## A MAJOR PROJECT REPORT

**On**

## AN APPLICATION OF A DEEP LEARNING ALGORITHM FOR AUTOMATIC DETECTION OF UNEXPECTED ACCIDENTS UNDER BAD CCTV MONITORING CONDITIONS ON ROADS

*Submitted by*

## N.SNEHA (20J41A05A0)

**K.SRAVANI (20J41A0587)** **E.PAVANKUMAR (20J41A0578)** **CH.VASU (20J41A0572)**

*in partial fulfillment of the requirement for the award of degree* *of*

***Bachelor of Technology***

## In

**Computer Science and Engineering** Under the guidance of **Mr.B.Dhanaveera Pavan Kumar**

Assistant Professor Computer Science And Engineering

## Department of Computer Science and Engineering MALLA REDDY ENGINEERING COLLEGE

(An UGC Autonomous Institution, Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad) Maisammaguda, Secunderabad, Telangana, India 500100

#### MAY-2024

**MALLA REDDY ENGINEERING COLLEGE**

Maisammaguda, Secunderabad, Telangana, India 500100



**BONAFIDE CERTIFICATE**

This is to certify that this major project work entitled **“AN APPLICATION OF DEEP LEARNING ALGORITHM FOR AUTOMATIC DETECTION OF UNEXPECTED ACCIDENTS UNDER BAD CCTV MONITORING CONDITIONS ON ROADS ”,** submitted by **N.SNEHA (20J41A05A0) , K.SRAVANI (20J41A0587), E.PAVANKUMAR(20J41A0578),CH.VASU(20J41A0572)** to Malla Reddy Engineering College affiliated to JNTUH, Hyderabad in partial fulfillment for the award of **Bachelor of Technology** in **Computer Science and Engineering** is a bonafide record of project work carried out under my/our supervision during the academic year 2023-2024 and that this work has not been submitted elsewhere for a degree.

#### SIGNATURE SIGNATURE

Mr.B.Dhanaveera Pavan Kumar

#### SUPERVISOR

**Assistant Professor** Computer Science And Engineering Malla Reddy Engineering College

Secunderabad, 500 100

Dr.PSRC Murthy

#### PROFESSOR

#### HOD

Computer Science And Engineering Malla Reddy Engineering College Secunderabad, 500 100

#### Submitted for Major Project viva-voice examination held on

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**MALLA REDDY ENGINEERING COLLEGE**

Maisammaguda, Secunderabad, Telangana, India 500100

**ACKNOWLEDGEMENT**

We would like to express our sincere thanks to our **Dr. A. Ramaswami Reddy, Principal, MREC(A)** for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr.PSRC Murthy, Professor & Head**, **Department of CSE, MREC(A)** for all the timely support and valuable suggestions during the period of our project.

We are thankful to our Project Coordinator **Dr. K. Arun Kumar**, Associate Professor, Department of Computer Science and Engineering, for his cooperation during the project work.

We are extremely thankful and indebted to our internal guide, **Mr.B.Dhanaveera Pavan Kumar, Assistant Professor, Department of CSE, MREC(A)** for his constant guidance, encouragement and moral support throughout the project.

Finally, We would also like to thank all the faculty and staff of the CSE Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

N.SNEHA 20J41A05A0 K.SRAVANI 20J41A0587 E.PAVAN 20J41A0578 CH.VASU 20J41A0572

## ABSTRACT

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 (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 BBoxs 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 event images in tunnels to 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 which including each accident. As a result, the system can detect all accidents within 10 seconds. The more important point 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

**ABSTRACT**

iv

**LIST OF TABLES** v-vi

**LIST OF TABLES** v-vi

**LIST OF FIGURES** vii-viii

**LIST OF ABBREVIATIONS** ix

|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter** | **Description** | | **Page** |
|  |  | | **No.** |
| **1** | **INTRODUCTION** | | **1-6** |
| **2** | **LITERATURE SURVEY** | |  |
| **3** | **SYSTEM ANALYSIS** | | **7-9** |
|  | 3.1. Existing System | | 7 |
|  | 3.1.2. Existing System Disadvantages | | 7 |
|  | 3.2. Proposed System | | 8-9 |
|  | 3.2.1. Proposed System Advantages | | 9 |
| **4** | **SYSTEM REQUIREMENTS** | | **10-11** |
|  | 4.1.Software Requirements | | 10 |
|  | 4.2.Hardware Requirements | | 11 |
| 5 | **SYSTEM DESIGN** | | **12-17** |
|  | 5.1.System Architecture | | 12 |
|  | 5.2.Data Flow Diagram | | 12-13 |
|  | 5.3.UML Diagrams  5.3.1.UseCase Diagram  5.3.2.Class Diagram  5.3.3.Sequence Diagram  5.3.4.Activity Diagram | | 13-17  14  15  16  17 |
| 6 | **IMPLEMENTATION** | | **18-29** |
|  | 6.1.Python | | 18-25 |
|  | 6.2.Modules | | 26-37 |
|  | 6.2.1.User | | 26 |
|  | 6.2.2.Object Detection and Tracking | | 26 |
|  | 6.2.3.RCNN | | 26 |
|  | 6.2.4.Average Precision | | 27 |
|  | 6.3.Input And Output Devices | | 28-29 |
| 7 | **OUTPUT** | | **30-39** |
|  | 7.1.Starting Project | | 30 |
|  | 7.2.Loading Training Model | | 31 |
|  | 7.3.Video Separated as Frames | | 31 |
|  | 7.4.Starting each Frames | | 32 |
|  | 7.5.Output Frames | | 32 |
|  | 7.6.Object Detection | | 33 |
|  | 7.7.Preprocess done | | 33 |
|  | 7.8. First Image Compare | | 34 |
|  | 7.9.Second Image Compare | | 34 |
|  | 7.10.Smooth Detection  7.11.Space Count  7.12.Making video  7.13.Crash Detected  7.14.Identifying Driving Line  7.15.Image with left mask  7.16.Average Gradient Image  7.17.UV orientation  7.18.Detect VehicleDirection | | 35  35  36  36  37  37  38  39  39 |
| 8 | **SYSTEM TESTING** | | **40-42** |
|  | 8.1.TYPES OF TESTS | |  |
|  | 8.1.1.Unit Testing | | 40 |
|  | 8.1.2.Integration Testing | | 40 |
|  | 8.1.3.Functional Testing | | 40 |
|  | 8.1.4.System Testing | | 41 |
|  | 8.1.5.WhiteBox Testing | | 41 |
|  | 8.1.6.BlackBox Testing | | 41 |
|  | 8.1.7.Acceptance Testing | | 42 |
| 9 | **CONCLUSION AND FUTURE ENHANCEMENT** | | **43-44** |
|  | 9.1.Conclusion | | 43 |
|  | 9.2.Future Enhancement | | 44 |
|  |  | |  |
| 10 | **REFERENCES** | | 45-47 |
|  |  | |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **LIST OF FIGURES**  **TITLE** | **PAGE NO.** |
| 5.1 | Architecture Diagram | 12 |
| 5.2 | Data Flow Diagram | 13 |
| 5.3.1 | Use Case Diagram | 14 |
| 5.3.2  5.3.3  5.3.4  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9  7.10 | Class Diagram  Sequence Diagram  Activity Diagram  Starting Project  Loading Training Model  Video Separated as Frames  Starting each Frames  Output Frames  Object Detection  Preprocess done  First Image Compare  Second Image Compare  Smooth Detection | 15  16  17  30  31  31  32  32  33  33  34  35  35 |  |  |

7.11Space Count 35

7.12 Making video 36

7.13 Crash Detected 36

7.14 Identifying Driving Line 37

7.15 Image with left mask 37

7.16 Average Gradient Image 38

7.17-1 UV orientation 39

7.17-2 UV orientation 39

7.18 Detect VehicleDirection 39

**LIST OF ABBREVIATIONS**

CNN- Convolutional Neural Network

SVM- Support Vector Machine

RCNN- Region-Based Convolutional Neural Network

ODTS- Object Detection & Tracking System

MOT- Multiple Object Tracking

MHT- Multiple Hypothesis Tracking

JPDA- Joint Probabilistic Data Association

ACF- Aggregate Channel Filter

DNN- Deep Convolutional Neural Network

WWD- Wrong-Way Driving

BBOX- Bounding Box

AP- Average Precision

CASH- Combined Algorithm Selection and Hyperparameter

DFD-Data Flow Diagram

UML- Unified Modeling Language

## CHAPTER-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. However, they all were hard to assign unique IDs to the detected objects and track them by keeping the same ID over time.

Therefore, in this paper, an attempt is made for generate an object detection & tracking system (ODTS), that can obtain moving information of target objects by combining object tracking algorithm with the deep learning-based object detection process. The full ODTS procedures will be described in details in the following section. Also, the tunnel accident detection system in the framework of ODTS will be taken into consideration. This system is used for detecting accident or unexpected events taking place on moving object and target local region on CCTV.

A lean implementation of a tracking-by detection framework for the problem of multiple object tracking (MOT) where objects are detected each frame and represented as bounding boxes. In contrast to many batch based tracking approaches, this work is primarily targeted toward online tracking where only detections from the previous and the current frame are presented to the tracker. Additionally, a strong emphasis is placed on efficiency for facilitating realtime tracking and to promote greater uptake in applications such as pedestrian tracking for autonomous vehicles. The MOT problem can be viewed as a data association problem where the aim is to associate detections across frames in a video sequence. To aid the data association process, trackers use various methods for modelling the motion and appearance of objects in the scene.

The methods employed by this paper were motivated through observations made on a recently established visual MOT benchmark. Firstly, there is a resurgence of mature data association techniques including Multiple Hypothesis Tracking (MHT) and Joint Probabilistic Data Association (JPDA) which occupy many of the top positions of the MOT benchmark. Secondly, the only tracker that does not use the Aggregate Channel Filter (ACF) detector is also the top-ranked tracker, suggesting that detection quality could be holding back the other trackers. Furthermore, the trade-off between accuracy and speed appears quite pronounced, since the speed of most accurate trackers is considered too slow for realtime applications. With the prominence of traditional data association techniques among the top online and batch trackers along with the use of different detections used by the top tracker, this work explores how simple MOT can be and how well it can perform. Keeping in line with Occam’s Razor, appearance features beyond the detection component are ignored in tracking and only the bounding box position and size are used for both motion estimation and data association. Furthermore, issues regarding short-term and long-term occlusion are also ignored, as they occur very rarely and their explicit treatment intro duces undesirable complexity into the tracking framework.

We argue that incorporating complexity in the form of object re-identification adds significant overhead into the tracking framework – potentially limiting its use in realtime applications. This design philosophy is in contrast to many proposed visual trackers that incorporate a myriad of components to handle various edge cases and detection errors. This work instead focuses on efficient and reliable handling of the common frame-to-frame associations. Rather than aiming to be robust to detection errors, we instead exploit recent advances in visual object detection to solve the detection problem directly. This is demonstrated by comparing the common ACF pedestrian detector with a recent convolutional neural network (CNN) based detector. This minimalistic formulation of tracking facilitates both efficiency and reliability for online tracking, In this paper, this approach is only applied to tracking pedestrians in various environments, however due to the flexibility of CNN based detectors, it naturally can be generalized to other objects classes. The main contributions of this paper are:

• We leverage the power of CNN based detection in the context of MOT.

• A pragmatic tracking approach based on the Kalman filter and the Hungarian algorithm is presented and evaluated on a recent MOT benchmark.

• Code will be open sourced to help establish a baseline method for research experimentation and uptake in collision avoidance applications.

Traditionally MOT has been solved using Multiple Hypothesis Tracking (MHT) or the Joint Probabilistic Data Association (JPDA) filters, which delay making difficult decisions while there is high uncertainty over the object assignments. The combinatorial complexity of these approaches is exponential in the number of tracked objects making them impractical for real time applications in highly dynamic environments. Recently, Rezatofighi et al., revisited the JPDA formulation in visual MOT with the goal to address the combinatorial complexity issue with an efficient approximation of the JPDA by exploiting recent developments in solving integer programs. Similarly, Kim et al. used an appearance model for each target to prune the MHT graph to achieve state-of-the-art performance. However, these methods still delay the decision making which makes them unsuitable for online tracking. Many online tracking methods aim to build appearance models of either the individual objects themselves or a global model through online learning. In addition to appearance models, motion is often incorporated to assist associating detections to tracklets. When considering only one-to-one correspondences modelled as bipartite graph matching, globally optimal solutions such as the Hungarian algorithm can be used. The method by Geiger et al. uses the Hungarian algorithm in a twostage process. First, tracklets are formed by associating detections across adjacent frames where both geometry and appearance cues are combined to form the affinity matrix.

**CHAPTER 2**

**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. The proposed system successfully performs partially visible vehicle detection and tracking. 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.

# **4) Development of a deep-learning based automatic tunnel incident**

# **detection system on cctvs**

**AUTHORS** : **K. B. Lee, H. S. Shin, D. G. Kim**

In this paper, 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 (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 BBoxs 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 event images in tunnels to 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 which including each accident. As a result, the system can detect all accidents within 10 seconds. The more important point is that the detection capacity of ODTS could be enhanced automatically without any changes in the program codes as the training dataset becomes rich.

**5)Online Self-Supervised Multi-Instance Segmentation of Dynamic Objects**

**AUTHORS**: A. Bewley, V. Guizilini, F. Ramos, and B. Upcroft

This paper presents a method for the continuous segmentation of dynamic objects using only a vehicle mounted monocular camera without any prior knowledge of the object's appearance. Prior work in online static/dynamic segmentation [1] is extended to identify multiple instances of dynamic objects by introducing an unsupervised motion clustering step. These clusters are then used to update a multi-class classifier within a self-supervised framework. In contrast to many tracking-by-detection based methods, our system is able to detect dynamic objects without any prior knowledge of their visual appearance shape or location. Furthermore, the classifier is used to propagate labels of the same object in previous frames, which facilitates the continuous tracking of individual objects based on motion. The proposed system is evaluated using recall and false alarm metrics in addition to a new multi-instance labelled dataset to measure the performance of segmenting multiple instances of objects.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**EXISTING SYSTEM:**

In the existing system the data preprocess has dine with structured data. Even though data pre-processing consumes a large chunk of time in an ML pipeline, it is astonishing to see the inadequate amount of work done to automate it. For data preprocessing, it can be noted that while the data pre process approaches are adequate for structured data, work still needs to be done to assimilate on Structured data. We suggest the incorporation of data-mining methods as they can deal with such unformed data. This can allow AutoML pipelines to create models capable of learning from Internet sources. In feature engineering, it should be noted that most methods used until now adhere to supervised learning. However, dataset specificity is high, and therefore, AutoML pipelines should be as generic as possible to accommodate the diverse datasets. Therefore, a gradual paradigm shift towards unsupervised.

**DISADVANTAGES OF EXISTING SYSTEM:**

* Feature Generation is not up to the mark where domain experts excepted results.
* Most AutoML tools emphasize the performance but in the real world, that’s just one aspect being covered in machine learning projects. So the companies can’t compromise the computing plus storage specification sheet.
* CASH(Combined Algorithm Selection and Hyperparameter) problem considers model selection and hyperparameters optimization as a single hierarchical parameter.
* **Algorithm**: SmartML,J48,C50

**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:**

The proposed System aims at providing an overview of the advances seen in the realm of AutoML in recent years. We focus on individual aspects of AutoML and summarize the improvements achieved in recent years. The motivation of proposed system stems from the unavailability of a compact study of the current state of AutoML. While we acknowledge the existence of other surveys, their motive is to either provide an in-depth understanding of a particular segment of AutoML, provide just an experimental comparison of various tools used or are fixated towards deep learning models.

**ADVANTAGES OF PROPOSED SYSTEM:**

* We segment the AutoML pipeline into parts and review the contributions in each of these segments.
* We explore the various state-of-the-art tools currently available for AutoML and evaluate them.
* We also incorporate the advancements seen in machine learning which seems to be overshadowed by deep learning in recent years.

**Algorithm**:H2O-AutoML, Linear Regression, Gradient Boosting Regressor

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

## CHAPTER 4

**SYSTEM REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

* **System :** Intel i3
* **Hard Disk :** 1 TB.
* **Monitor** : 14’ Colour Monitor.
* **Mouse :** Optical Mouse.
* **Ram :** 4GB.

**SOFTWARE REQUIREMENTS:**

* **Operating system :** Windows 10.
* **Coding Language :** Python.
* **Front-End :** Html. CSS
* **Designing :** Html,css,javascript.
* **Data Base :** SQLite.

**REQUIREMENT ANALYSIS**

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

**REQUIREMENT SPECIFICATION**

**Functional Requirements**

* Graphical User interface with the User.

**Software Requirements**

For developing the application the following are the Software Requirements:

1. Python

**Operating Systems supported**

1. Windows 10 64 bit OS

**Technologies and Languages used to Develop**

1. Python

**Debugger and Emulator**

* Any Browser (Particularly Chrome)

**Hardware Requirements**

For developing the application the following are the Hardware Requirements:

* Processor: Intel i7
* RAM: 16 GB

Space on Hard Disk: minimum 1

## CHAPTER 5

## SYSTEM DESIGN

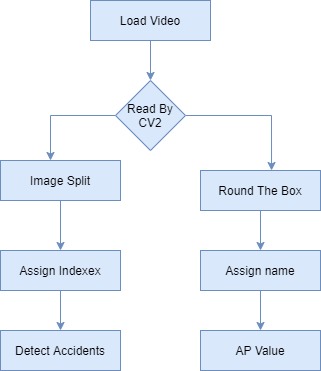
**SYSTEM ARCHITECTURE:**

****

**FIG: 5.1 SYSTEM ARCHITECTURE**

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



**FIG: 5.2 DATA FLOW DIAGRAM**

**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 object-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.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**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. 

**FIG:5.3.1 USECASE DIAGRAM**

**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.



**FIG: 5.3.2 CLASS DIAGRAM**

**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. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**FIG: 5.3.3 SEQUENCE DIAGRAM**

**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

## 

## FIG:5.3.4 ACTIVITY

## DIAGRAM

## CHAPTER 6

**IMPLEMENTATION**

**PYTHON**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython" \o "CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

**Interactive Mode Programming**

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>

Type the following text at the Python prompt and press the Enter −

>>> print "Hello, Python!"

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");.However in Python version 2.4.3, this produces the following result −

Hello, Python!

**Script Mode Programming**

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

print "Hello, Python!"

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

$ python test.py

This produces the following result −

Hello, Python!

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

#!/usr/bin/python

print "Hello, Python!"

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

$ chmod +x test.py # This is to make file executable

$./test.py

This produces the following result −

Hello, Python!

**Python Identifiers**

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

**Reserved Words**

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and exec not

assert finally or

break for pass

class from print

continue global raise

def if return

del import try

elif in while

else is with

except lambda yield

**Lines and Indentation**

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

if True:

print "True"

else:

print "False"

However, the following block generates an error −

if True:

print "Answer"

print "True"

else:

print "Answer"

print "False"

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

#!/usr/bin/python

import sys

try:

# open file stream

file = open(file\_name, "w")

except IOError:

print "There was an error writing to", file\_name

sys.exit()

print "Enter '", file\_finish,

print "' When finished"

while file\_text != file\_finish:

file\_text = raw\_input("Enter text: ")

if file\_text == file\_finish:

# close the file

file.close

break

file.write(file\_text)

file.write("\n")

file.close()

file\_name = raw\_input("Enter filename: ")

if len(file\_name) == 0:

print "Next time please enter something"

sys.exit()

try:

file = open(file\_name, "r")

except IOError:

print "There was an error reading file"

sys.exit()

file\_text = file.read()

file.close()

print file\_text

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

total = item\_one + \

item\_two + \

item\_three

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

days = ['Monday', 'Tuesday', 'Wednesday','Thursday', 'Friday']

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

word = 'word'

sentence = "This is a sentence."

paragraph = """This is a paragraph. It is

made up of multiple lines and sentences."""

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

#!/usr/bin/python

# First comment

print "Hello, Python!" # second comment

This produces the following result −

Hello, Python!

You can type a comment on the same line after a statement or expression −

name = "Madisetti" # This is again comment

You can comment multiple lines as follows −

# This is a comment.

# This is a comment, too.

# This is a comment, too.

# I said that already.

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

'''

This is a multiline

comment.

'''

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

#!/usr/bin/python

raw\_input("\n\nPress the enter key to exit.")

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon ( ; ) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

import sys; x = 'foo'; sys.stdout.write(x + '\n')

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon ( : ) and are followed by one or more lines which make up the suite. For example −

if expression :

suite

elif expression :

suite

else :

suite

**Command Line Arguments**

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

$ python -h

usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...

Options and arguments (and corresponding environment variables):

-c cmd : program passed in as string (terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

**Python Lists**

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5];

list3 = ["a", "b", "c", "d"]

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

tup1 = ('physics', 'chemistry', 1997, 2000);

**MODULES:**

* **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. Video-Capture(Videfile path) means without camera we can load the pre-recorded video 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 identified human 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.

**R-CNN(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.

**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.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

**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.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

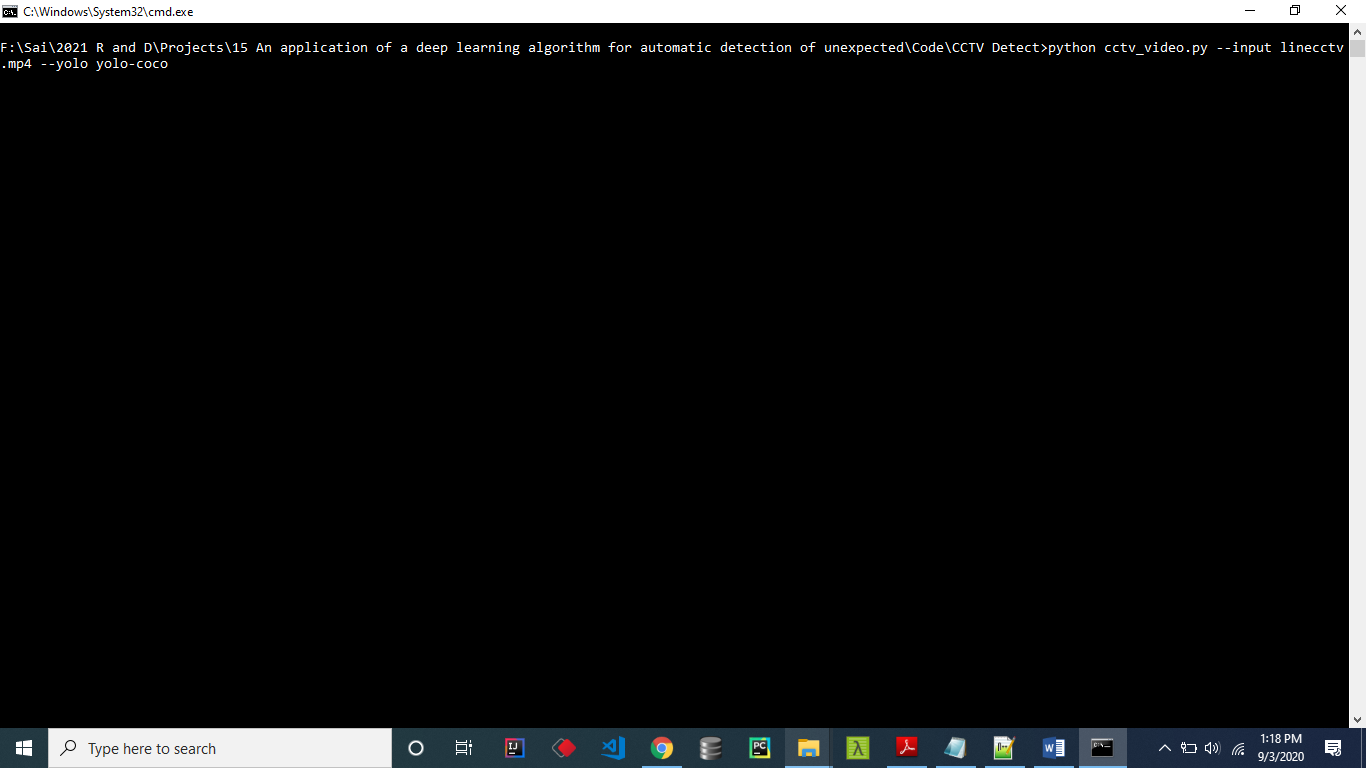
3.Create document, report, or other formats that contain information produced by the system.

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

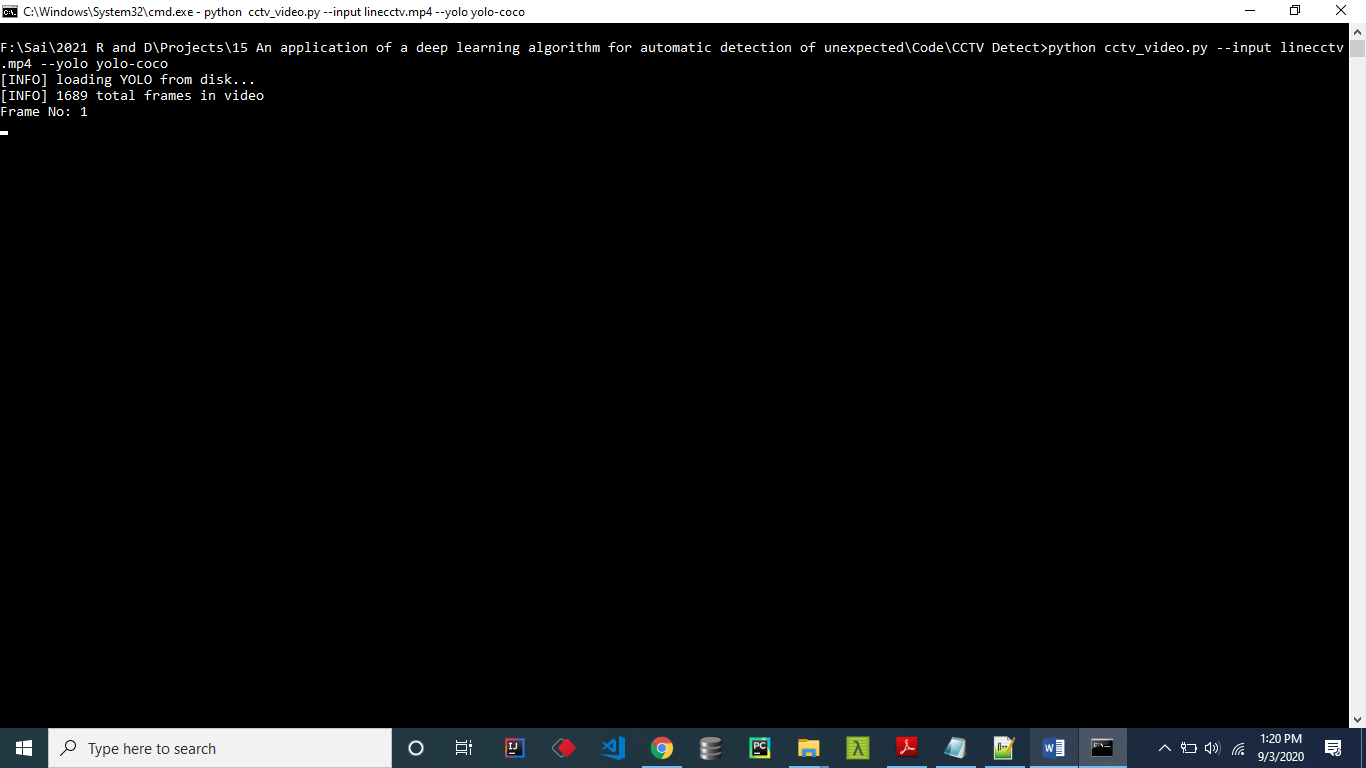
* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

## CHAPTER 7 OUTPUTS

**Starting project:**

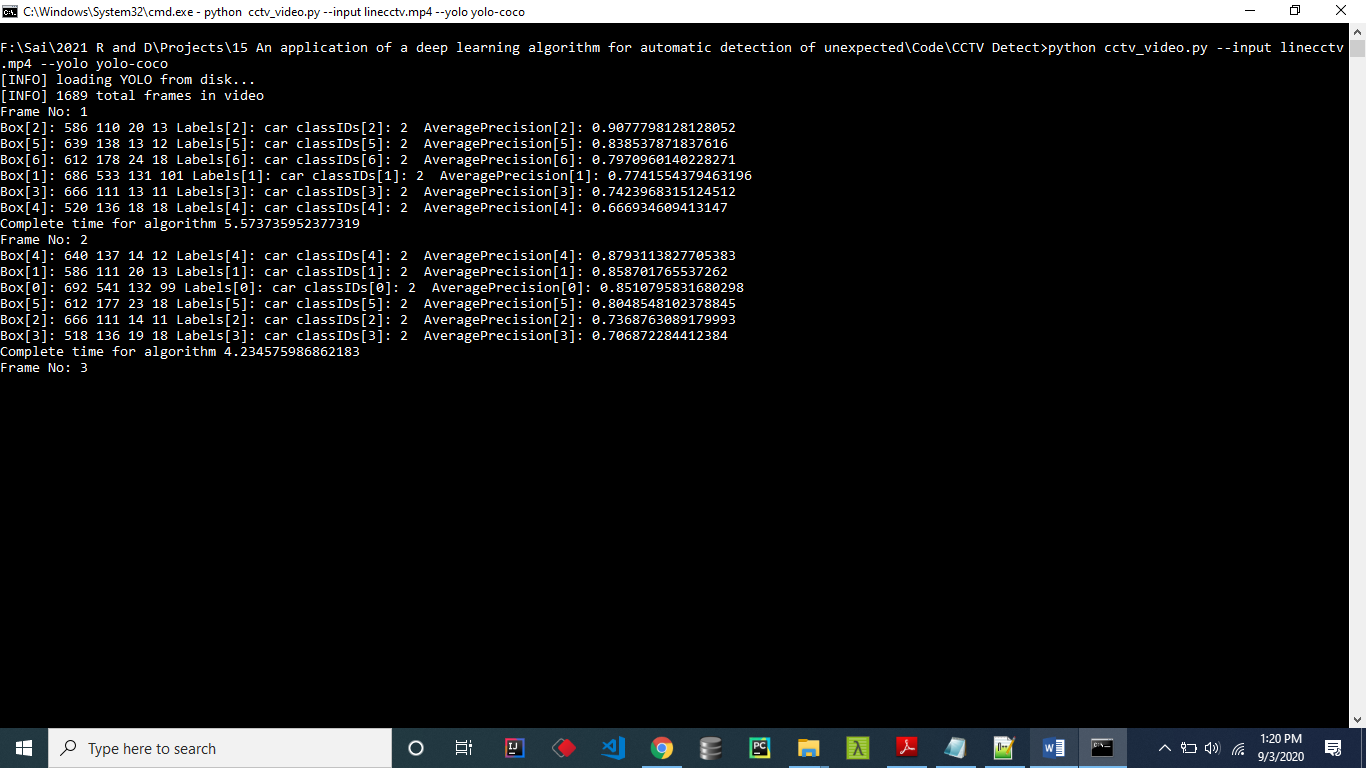


**FIG:7.1 STARTING PROJECT**

**Loading Training Model:** 

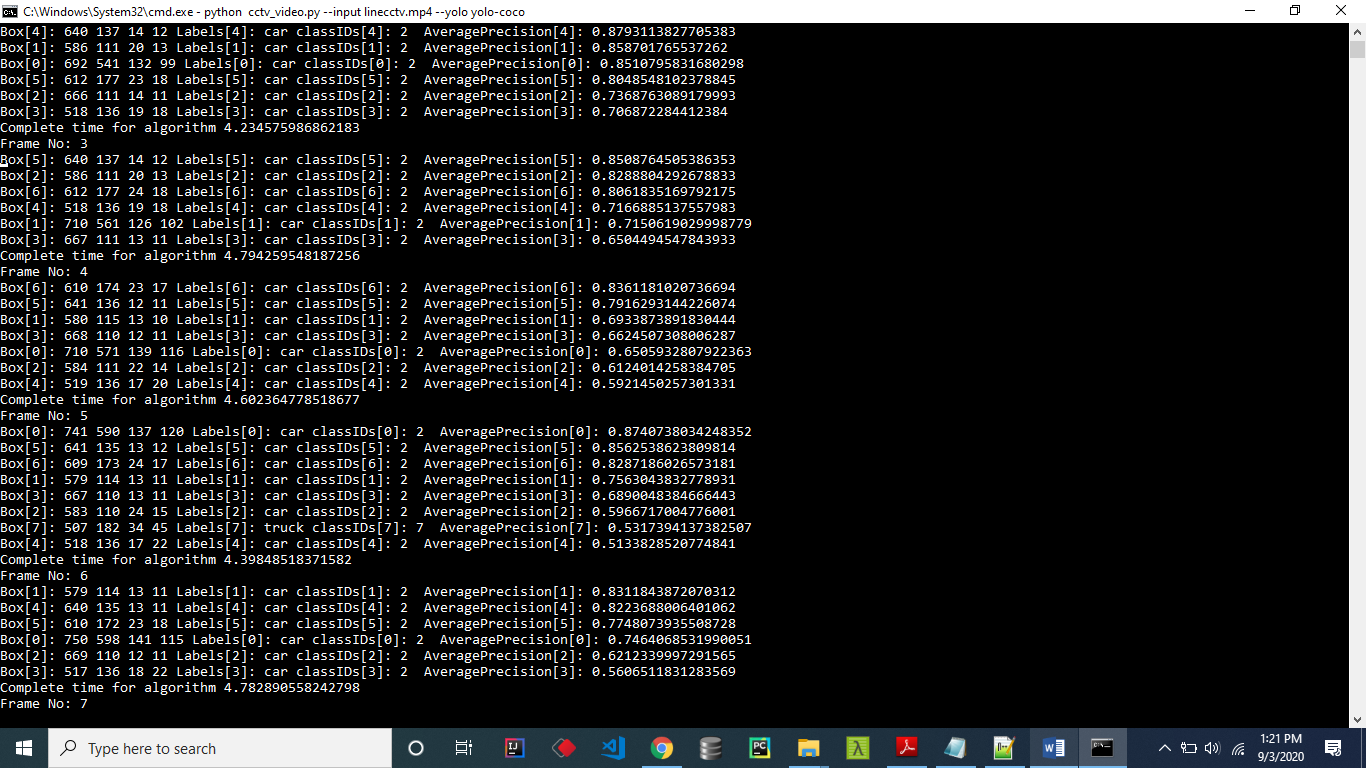
**FIG:7.2 Loading Training Model**

**Video separated as frames:**



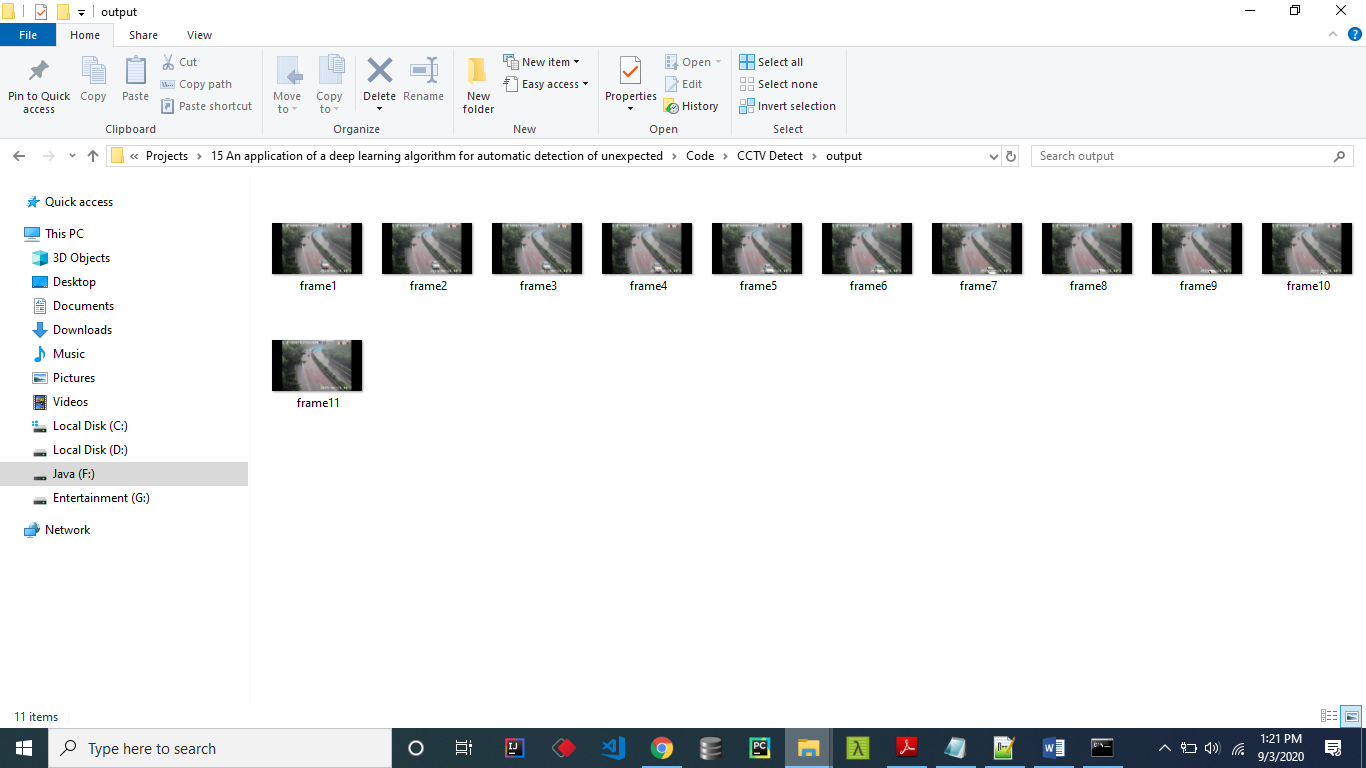
**FIG:7.3 VIDEO SEPARATED AS FRAMES**

**Staring each frames:**



**FIG:7.4 STARING EACH FRAMES**

**Output frames:**



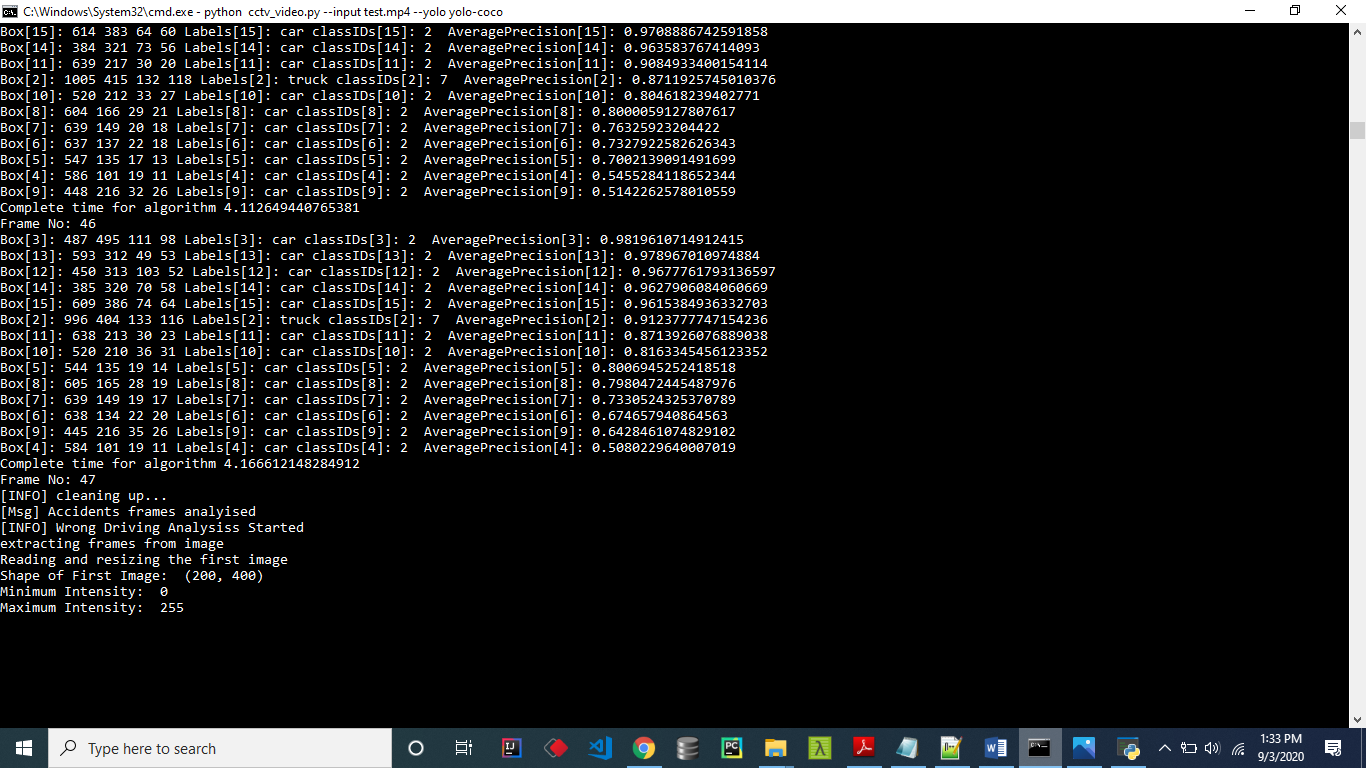
**FIG:7.5 OUTUT FRAMES**

**Object Detetction:**



**FIG:7.6 OBJECT DETECTION**

**Preprocess done:**



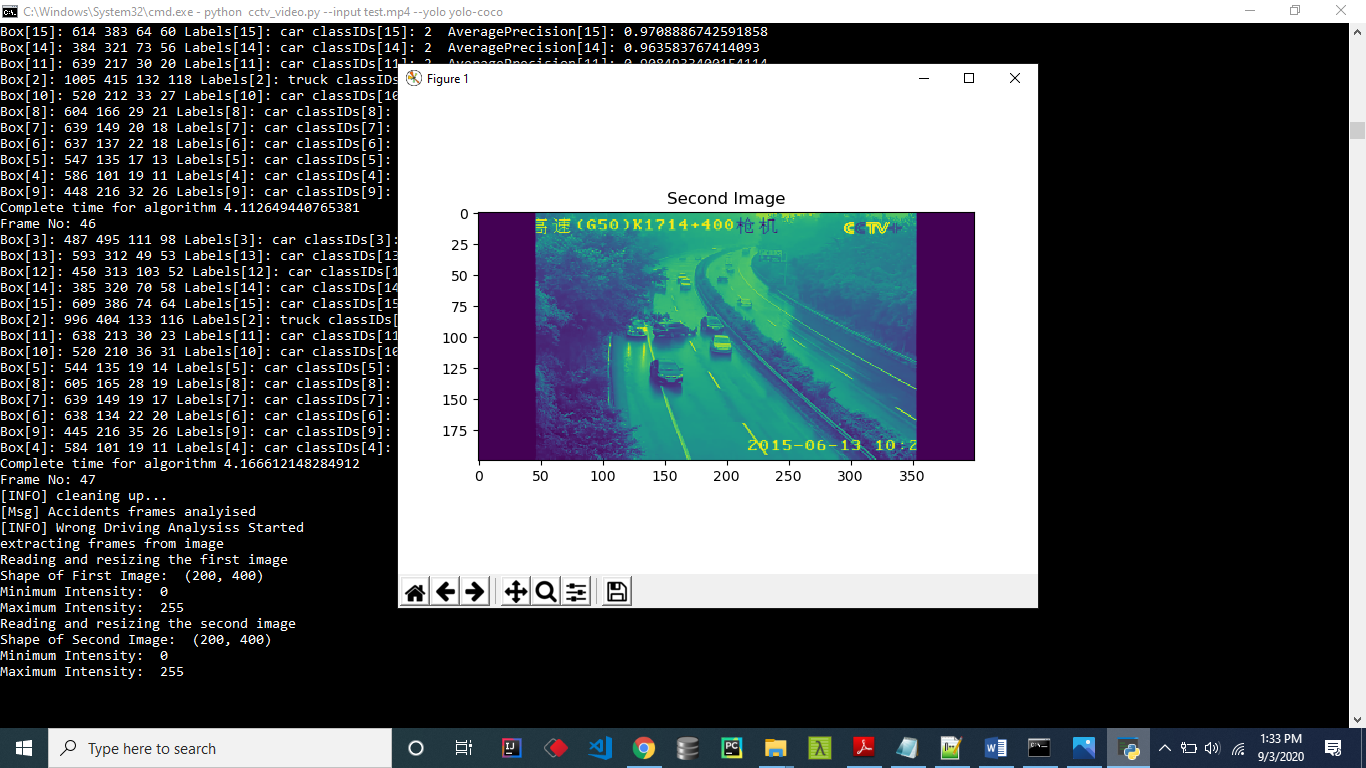
**FIG:7.7 PREPROCESS DONE**

**First Image Compare:**



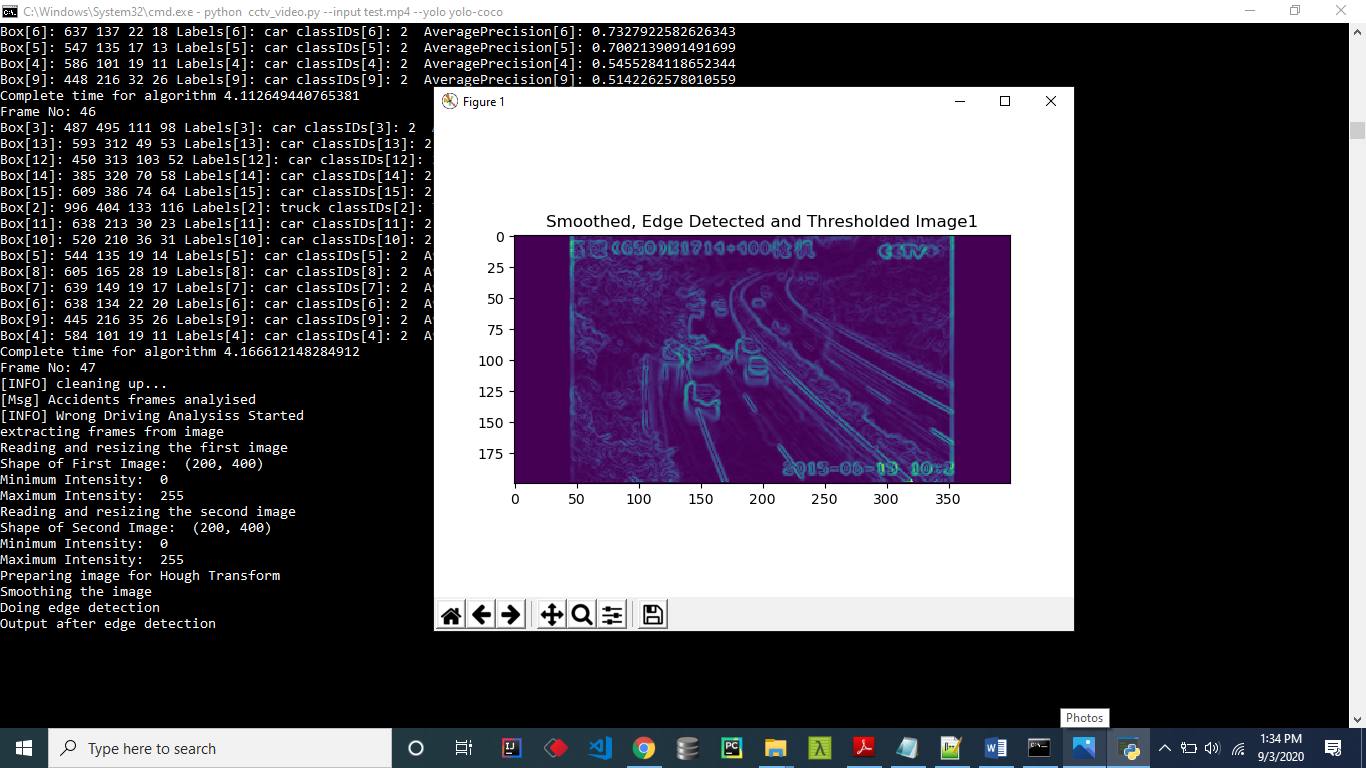
**FIG:7.8 FIRST IMAGE COMPARE**

**Second Image Compare:**



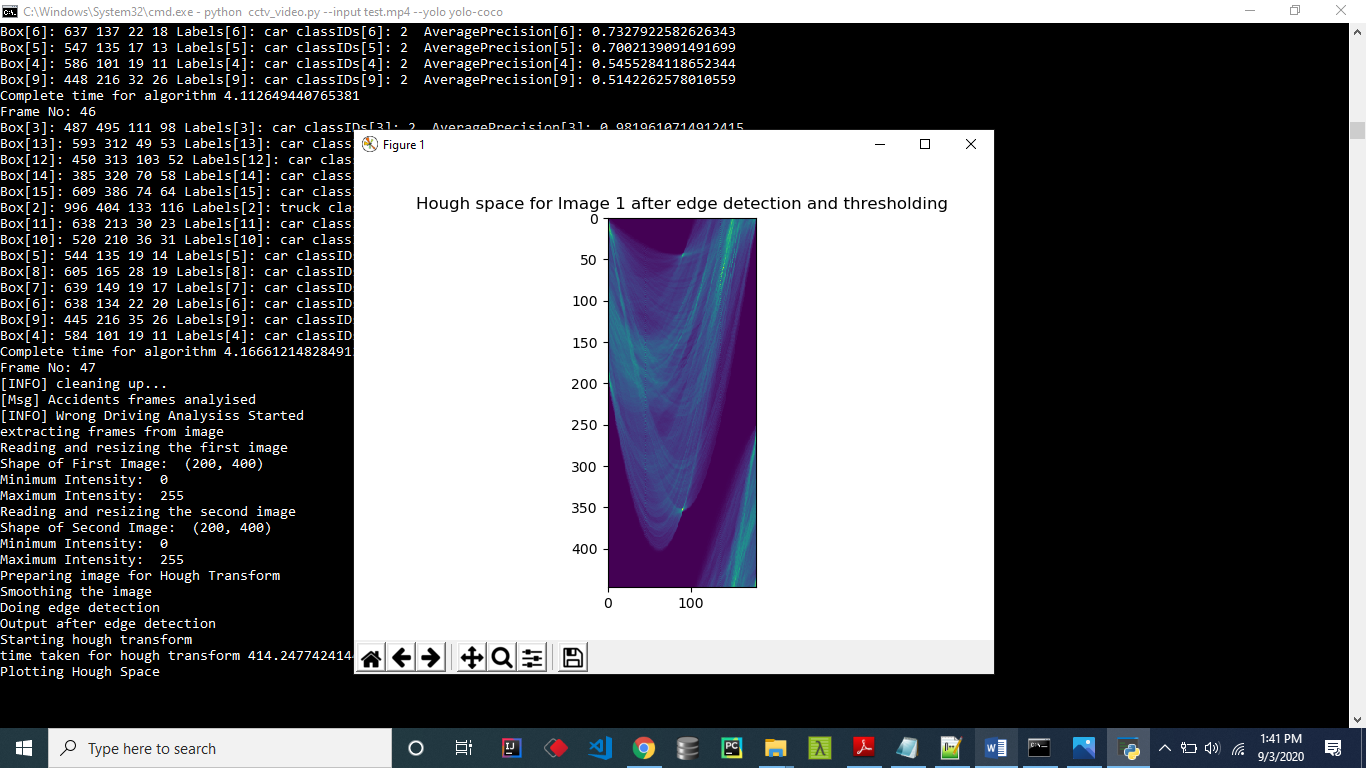
**FIG:7.9 SECOND IMAGE COMPARE**

**Smooth Detection:**



**FIG:7.10 SMOOTH DETECTION**

**Space Count:**



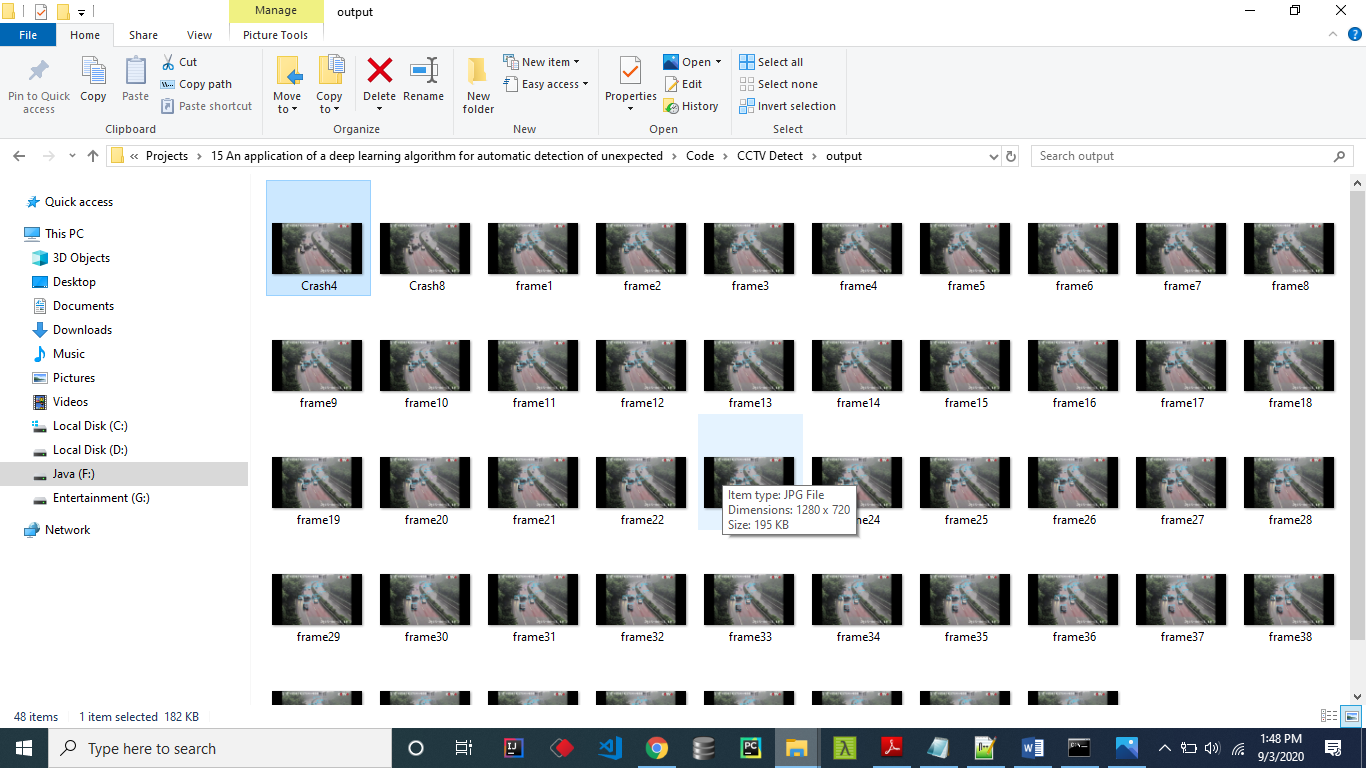
**FIG:7.11 SPACE COUNT**

**Making Video:**



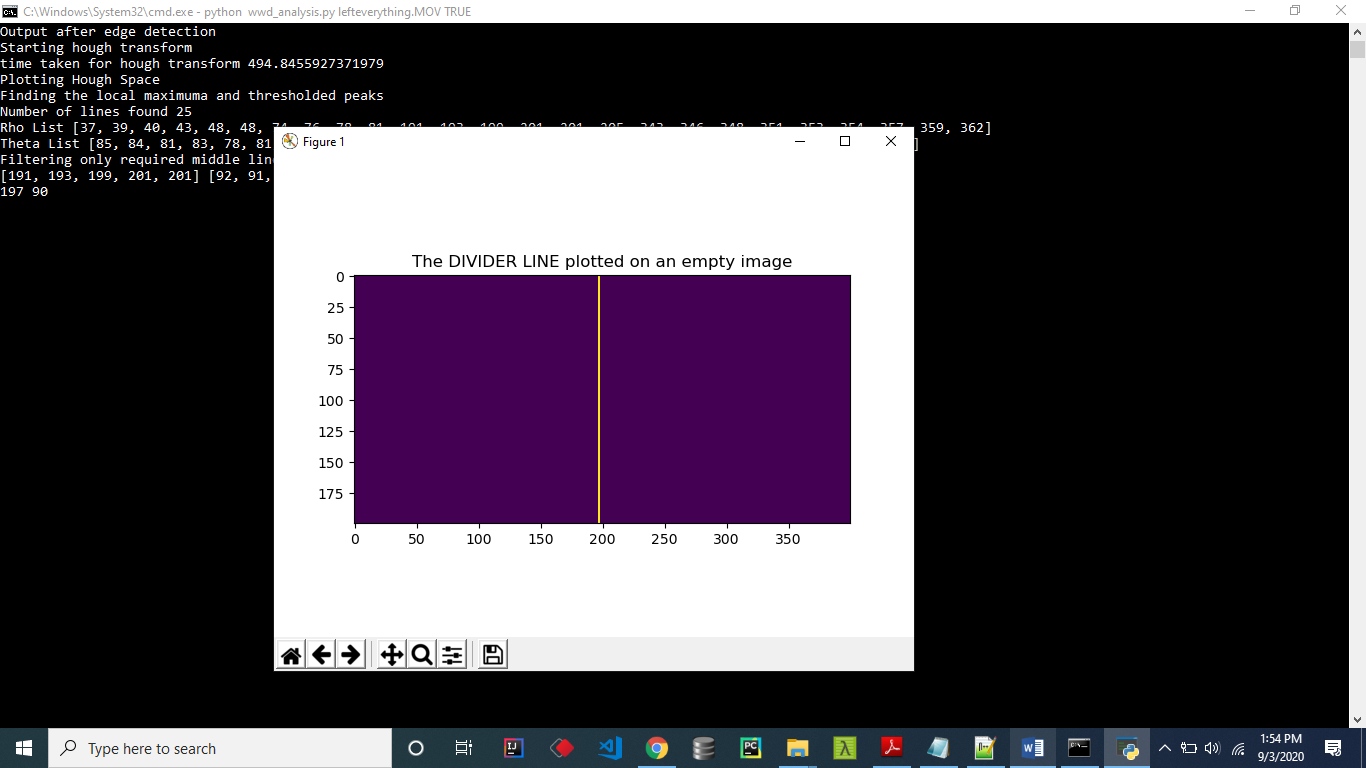
**FIG:7.12 MAKING VIDEO**

**Crash Detected:**



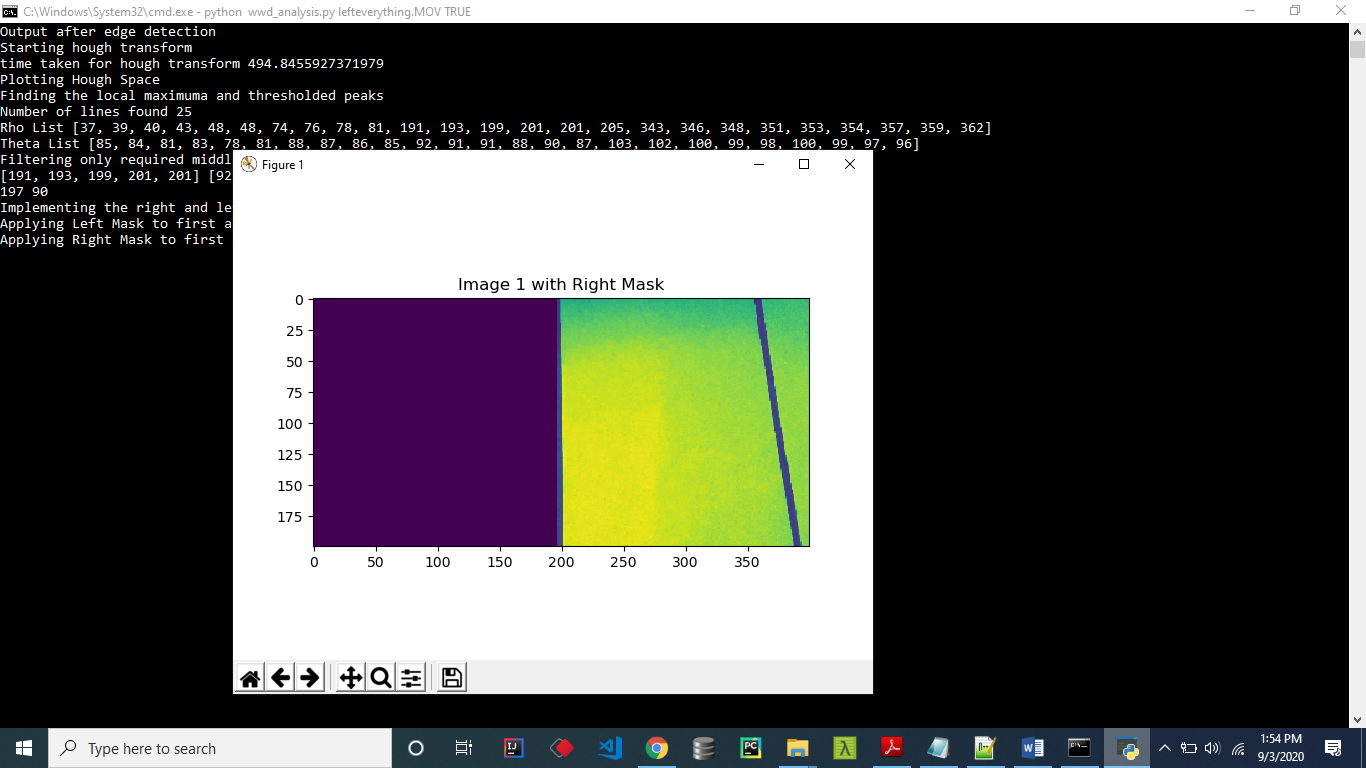
**FIG:7.13 CRASH DETECTED**

**Identifying driving line:**



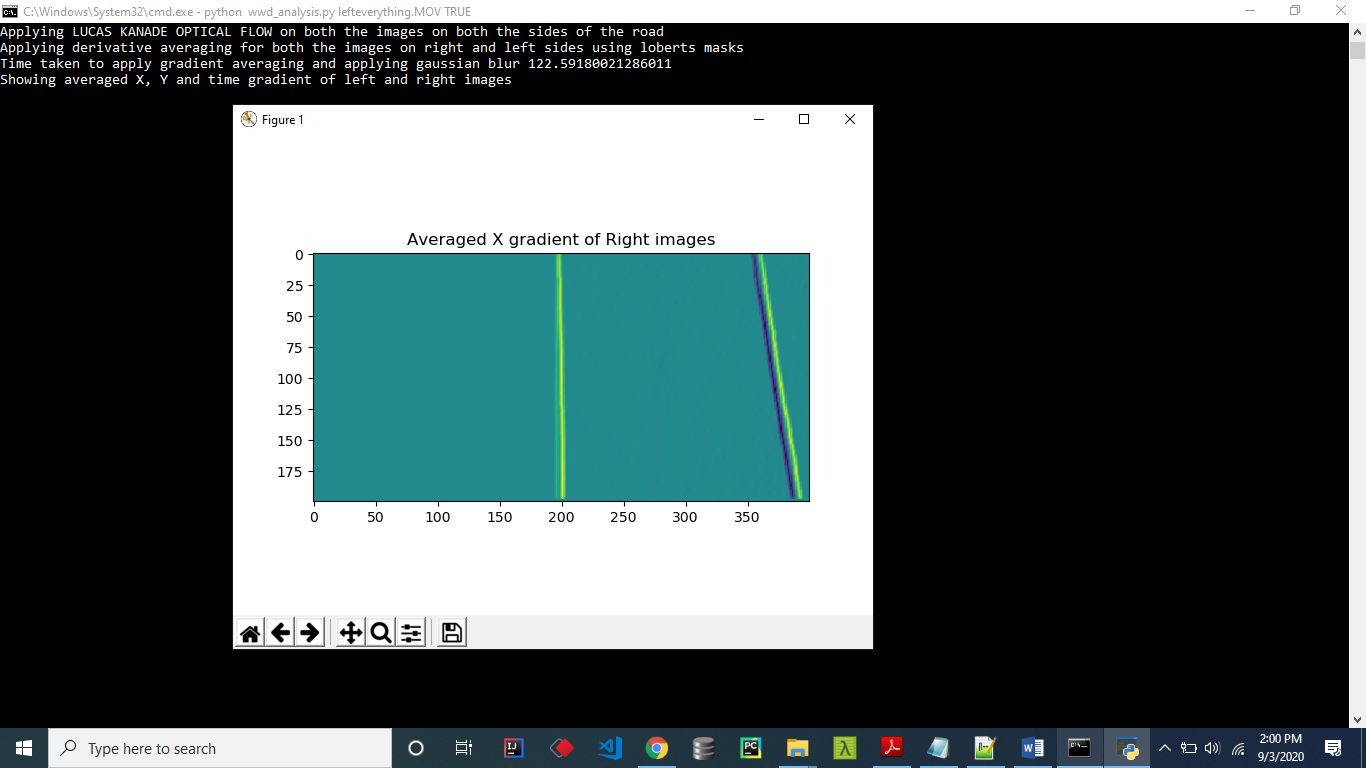
**FIG:7.14 IDENTIFYING DRIVING LINE**

**Image with left mask:**



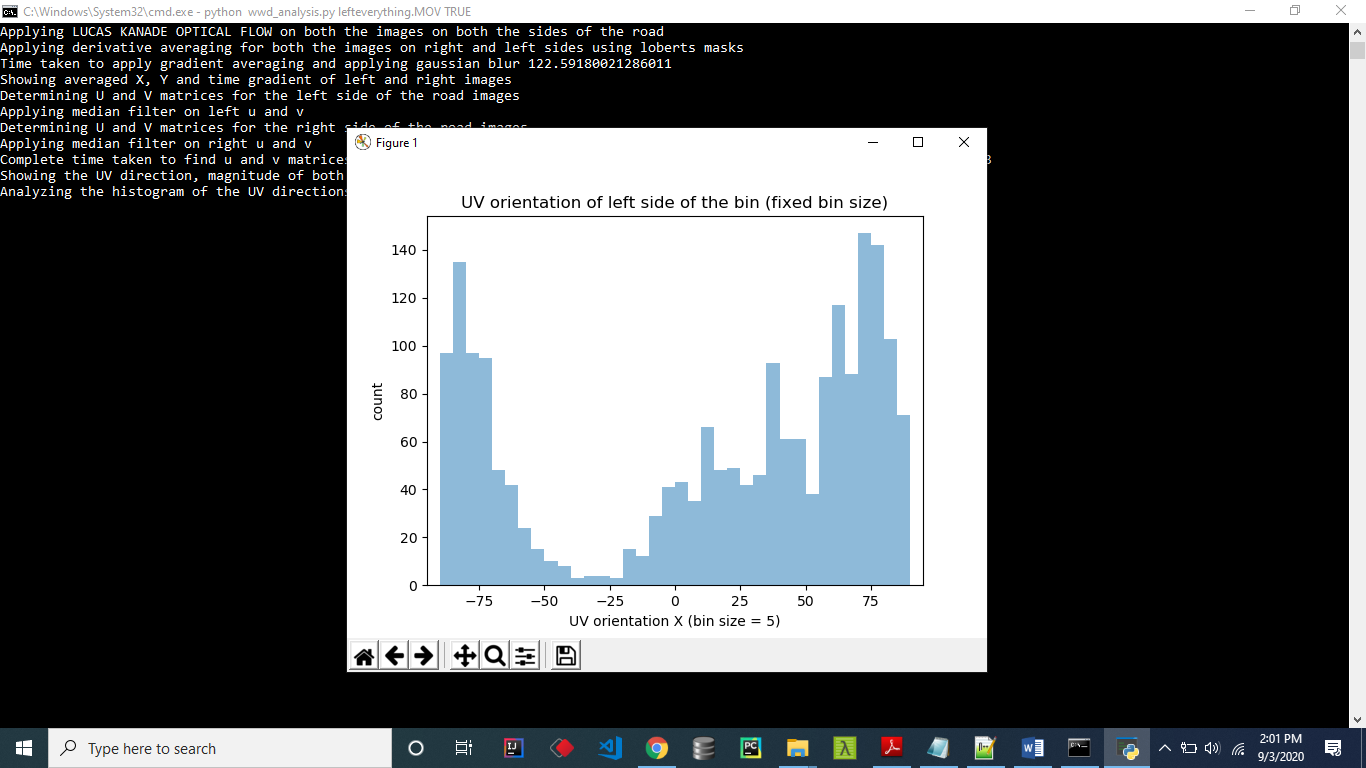
**FIG:7.15 IMAGE WITH LEFT MASK**

**Average Gradient Image:**



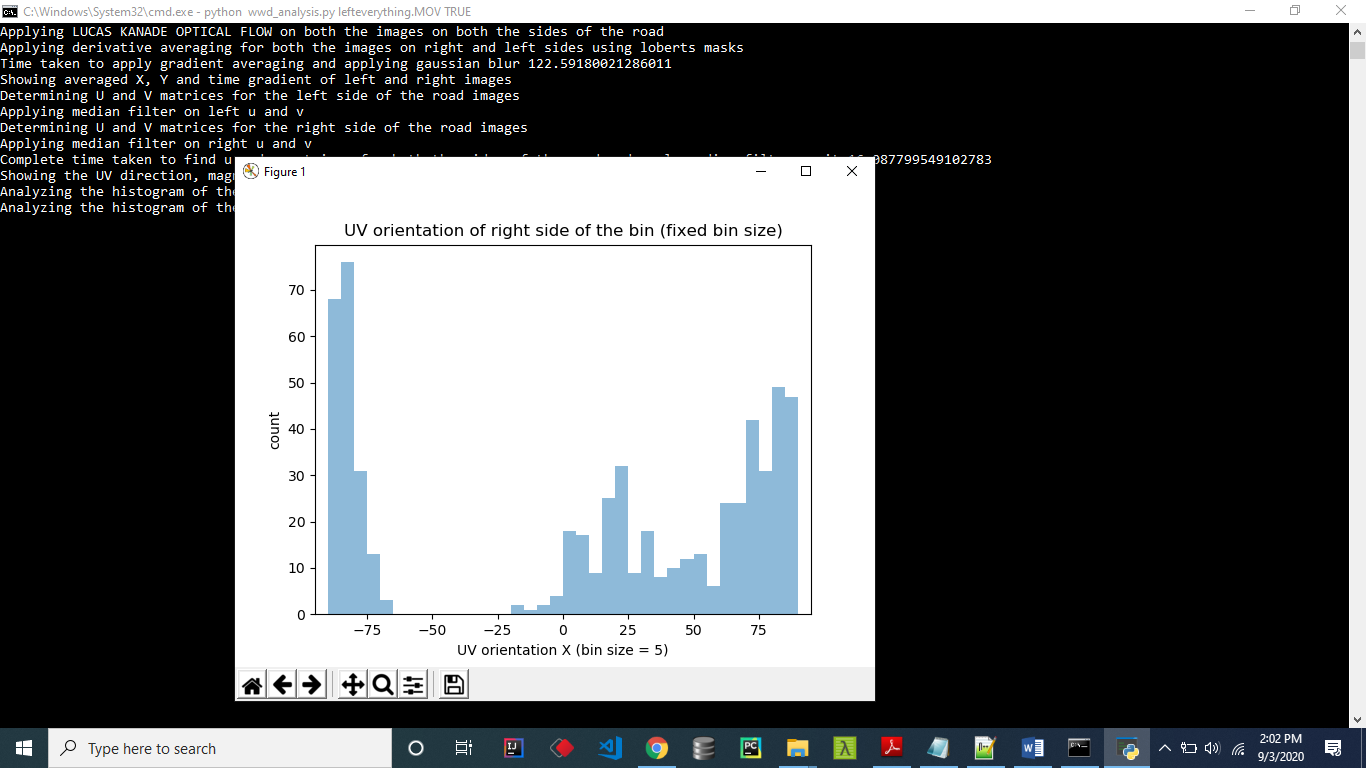
**FIG:7.16 AVERAGE GRADIENT IMAGE**

**UV orientation:**



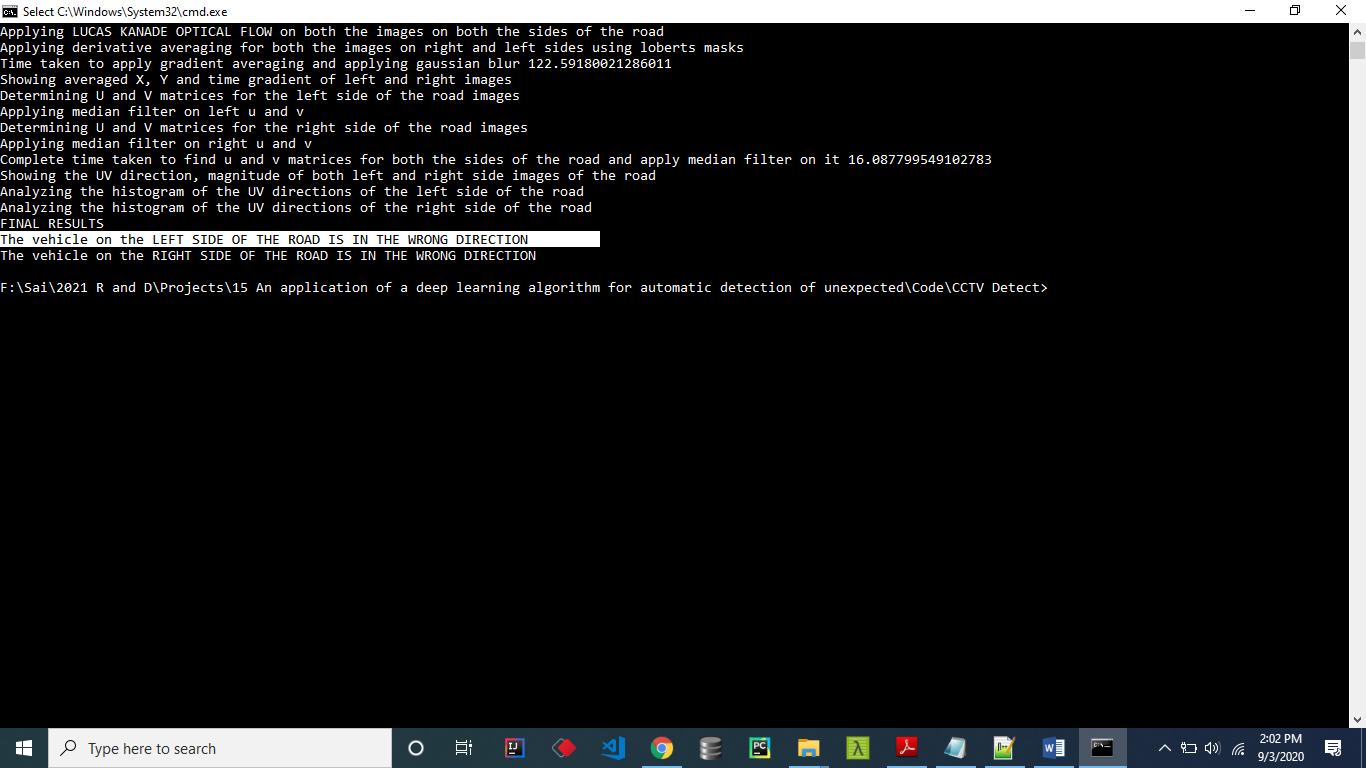
**FIG:7.17-1 UV ORIENTATION**

**UV orientation**



**FIG:7.17-2 UV ORIENTATION**

**Detect Vehicle Direction:**



**FIG:7.18 DETECT VEHICLE DIRECTION**

## CHAPTER 8

**SYSTEM TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail unacceptably. There are various types of tests. Each test type addresses a specific testing requirement.

### TYPES OF TESTS

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application.it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**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 testing**

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 Testing**

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.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# **Integration Testing:**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

## 

## CHAPTER 9

## CONCLUSION

Our project 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.

**Future 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 to 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.

## CHAPTER 10

## 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.

[17] A. Perera, C. Srinivas, A. Hoogs, and G. Br ooksby, “Multi-Object Tracking Through Simultaneous Long Occlusions and Split-Merge Conditions,” in Computer Vision and Pattern Recognition. 2006, IEEE.

[18] W. Choi, “Near-Online Multi-target Tracking with Aggregated Local Flow Descriptor,” in International Conference on Computer Vision, 2015.

[19] Y. Xiang, A. Alahi, and S. Savarese, “Learning to Track : Online Multi-Object Tracking by Decision Making,” in International Conference on Computer Vision, 2015.

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

[21] R. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” Journal of Basic Engineering, vol. 82, no. Series D, pp. 35–45, 1960.

[22] H. W. Kuhn, “The Hungarian method for the assignment problem,” Naval Research Logistics Quarterly, vol. 2, pp. 83–97, 1955.

[23] Y. Bar-Shalom, Tracking and data association, Academic Press Professional, Inc., 1987.

[24] S. H. Bae and K. J. Yoon, “Robust Online Multi-Object Tracking based on Tracklet Confidence and Online Discriminative Appearance Learning,” Computer Vision and Pattern Recognition, 2014.

[25] Y. Min and J. Yunde, “Temporal Dynamic Appearance Modeling for Online Multi-Person Tracking,” oct 2015. [19] A. Bewley, V. Guizilini, F. Ramos, and B. Upcroft, “Online Self-Supervised Multi-Instance Segmentation of Dynamic Objects,” in International Conference on Robotics and Automation. 2014, IEEE.

[26] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, “3D Traffic Scene Understanding from Movable Platforms,” Pattern Analysis and Machine Intelligence, 2014.

[27] M. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” in European Conference on Computer Vision, 2014.

[28] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in International Conference on Learning Representations, 2015.

[29] H. Pirsiavash, D. Ramanan, and C. Fowlkes, “Globallyoptimal greedy algorithms for tracking a variable number of objects,” in Computer Vision and Pattern Recognition. 2011, IEEE.

[30] Y. Li, C. Huang, and R. Nevatia, “Learning to associate: HybridBoosted multi-target tracker for crowded scene,” in Computer Vision and Pattern Recognition. 2009, IEEE.

[31]K.BernardinandR.Stiefelhagen,“EvaluatingMultipleObjectTrackingPerformance: The CLEAR MOTMetrics,” ImageandVideo Processing, , no. May,2008.