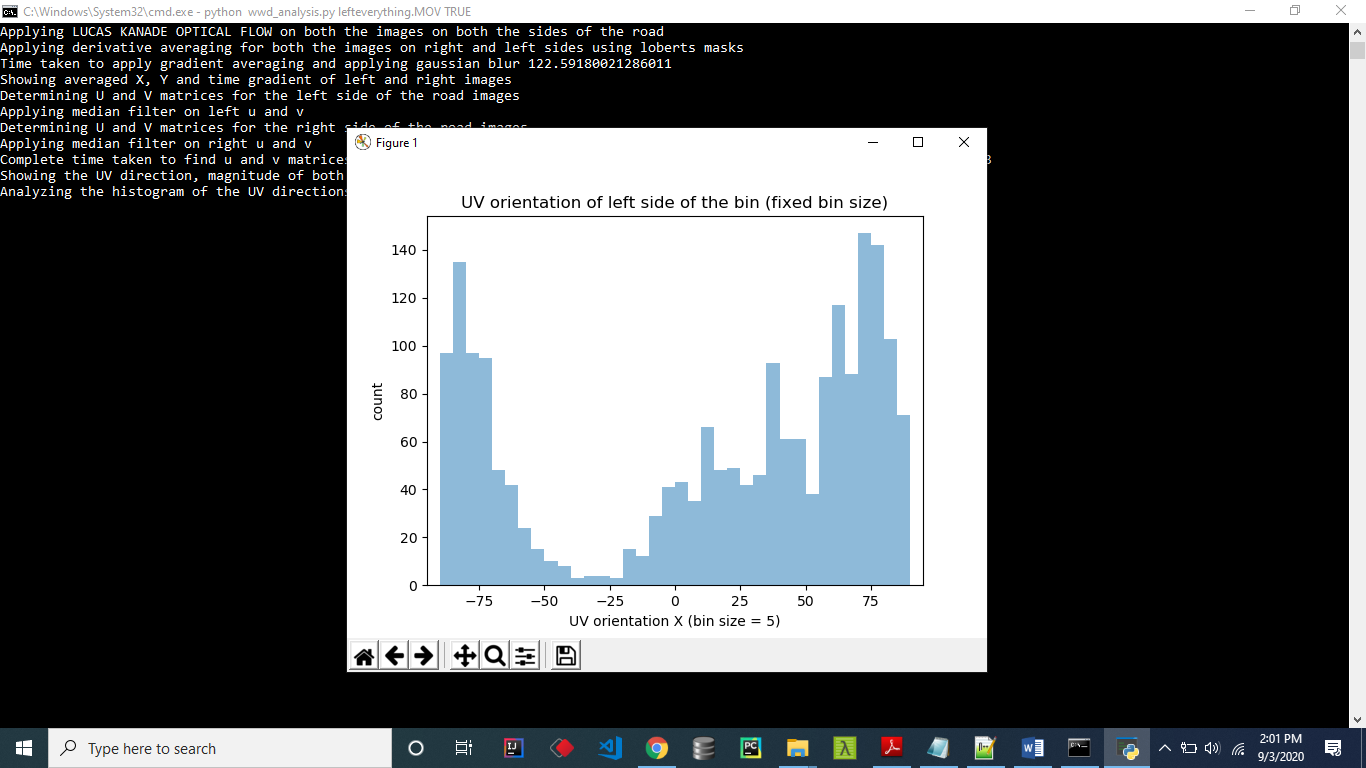
**Fig 13: Image with left mask**

**Average Gradient Image:**



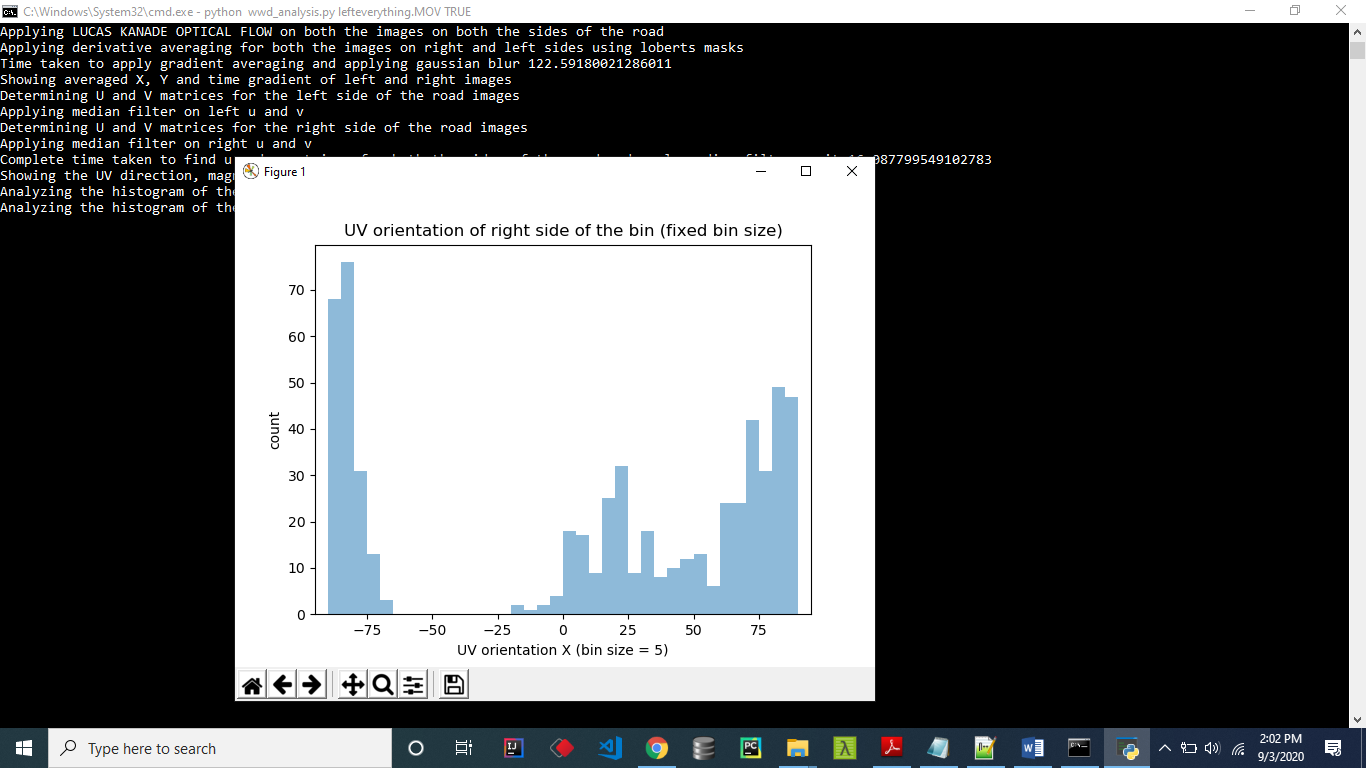
**Fig 14: Average Gradient Image**

**UV orientation:**



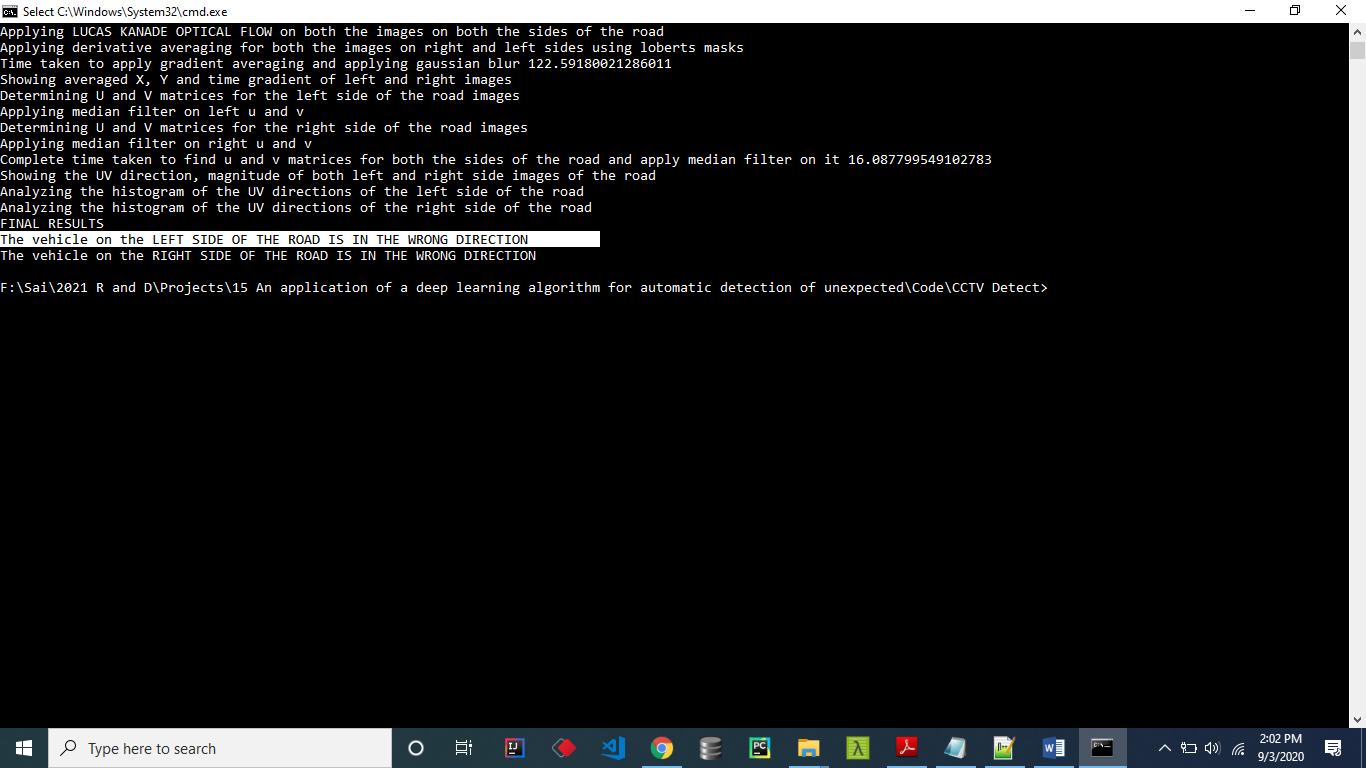
**Fig 15: UV orientation**

**UV orientation**



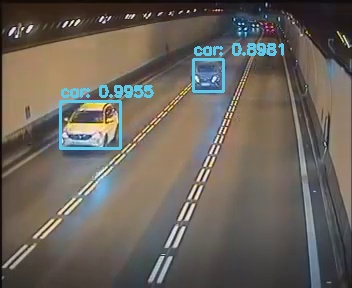
**Fig 16: UV orientation 2**

**Detect Vehicle Direction:**



**Fig 17: Detect Vehicle Direction**

**Object & Crash Detection (Tunnel):**

** **

**Fig 18: Object & Crash Detection (Tunnel)**

**TESTING**

**7.1 TESTING STRATEGY**

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

**SYSTEM TESTING**

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

**Unit Testing**

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this.The future holds a lot to offer to the development and refinement of this project.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**7.2 Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Test Case** | **Excepted Result** | **Result** |
| 1. | Loading Yolo | Yolo names and Yolo configuration loaded from disk | Pass |
| 2. | Load CCTV video file disk | Loading CCTV video through cv2 | Pass |
| 3. | Split video into images | Splitting video into images and stored in disc | Pass |
| 4. | Applying object detecting algorithm with CNN | For each image object detection algorithm applied | Pass |
| 5. | Box the detected object | Round the vehicle object with names | Pass |
| 6. | Get accident images | For moving vehicles accident Happened name with crash images | Pass |

|  |  |  |  |
| --- | --- | --- | --- |
| 7. | Detect image wrong side or not | Detecting the images wrong side driving of Vehicles | Pass |
| 8. | Identifying the vehicles lines in the road | Lines identified then only wrong driving is Detected | Pass |
| 9. | Fire Detected | User should upload a video which is contain fires | Pass |
| 10. | All images combine the a video file | Video detected and played in cv2 frames | Pass |

**CONCLUSION & FURTHER ENHANCEMENT**

**Conclusion**

This paper proposes a new process of ODTS by combining deep learning-based object detection network and object tracking algorithm, and it shows dynamic information of an object for a specific object class can be obtained and utilized. On the other hand, the object detection performance is important because SORT used in ODTS object tracking uses only information of BBox without using an image. Therefore, continuous object detection performance may be less needed unless the object tracking algorithm is relatively dependent on object recognition performance. And Tunnel CCTV Accident Detection System based on ODTS was developed. The experiments on training and evaluation of deep learning object detection network and detection of an accident of the whole system were conducted. This system adds CADA that discriminates every cycle based on dynamic information of the car objects. As a result of experimenting with the image containing each accident, it was possible to detect the accidents within 10 seconds. On the other hand, training of deep learning secured the object detection performance of a reliable Car object, and Person showed relatively low object detection performance. However, in the case of Fire, there is a high probability of false detection in the untrained videos due to the insufficient number of Fire objects. Nonetheless, it is possible to reduce the occurrence of false detections by simultaneously training objects that are No Fire.

**Further Enhancement**

The fire object detection performance of the deep learning object detection network should be improved by securing the Fire image later. Although the ODTS can be applied as an example of a Tunnel CCTV Accident Detection System, it is also used in fields that need to monitor the dynamic movement of a specific object such as vehicle speed estimation or illegal parking monitoring will be possible. To increase the reliability of the system, it is necessary to secure various images and to secure Fire and Person objects. Besides, through the application and continuous monitoring of the tunnel management site, the reliability of the system could be improved.

**REFERENCES**

[1] E. S. Lee, W. Choi, D. Kum, “Bird’s eye view localization of surrounding vehicles :Longitudinal and lateral distance estimation with partial appearance,” Robotics and Autonomous Systems, 2019, vol. 112, pp. 178-189.

[2] L. Cao, Q. Jiang, M. Cheng, C. Wang, “Robust vehicle detection by combining deep features with exemplar classification,” Neurocomputing, 2016, vol. 215, pp. 225-231.

[3] A. Arinaldi, J. A. Pradana, A. A. Gurusinga, “Detection and classification of vehicles for traffic video analytics,” Procedia computer science, 2018, vol. 144, pp. 259-268.

[4] K. B. Lee, H. S. Shin, D. G. Kim, “Development of a deep-learning based automatic tunnel incident detection system on cctvs,” in Proc. Fourth International Symposium on Computational Geomechanics, 2018, pp. 140-141.

[5] S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” in Proc. Neural Information Processing Systems, 2015, pp. 91-99.

[6] A. Bewley, Z. Zongyuan, L. Ott, F. Ramos, B. Upcroft, “Simple Online and Realtime Tracking,” in Proc. IEEE International Conference on Image Processing, 2016, pp. 3464-3468.

[7] K. B. Lee, H. S. Shin, D. G. Kim, “Development of a deep-learning based automatic tracking of moving vehicles and incident detection processes on tunnels,” Korean Tunnelling and Underground Space Association, 2018, vol. 20, no.6, pp. 1161-1175.

[8] C. Dicle, M. Sznaier, and O. Camps, “The way they move: Tracking multiple targets with similar appearance,” in International Conference on Computer Vision, 2013.

[9] S. H. Rezatofighi, A. Milan, Z. Zhang, A. Dick, Q. Shi, and I. Reid, “Joint Probabilistic Data Association Revisited,” in International Conference on Computer Vision, 2015.

[10] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, “Multiple Hypothesis Tracking Revisited,” in International Conference on Computer Vision, 2015.

[11] J. H. Yoon, M. H. Yang, J. Lim, and K. J. Yoon, “Bayesian Multi-Object Tracking Using Motion Context from Multiple Objects,” in Winter Conference on Applications of Computer Vision, 2015.

[12] A. Bewley, L. Ott, F. Ramos, and B. Upcroft, “ALExTRAC: Affinity Learning by Exploring Temporal Reinforcement within Association Chains,” in International Conference on Robotics and Automation. 2016, IEEE.

[13] L. Leal-Taixe, A. Milan, I. Reid, S. Roth, and ´ K. Schindler, “MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking,” arXiv preprint, 2015.

[14] D. Reid, “An Algorithm for Tracking Multiple Targets,” Automatic Control, vol. 24, pp. 843–854, 1979.

[15] P. Dollar, R. Appel, S. Belongie, and P. Perona, “Fast Feature Pyramids for Object Detection,” Pattern Analysis and Machine Intelligence, vol. 36, 2014.

[16] S. Oh, S. Russell, and S. Sastry, “Markov Chain Monte Carlo Data Association for General MultipleTarget Tracking Problems,” in Decision and Control. 2004, pp. 735–742, IEEE.