Cairo University
Faculty of Engineering
Department of Computer Engineering

Argus

A Graduation Project Report Submitted
to
Faculty of Engineering, Cairo University
in Partial Fulfillment of the requirements of the degree
of
Bachelor of Science in Computer Engineering.

Presented by:
Khaled Sabry Esraa Mamdouh
Mohamed Emad Roba Gamal

Supervised by:
Nevin Darwish

Cairo University, August 2020

Abstract

Over 1.35 million people die in a road traffic crash each year.Unless action is taken, road traffic injuries are predicted to become the fifth leading cause of death by 2030. The main reason for deaths is the long delay in reporting the accidents and the arrival of help. This project presents a system based on computer vision techniques that detects road accidents and reports them in nearly real-time as well as allowing the monitoring of accidents using a client server architecture and an interactive GUI. It is noted that it will save accidents in the database for later inspection and analysis. In short, the project objectives are real time crash detection and notifying authorities as soon as possible. The word Argus is inspired form Greek mythology.Argus is a many-eyed giant in an ancient Greek myth.The name was chosen to emphasize the monitoring action supposed to be done by the system. Our proposed solution adopts both a computer vision and machine learning approach to address the problem, where the output of the system is a message on the client side indicating a crash. The tools used for development were the servers of Google Colab, the frameworks of Tensorflow and Pytorch as well as as SQLite. The system has been successfully implemented using python and tested using pytest framework. This first version has an accuracy of 95% in crash detection and achieves speed around 30-35 fps.

الملخص

يموت أكثر من 1.35 مليون شخص في حادث سير مروري كل عام. ما لم يتم اتخاذ إجراء ، من المتوقع أن تصبح الإصابات الناجمة عن حركة المرور على الطرق خامس سبب رئيسي للوفاة بحلول عام 2030. والسبب الرئيسي لإرتفاع معدل الوفيات هو التأخير الطويل في الإبلاغ عن الحوادث ووصول المساعدة. مشروعنا عبارة عن نظام يعتمد على تقنيات رؤية الكمبيوتر التي تكشف عن حوادث الطرق وتبلغ عنها في الوقت الفعلي تقريبًا وتسمح بمراقبة الحوادث باستخدام بنية خادم العميل وواجهة المستخدم الرسومية التفاعلية. يشار إلى أنه سيوفر الحوادث في قاعدة البيانات لفحصها لاحقًا.

من الشرح السابق ، أهداف المشروع هي الكشف عن التصادم في الوقت الحقيقي وإخطار السلطات في أقرب وقت ممكن. عند البحث عن حل ، اتخذنا نهج رؤية الكمبيوتر والتعلم الآلي لمعالجة المشكلة ، حيث يكون إخراج البرنامج عبارة عن رسالة على جانب العميل تشير إلى تعليق توضيحي لحادث.

الأدوات المستخدمة في التطوير كانت خوادم google colab ، وايضا tensorflow, pytorch ,SQLite

يتم ارسال الصور الملتقطة من كاميرات المراقبة المثبتة على الطرق العامة إلى السيرفر لتحديد ما إذا كان هناك حادثة فإذا كان هناك حادثًا سوف يرسل إشعار لبرنامج مثبت عند السلطات المختصة ليرسلوا الدعم الطبي لموقع الحادثة. يتمتع هذا الإصدار الأول بدقة 95٪ في الكشف عن التصادم ويحقق سرعة حوالي 30-35 إطارًا في الثانية.

Acknowledgment

Many thanks to Dr.Nevin Darwish for her help and recommendations through out the project

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
CAPEX	Capital expenditures
CCTV	Closed-circuit television
CNN	Convolutional Neural Network
DL	Deep Learning
FPN	Feature pyramid network
GPUs	Graphics processing units
IOU	Intersection over union
ML	Machine Learning
MOSSE	Minimum Output Sum of Squared
OPEX	Operating expenses
SVM	Support vector machine
VIF	Violent Flow Descriptor
Yolo	You Only Look Once

List of Symbols

α parameter to control the weight of the smoothness term compared to the optical flow constraint

γ The focusing parameter for focal loss

Contacts

Team members

Name	Email	Phone
Khaled Sabry	khaledsab@gmail.com	+2 01066656297
Mohamed Emad	me24116@gmail.com	+2 01115653305
Esraa Mamdouh	esraa.ali9798@gmail.com	+2 01201510767
Roba Gamal	Eng.robagamal@gmail.com	+2 01112778864

Supervisor

Name	Email	Phone
Nevin Darwish	ndarwish@eng.cu.edu.eg	+201222247364

This page is intentionally left blank

1. Introduction

Our project is a complete system to detect car crashes on roads and report them to the authorities to send medical help. It is obvious how important the problem that it is certainly both a national and global issue of concern. Latest statistics show that more than 1.35 million people die in road accidents every year [1].

1.1 Motivation and Justification

Car crash detection is an international problem of high importance. Making use of Closed-circuit television (CCTV) cameras for detecting crashes immediately and reporting them can help save a lot of souls. Approximately 1.35 million people die in road crashes each year, on average 3,700 people lose their lives every day on the roads. An additional 20-50 million suffer non-fatal injuries, often resulting in long-term disabilities. Egypt loses about 12 000 lives due to road traffic crashes every year. It has a road traffic fatality rate of 42 deaths per 100 000 population. Majority (48%) of those killed are passengers of four-wheelers, though pedestrians also constitute a significant proportion (20%) of these fatalities. Being engineers, we feel it is our duty to try to contribute to relaxing this problem, thence, the main goal is tackling car crashes in a way that would reduce its harsh effects.

1.2 Project Objectives and Problem Definition

One of the main reasons for car crash after effects is the delay in help received to persons involved in the crash. The project problem is then defined as, given a video taken from a surveillance camera placed on public roads, it is required to detect a car crash by proper processing of the camera stream. The stream is fed to a server for car crash detection and upon detection, a notification is sent to a web client which should instantaneously send medical help.

1.3 Project Outcomes

The outcome of the project is a system that can be deployed as a server-client app.A server-client architecture is a computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client.The server side receives the stream of frames from the camera. It performs the computer vision algorithms for detecting and tracking vehicles. The web server is a traditional web server, It receives the results from the processing side and notifies (Updates) the clients. The clients can be traffic controllers, ambulance,..etc and or emergency services, hospitals and any other entity as required.

1.4 Document Organization

This document presents details of the project design and implementation. In addition to this introductory chapter, it consists of 6 chapters and 4 illustrative appendices. Chapter 2, covers the market feasibility study with a detailed business case while Chapter 3 presents the necessary science background related to object detection and tracking. A detailed description for the system architecture is covered in Chapter4. Chapter 5 discusses the implementation issues and successful verification and testing techniques. Discussion of results are also included in Chapter 5. Finally main conclusions and future work directions are presented in Chapter 6.

2. Market Feasibility Study

Although there are many applications for car crash detection using sensor data, there are no applications based on machine vision techniques in production. It is more of a research topic right now.

2.1 Targeted Customers

The main audience targeted by these projects are governments. It can help in saving lots of lives by notifying the medical specialists and asking for immediate help. Also, Deploying such a system can help the authorities further investigate the reasons of accidents through examining recorded videos taken by the system and stored in a well organized database.

2.2 Market Survey

In the context of the Market survey we started searching for similar applications and companies that try to tackle the same problem.

2.2.1 Google automatic car crash detection

With the help of various sensors including accelerometer and even the microphone, Pixel phones will attempt to detect an accident. If one occurs, the phone will loudly sound an alarm, and if there's no response, it will automatically call and provide your location to emergency services.

Advantages of such an application is that it is simple in principle and expected to have excellent accuracy. Because during a severe car crash every single thing inside the car is subject to high changes of speed suffering massive accelerations that could reach up to 200 times the gravity. Nowadays Smartphones count with a sensor that can measure accelerations, the accelerometer. Using this sensor output in a detection algorithm the application detects if the user has been in a car accident. While the disadvantages of this application is that it is Limited to pixel devices and treats governments as third parties which mean more latency until authorities are informed in a time where minutes count. Beside it requires mobile network availability to be effective. So, in remote places it will be cheaper to install a camera and use wired communication rather than having to provide network coverage to the whole area.

2.2.2 SOSmart

SOSmart detects car accidents using the accelerometer sensor and GPS of the smartphone, and sends an emergency notification with the location to a preselected emergency contact. This allows the contacts to send help as soon as possible.

It has the Same advantages and disadvantages like google automatic car crash detection. Except that it is available for all phones not exclusive to any company.

2.3 Business Case and Financial Analysis

2.3.1 Business Case

The target customers are higher economy countries. Majority of countries are too poor to maintain such a system. So, target customers are more like East Asia rich countries like Japan and Singapore or most European countries, North America countries and some oil rich Arab countries. The number of countries that could be interested in such a system is estimated to be about 75 countries. We expect that most of these countries would be interested in purchasing the application. So, as a matter of fact the number of products that will be sold will be limited by how many countries the company is ready to serve at the same time. Some countries like England already have fascinating infrastructure therefore the system can be set up much faster than countries like Egypt which will require planning, time to place CCTVs in places of interest and setting up servers. So, countries with CCTVs already installed and monitoring stations will require about three months to install. While countries like Egypt will need about a year to get the application installed and distributed among servers. In conclusion, It is expected that the company will serve about ten countries in the five years some of them have more developed infrastructure than others.

2.3.2 Financial Analysis

Capital expenditures (CAPEX)

System parts	Estimated price	Purpose
CCTV with built-in router	500 EGP	Capture the footage and send it to the satellite.
Subscription to Tiba-1 Satellite [1- month]	1000EGP	To receive footage from CCTV cameras and send them to servers. It can be used for more than one camera.
GPU - NVIDIA Tesla K80	15, 000 EGP	To receive footage from CCTV cameras and send them to servers. It can be used for more than one camera
GPU - NVIDIA Tesla K80	15, 000 EGP	For high-performance processing. It can be used for more than one camera.
HP 6305 pro-Desktop	1500 EGP	It can be used for more than one camera.
Monitor 17 inch Dell	1250 EGP	Display for the system. It can be used for more than one camera
Dell Multi-Device Wireless Keyboard and Mouse	500 EGP	Take user actions

Table 2.1: Estimated cost

.

Operating expenses (OPEX)

The best part of our project is that it requires very little man power to operate. Two employees will be sufficient to manage a complete city footage. Employees do not need to be highly educated so they will be paid minimum wage; About 4000 EGP.

Periodic check and maintenance will be required every six months to fix CCTVs and servers. It will cost about 10000 EGP.

3. Literature Survey

Car crash detection is a real time problem, which not only needs accuracy but has to be fast as well. This trade-off between speed and accuracy is the main initiator of many ideas presented in lots of papers recently published.Most of the papers make use of the latest revolution in the field of Artificial Intelligence (AI) and computer vision, which is basically a natural result of the huge development in the field of Graphics processing units (GPUs) architecture. In this Chapter we will discuss three topics: object detection, object tracking and crash detection -using techniques like collision estimation and Violent Flow Descriptor (VIF). We start by introducing the concepts and then go on to review some of the most famous architectures and the algorithms of interest to this project. Finally, we conclude with our proposal for our implemented approach Illustrating and how it made use of these 3 topics.

3.1 Object detection

Object Detection is a common Computer Vision problem which deals with identifying and locating objects of certain classes in the image. Interpreting the object location can be done in various ways, including creating a bounding box around the object or marking every pixel in the image which contains the object (called segmentation). Object detection before Deep Learning (DL) was a several step process, starting with edge detection and feature extraction using techniques like SIFT, HOG etc. These images were then compared with existing object templates, usually at multi scale levels, to detect and localize objects present in the image. The evolution of Machine Learning (ML) into deep learning and introducing Convolutional Neural Network (CNN)s into the field of computer vision revolutionized how the problem is targeted and introduced even harder problems for segmentation. The Deep learning techniques used so far can be summarized into 2 types of detectors to follow:

- Two-Step Object Detection
- One-Step Object Detection

Two-Step Object Detection involves algorithms that first identify bounding boxes which may potentially contain objects and then classify each bounding separately. The first step requires a Region Proposal Network, providing a number of regions which are then passed to common DL based classification architectures. On the other hand, with the need for real time object detection, many One-Step Object Detection architectures have been proposed, like YOLO, YOLOv2, YOLOv3, SSD, RetinaNet and many more which try to combine the detection and classification step. One of the major accomplishments of these one-step algorithms have been introducing the idea of 'regressing' the bounding box predictions. When everybounding box is represented easily with a few values (for example, xmin, xmax, ymin and ymax), it becomes easier to combine the detection and classification step and dramatically speed up the pipeline.

Object detection is essential in our problem since it will be used in detecting and locating the vehicles and pedestrians in a given frame.[2]

3.2 Object tracking

Object tracking is the task of taking an initial set of object detections , creating a unique ID for each of the initial detections, tracking each of the objects as they move around frames in a video and maintaining the ID assignment.In our problem, after vehicle detection, the tracker should track for (n) numbers of frames to evaluate the changes happening to the objects over these frames and decide whether there is an accident or not.

In the following sections there will be given some examples for the most famous tracking algorithms,As well as one of the most famous flow estimation algorithms which is the optical flow algorithm .

3.3 Collision Estimation and Crash Detection

The Violent Flow Descriptor (VIF) -which will be explained in a later section- is mainly used for crash detection.However,it shows somehow weak performance in practice .A new approach using gaming techniques was introduced . The new approach took in consideration the variety of camera angles, resolutions and lighting . The idea is to limit the trackers entering the Violent Flow Descriptor (VIF) descriptor according to the following algorithm. First, we need to estimate vehicles speed in the video. Having a tracker on every vehicle,we can easily estimate the average speed of the vehicle by pixel unit,However the camera angle differs from cctv camera to another as well the camera position ,height and resolution.Thus another unit instead of pixel unit is needed to estimate the average speed of a tracker. To solve the camera resolution problem, we need to resize every input feed to a fixed width and height (480,360), The height/scale problem, can be solved by multiplying the average speed of a vehicle to a speed coefficient parameter. This parameter has an inverse relation to the area which means that when the vehicle has a bigger area it has lower speed. The angle of the camera plays an important role in the crash estimation process. The best camera angle is the one with the top view, since the focus is on the x,y plane where the vehicles move.

In the real world it is impossible to capture such a top view image as in figure 3.1 unless you use a drone as a cctv camera. this makes the perfect possible perfect cctv angle is the one in 3.2, this will give us the depth we want.

It was found by experiment that the best way to calculate the average speed is by taking the average of every 10 frames along with its corresponding angle. you can expect the next frame after another 10 frames in the future.

Another thing to put in mind, this will make a gap which is the first 15 frames of the tracker, and to solve this gap we need to make another detection for the tracker at the

Figure 3.1: Example of top view image

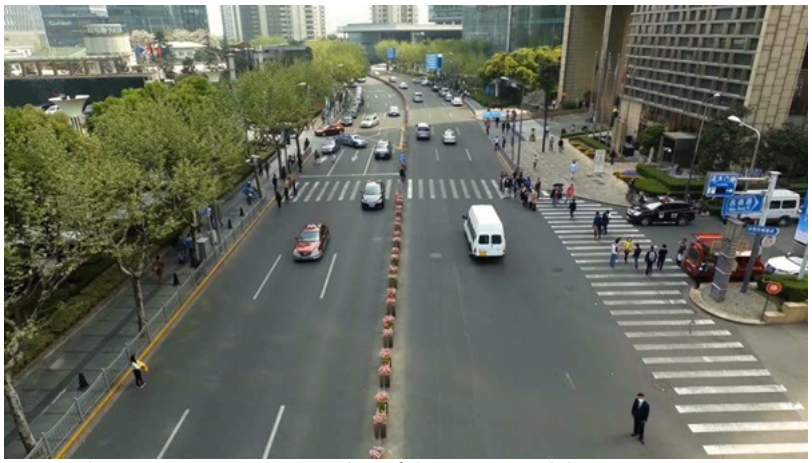

Figure 3.2: Example of best possible view image

15^{th} frame and the next 45^{th} frame and so on. Second, now we can determine if the vehicle speed is moving above the speed limit or not, so we put a threshold of 50 if above then the vehicle moves fast. Now, the speed and the expectation about the future position of the vehicle are known. Third, compare every two vehicles with each other in different frames, Subsample the estimated future centers to 3 so you will use 15, 18, 21 frames, and so on. If the two vehicle's average speed is below the limit speed then ignore them if one of them is above the limit then you need to check them. Fourth, get the estimated centers in future frames for both vehicles then get the distance between them and we call this "distance between two estimated centers vehicles", If the two estimated centers are above some threshold then they will not crash because they are away from each other but if it is below then, go to the next step. Fifth, get the distance between the estimated center and the real center for each vehicle and get the max distance, that is because if the distance is so large then the vehicle center didn't reach this position as expected and it may tell us an accident has occurred, so we make this equation (the max distance between the actual and estimated center for each vehicle / the distance between the two estimated center in the future) if it's above 0.5 then it may be a crash go to next step. Sixth, here we filter out too small tracker size so the VIF descriptor could recognize it correctly.

3.4 Comparative Study of Previous Work

We will start this section by reviewing The previous work related to object detection architectures and will follow it by the tracking techniques

3.4.1 Object Detection Architectures

This section reviews the mostly used architecture introducing the state of the art one. We maily review four different architectures; AlexNet, Yolo and RetinaNet

Alex Net

AlexNet [3] is the name of a convolutional neural network, designed by Alex Krizhevsky, and published with Ilya Sutskever and Krizhevsky's PhD advisor Geoffrey Hinton. AlexNet [3] competed in the ImageNet Large Scale Visual Recognition Challenge on September 30, 2012. The network achieved a top-5 error of 15.3%, more than 10.8 percentage points lower than that of the runner up. The original paper's primary result was that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of graphics processing units (GPUs) during training.

AlexNet architecture

It consists of eight layers: five convolutional layers and three fully-connected layers. It used new features like:

- ReLU Nonlinearity :AlexNet uses Rectified Linear Units (ReLU) instead of the tanh function, which was standard at the time. ReLU's advantage is in training time; a CNN using ReLU was able to reach a 25% error on the CIFAR-10 dataset six times faster than a CNN using tanh.

- Multiple GPUs: Back in the day, GPUs were still rolling around with 3 gigabytes of memory (nowadays those kinds of memory would be for beginner applications). This was especially bad because the training set had 1.2 million images. AlexNet allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU. Not only does this mean that a bigger model can be trained, but it also cuts down on the training time.

- Overlapping Pooling: CNN s traditionally "pool" outputs of neighboring groups of neurons with no overlapping. However, when the authors introduced overlap, they saw a reduction in error by about 0.5% and found that models with overlapping pooling generally find it harder to over fit.

  Despite the fact that Overlapping was introduced to reduce overfitting the large number of AlexNet parameters -it had 60 million parameters-, was still a major issue in terms of over fitting. Two methods were employed to reduce overfitting

- Data Augmentation. The authors used label-preserving transformation to make their data more varied. Specifically, they generated image translations and horizontal reflections, which increased the training set by a factor of 2048. They also performed Principle Component Analysis (PCA) on the RGB pixel values to change the intensities of RGB channels, which reduced the top-1 error rate by more than 1%

- Dropout. This technique consists of "turning off" neurons with a predetermined probability (e.g. 50%). This means that every iteration uses a different sample of the model's parameters, which forces each neuron to have more robust features that can be used with other random neurons. However, dropout also increases the training time needed for the model's convergence.

Figure 3.3 illustrates Alex-Net architecture

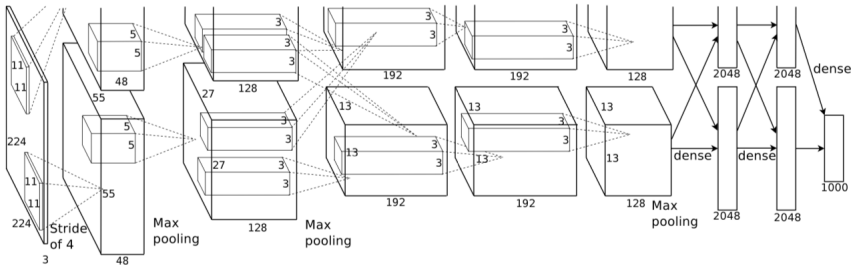

Figure 3.3: Alex-Net architecture

Results

In the 2010 version of the ImageNet competition, the best model achieved 47.1% top-1 error and 28.2% top-5 error. AlexNet vastly outpaced this with a 37.5% top-1 error and a 17.0% top-5 error. AlexNet is able to recognize off-center objects and most of its top five classes for each image are acceptable class predictions. AlexNet won the 2012 ImageNet competition with a top-5 error rate of 15.3%, compared to the second place top-5 error rate of 26.2%.

3.4.2 YOLO

You Only Look Once (Yolo) [4], is another network for object detection. The object detection task consists of determining the location on the image where certain objects are present, as well as classifying those objects. Previous Methods were slow and hard to optimize, thus not suitable for real time applications. Before going into Yolo architecture, we have to understand its encoding.

The first step to understanding YOLO is how it encodes its output. The input image is divided into an S x S grid of cells. For each object that is present in the image, one grid cell is said to be "responsible" for predicting it. That is the cell where the center of the object falls into. These are known as the predictions vector. Each grid cell predicts B bounding boxes as well as C class probabilities. The bounding box prediction has 5 components: (x, y, w, h, confidence). The (x, y) coordinates represent the center of the box, relative to the grid cell location (remember that, if the center of the box does not fall inside the grid cell, then this cell is not responsible for it). These coordinates are normalized to fall between 0 and 1. The (w, h) box dimensions are also normalized to [0, 1], relative to the image size.

As shown in figure 3.4 an example of calculating box co-ordinates according to yolo is provided.

There is still one more component in the bounding box prediction, which is the confidence score. according to Yolo paper : "Formally we define confidence as Pr(Object) * Intersection over union (IOU)(pred, truth) . If no object exists in that cell, the confidence score should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth."[4]

It is also necessary to predict the class probabilities, Pr(Class(i) | Object). This probability is conditioned on the grid cell containing one object. In practice, it means that if no object is present on the grid cell, the loss function will not penalize it for a wrong class prediction. The network only predicts one set of class probabilities per cell, regardless of the number of boxes B. That makes S x S x C class probabilities in total. Adding the class predictions to the output vector, we get a S x S x (B * 5 +C) tensor as output.

Table 3.1: Yolo's architecture

Name	Filters	Output Dimension
Conv 1	7 x 7 x 64, stride=2	224 x 224 x 64
Max Pool 1	2 x 2, stride=2	112 x 112 x 64
Conv 2	3 x 3 x 192	112 x 112 x 192
Max Pool 2	2 x 2, stride=2	56 x 56 x 192
Conv 3	1 x 1 x 128	56 x 56 x 128
Conv 4	3 x 3 x 256	56 x 56 x 256
Conv 5	1 x 1 x 256	56 x 56 x 256
Conv 6	1 x 1 x 512	56 x 56 x 512
Max Pool 3	2 x 2, stride=2	28 x 28 x 512
Conv 7	1 x 1 x 256	28 x 28 x 256
Conv 8	3 x 3 x 512	28 x 28 x 512
Conv 9	1 x 1 x 256	28 x 28 x 256
Conv 10	3 x 3 x 512	28 x 28 x 512
Conv 11	1 x 1 x 256	28 x 28 x 256
Conv 12	3 x 3 x 512	28 x 28 x 512
Conv 13	1 x 1 x 256	28 x 28 x 256
Conv 14	3 x 3 x 512	28 x 28 x 512
Conv 15	1 x 1 x 512	28 x 28 x 512
Conv 16	3 x 3 x 1024	28 x 28 x 1024
Max Pool 4	2 x 2, stride=2	14 x 14 x 1024
Conv 17	1 x 1 x 512	14 x 14 x 512
Conv 18	3 x 3 x 1024	14 x 14 x 1024
Conv 19	1 x 1 x 512	14 x 14 x 512
Conv 20	3 x 3 x 1024	14 x 14 x 1024
Conv 21	3 x 3 x 1024	14 x 14 x 1024
Conv 22	3 x 3 x 1024, stride=2	7 x 7 x 1024
Conv 23	3 x 3 x 1024	7 x 7 x 1024
Conv 24	3 x 3 x 1024	7 x 7 x 1024
FC 1	-	4096
FC 2	-	7 x 7 x 30 (1470)

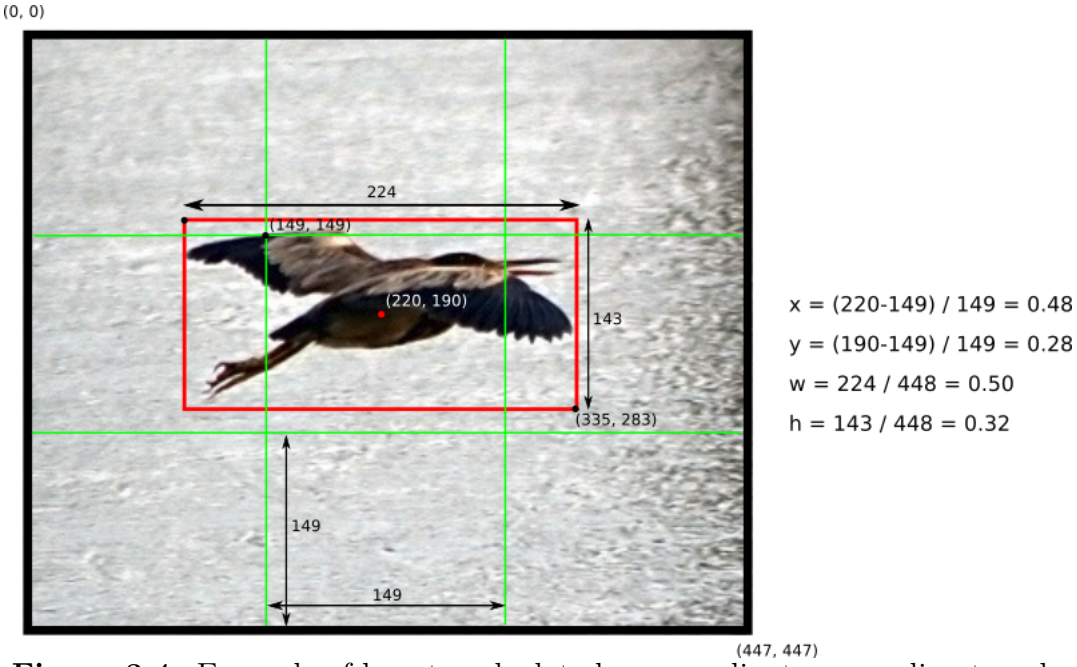

Figure 3.4: Example of how to calculate box co-ordinates according to yolo

Yolo architecture

Results

You only look once (YOLO) is a state-of-the-art, real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a mAP of 57.9% on COCO test-dev. Nevertheless, due to the flexible nature of YOLO architecture in addition to the emergence of more complex real time problems, it has encountered many variations and enhancements.

In the next section we present some of those enhancements and how they differ from the basic architecture.

YOLOv2

YOLOv2 [5] has numerous improvements over YOLO. At 67 FPS, YOLOv2 gets 76.8% mAP on PASCAL VOC 2007. At 40 FPS, YOLOv2 gets 78.6% mAP which is better than Faster R-CNN using ResNet and SSD. With such good results, YOLOv2 was published in 2017 CVPR and got over 1000 citations. The following specifies the cause of improvements over YOLO. Firstly, it used Batch normalization on all convolutional layers giving it 2% improvement in mAP. Secondly, it has a high resolution classifier. YOLOv2 uses 448×448 images for fine-tuning the classification network for 10 epochs on ImageNet yielding another 4% increase in mAP. Thirdly, it removes all fully connected layers and uses anchor boxes to predict bounding boxes. One pooling layer is removed to increase the resolution of output. And 416×416 images are used for training the detection network now.A 13×13

feature map output is obtained, i.e. 32× down sampled. It is noted that without anchor boxes, the intermediate model got 69.5% mAP and recall of 81%. While with anchor boxes, 69.2% mAP and recall of 88% are obtained. Though mAP has dropped a little, recall has increased by a large margin.

In addition to the previous four features, there are four more distinctive features for YOLOv2. First, it uses k-means clustering which leads to good IOU scores. IOU based clustering with 5 anchor boxes (61.0%) has similar results with the one in Faster R-CNN with 9 anchor boxes (60.9%) and IOU based clustering with 9 anchor boxes got 67.2%. Second, YOLO does not have constraints on location prediction which makes the model unstable at early iterations. The predicted bounding box can be far from the original grid location. Thence, YOLOv2 bounds the location using logistic activation,, which makes the value fall between 0 to 1. Third, The 13×13 feature map output is sufficient for detecting large objects. In order to detect small objects well, the 26×26×512 feature maps from earlier layers are mapped into 13×13×2048 feature map, then concatenated with the original 13×13 feature maps for detection. This achieves a 1% increase in mAP. Fourth, for every 10 batches, new image dimensions are randomly chosen among values such as 320, 352, . . . , 608. The network is then resized and training is continued.

It turns out that there was more room for improvement and YOLOv3 was introduced.

YOLOv3

YOLOV3 [6] is the latest variant of a popular object detection algorithm YOLO. The published model recognizes 80 different objects in images and videos, but most importantly it is super fast and nearly as accurate as Single Shot Multi-Box (SSD). It uses the same Bounding Box Prediction as YOLOv2 however it does not use soft-max. It uses three different scales for prediction. It adds to YOLOv2 several convolutional layers to the base feature extractor Darknet-53. The last of these layers predicts the bounding box and class predictions. The feature map is taken from 2 layers previous and is up-sampled by 2×. A feature map is also taken from earlier in the network and is merged with the up-sampled features using concatenation. This is actually the typical encoder-decoder architecture. This method allows it to get more meaningful semantic information from the up-sampled features and finer-grained information from the earlier feature map. Finally, a few more convolutional layers are added to process this combined feature map, eventually predicts a similar tensor, although now twice the size.

The second major variation from YOLOv2 is the use of the Darknet-19 classification network. While it is used in YOLOv2 for feature extraction, YOLOv3 uses a much deeper network Darknet-53. Using the same error rate measures for 100-class ImageNet Top1 and Top5, as well as Single Crop 256X 256,on a Titan X GPU, the authors mention that compared with ResNet-101, Darknet-53 has better performance and is 1.5× faster. Although, compared with ResNet-152, Darknet-53 has similar performance, it is still 2× faster.

3.4.3 RetinaNet

It is discovered that there is an extreme foreground-background class imbalance problem in one-stage detectors. It is also believed that this is the central cause which makes the performance of one-stage detectors inferior to two-stage detectors. RetinaNet which is a 2017 ICCV Best Student Paper Awarded paper with more than 500 citations solved the imbalance issue by introducing the focal loss function. (The first author, Tsung-Yi Lin, became a Research Scientist at Google Brain when he was presenting RetinaNet in 2017 ICCV).[7]

RetinaNet is a one-stage detector which uses focal loss. When using focal loss, lower loss is contributed by "easy" negative samples so the loss focuses on "hard" samples, which improves the prediction accuracy. With ResNet as backbone for feature extraction, plus two task-specific subnetworks for classification and bounding box regression.

RetinaNet Architecture

The architecture has ResNet at its backbone. Figure 3.5 depicts the RetinaNet architecture. It is used for deep feature extraction. Feature pyramid network (FPN) is used on top of ResNet for constructing a rich multi-scale feature pyramid from one single resolution input image. (Originally, FPN is a two-stage detector which has state-of-the-art results). The anchors have the areas of 32 2 to 512 2 on pyramid levels from P3 to P7 respectively, Three aspect ratios 1:2, 1:1, 2:1 are used. For denser scale coverage, anchors of sizes 2 0 , 2 (1/3), 2 (2/3) are added at each pyramid level, In total, 9 anchors are used per level. Across levels, scale is covered from 32 to 813 pixels. For each anchor, there is a length K one-hot vector of classification targets (K: number of classes), and a 4-vector of box regression targets. Anchors are assigned to ground-truth object boxes using Intersection over the union (IOU) threshold of 0.5 and to background if IOU is in [0,0.4). Each anchor is assigned at most one object box, setting the corresponding class entry to one and all other entries to 0 in that K one-hot vector. Anchor is unassigned a class if IOU is in [0.4,0.5) and ignored during training. Box regression is computed as the offset between anchor and assigned object box, or omitted if there is no assignment.

The classification subnet predicts the probability of object presence at each spatial position for each of the A anchors and K object classes. The subnet is a network which applies four 3×3 conv layers, each with C filters and each followed by ReLU activations, followed by a 3×3 conv layer with KXA filters. (K classes, A=9 anchors, and C = 256 filters).The box Regression Subnet is a network to each pyramid level for the purpose of regressing the offset from each anchor box to a nearby ground-truth object, if one exists.

The loss function used in training is called focal loss and is reshaped to down-weight easy examples and thus focus training on hard negatives. A modulating factor is added to the cross entropy loss where γ is tested from [0,5] in the experiment. There are two settings for the Focal loss. First setting is used when an example is misclassified and pt is small, the modulating factor is near 1 and the loss is unaffected. As pt approaches 1 , the factor

goes to 0 and the loss for well-classified examples is down-weighted. The second setting is for the focusing parameter. The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted. When $\gamma = 0$, focal loss is equivalent to cross entropy. When γ is increased, the effect of the modulating factor is likewise increased. (γ=2 works best in experiment.)

$$modulating factor = (1 - pt)^{\gamma} \tag{3.1}$$

$$Focal loss = (1 - pt)^{\gamma} \log pt \tag{3.2}$$

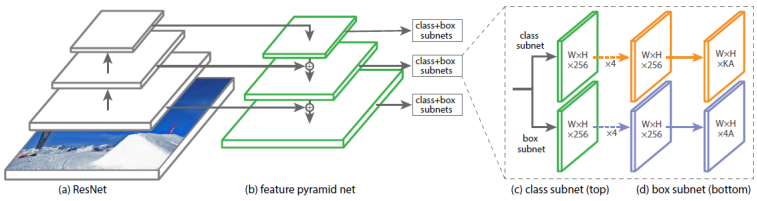

Figure 3.5: Retina-net architecture

The previous figure 3.5 describes the architecture of Retinanet,mainly the Feature pyramid network (FPN) concept. The network only decodes box predictions from at most 1k top-scoring predictions per Feature pyramid network (FPN) level after thresholding detector confidence at 0.05. The top predictions from all levels are merged and non-maximum suppression (NMS) with a threshold of 0.5 is applied to yield the final detections. Compared to existing one-stage detectors, it achieves a healthy 5.9 point AP gap (39.1 vs. 33.2) with the closest competitor,Deconvolutional Single Shot Detector (DSSD). Compared to recent two-stage methods, RetinaNet achieves a 2.3 point gap above the top-performing Faster R-CNN model based on Inception-ResNet-v2TDM.

3.4.4 Optical Flow Estimation

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. We need a way to estimate the change of a set of frames for a specific car to see the changes in its flow, so we want to calculate the optical flow to be able to describe its flow in a fast and good accuracy.

Horn and Schunck [8] method for optical flow estimation formulates the problem of optical flow estimation as a variation problem. The desired vector field h is defined as the minimizer of a certain energy function J(h). This function has two terms: a data attachment term, given by the optical flow constraint, and a regularity term that is based on the gradient of the flow:

$$J(h) = \int\int [(I_x u + I_y v + I_t])^2 + \alpha^2(||\nabla u||^2 + ||\nabla v||^2) d_x d_y] \tag{3.3}$$

where α is a parameter to control the weight of the smoothness term compared to the optical flow constraint. The parameter α is squared so that its units are units of grey-level, and it can be regarded as the intensity of an additive Gaussian noise present in the input images. This energy model uses quadratic functions in both terms. This assumes that the image noise and the flow derivatives are expected to follow a Gaussian distribution. A direct consequence of this kind of function is that the method is very sensitive to the presence of noise and the computed flow fields are very smooth. These shortcomings have led to the appearance of many research works that try to deal with these limitations. The minimization of the above function yields the following Euler-Lagrange equations:

$$I_x u^2 + I_{xy} v^2 = \alpha^2 div(\nabla u) - I_x I_t \tag{3.4}$$

$$I_{xy} u^2 + I_y v^2 + = \alpha^2 div(\nabla v) - I_y I_t \tag{3.5}$$

The Laplacian can be approximated with the following expressions, which will be useful for the discretization below:

$$div(\nabla u) \simeq (\tilde{u} - u) \tag{3.6}$$

$$div(\nabla v) \simeq (\tilde{v} - v) \tag{3.7}$$

where u and v bar are local averages of (u, v). This approximation is analogous to the commonly used "difference of gaussians", where the Laplacian operator is approximated by the difference of blurred versions of the image. In this case the smallest blur is zero. Solving the equations 3.4 and 3.5 above for (u, v) and rearranging the terms, we obtain the following system of equations

$$(\alpha^2 + I_x^2 + I_y^2)(\tilde{u} - u) = -I_x(I_x \tilde{u} + I_y \tilde{v} + I_t) \tag{3.8}$$

$$(\alpha^2 + I_x^2 + I_y^2)(\tilde{v} - v) = -I_y(I_x \tilde{u} + I_y \tilde{v} + I_t) \tag{3.9}$$

Writing these equations for each pixel of the input images, we obtain a sparse system of linear equations. This system can be solved efficiently with an iterative scheme. The partial derivatives, I_x, I_y and I_t, are approximated using forward differences and averaging between two consecutive frames.The local averages ($\tilde{u}$, ($\tilde{v}$) are estimated from the eight neighbors of (u, v) as:

$$\tilde{u} \simeq \frac{1}{6}(u^n_{i-1,j} + u^n_{i+1,j} + u^n_{i,j-1} + u^n_{i,j+1}) + \frac{1}{12}(u^n_{i-1,j} + u^n_{i+1,j} + u^n_{i,j-1} + u^n_{i,j+1}) \tag{3.10}$$

$$\tilde{v} \simeq \frac{1}{6}(v^n_{i-1,j} + v^n_{i+1,j} + v^n_{i,j-1} + v^n_{i,j+1}) + \frac{1}{12}(v^n_{i-1,j} + v^n_{i+1,j} + v^n_{i,j-1} + v^n_{i,j+1}) \tag{3.11}$$

3.4.5 Object Tracking techniques

This section reviews three main tracking techniques; namely. Lucas-Kanade, Mosse and Kalman Filter based.

Object tracking is the task of taking an initial set of object detections, creating a unique ID for each of the initial detections, and then tracking each of the objects as they move around frames in a video, maintaining the ID assignment. After vehicle detection, they should track for (n) numbers of frames to evaluate the changes happening to them over these frames and decide whether there is an accident or not.

Lucas-Kanade Tracker

The Lucas-kanade [9]is one of various is a tracker based on optical flow techniques described in section 3.2.1. The Lucas-Kanade algorithm basically computes the three partial derivatives in the given linear equation (the gradients of a single image with respect to x and y and the change in intensity of each pixel between images) and solves a least-squares estimation problem using a window of pixels around each one to compute the best fitting V_x and V_y.

$$\frac{dI}{dx} * V_x + \frac{dI}{dy} * V_y + \frac{dI}{dt} = 0$$

The processing time depends mainly on the number of features detected for every vehicle object which can vary according to the distance between the vehicles and the camera.

Finding features for every vehicles can be obtained yhrough corner detection.The most known algorithms in finding corners for the Lucas Kanade tracking method are the Harris Corner Detection [10] and J. Shi and C. Tomasi [11] which is a small modification to harris corner detection method showing better results compared to Harris Corner Detector. The only change between the two methods was in the scoring function, which became the minimum of the two eigenvalues.Thus, J. Shi and C. Tomasi Algorithm was selected [11]. In short to find a corner, find the difference in intensity for a displacement of (u,v) in all directions. This is expressed as below:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y]^2 \tag{3.12}$$

The window function is either a rectangular window or gaussian window which gives weights to pixels underneath,to maximize this function E(u,v) for corner detection -which means to

maximize the second term-. Applying Taylor Expansion to the above equation and using some mathematical steps, we get the final equation as follows:

$$E(u,v) \simeq \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.13}$$

where:

$$M = \sum w(x,y) \begin{bmatrix} I_{x,x} & I_{x,y} \\ I_{x,y} & I_{y,y} \end{bmatrix} \tag{3.14}$$

Ix and Iy are image derivatives in x and y directions respectively, and then apply the score function to select the corners:

$$R = min(\lambda_1 \lambda_2) \tag{3.15}$$

MOSSE Tracker

Minimum Output Sum of Squared (MOSSE) [12] is a car tracker that runs using a correlation filter which depends on the frequency domain. Using fast Fourier transform, it can get the frequency transformation of the car image fast. Since MOSSE filter is a correlation filter so it can track complex objects through rotations, occlusions and other distractions. MOSSE filter produces stable correlation filters when initialized using a single frame. It is also robust to variations in lighting, scale, pose and non-rigid deformations.

For better understanding of the algorithm consider a target car which is initially selected based on a small tracking window centered on the car in the first frame. Then a sequence of stages takes place. The first stage is the preprocessing stage. First, the pixel values are transformed using a log function -which helps with the low contrast lighting situations- ,The pixel values are normalized then the image is multiplied by a cosine window which gradually reduces the pixel values near the edge to zero which will put more focus on the center of the cut image.Afterwards the second stage starts which is filter training.This stage will require training images.The current image is used but after going through some rotations, then a trained filter can be obtained using these rotated images.From this point the tracking and filter training work together, The tracked car is tracked by correlating the filter over a search window in next frame, the location corresponding to the maximum value on the correlation output indicates the new position of the target, an online update is then performed based on the new location, The correlation is computed in the Fourier domain using Fast Fourier transform.

To compute the correlation as stated above,the Fast Fourier transform is applied to both the input car image and the transfrom filter.The element wise multiplication between the transform filter and the car image is computed based on the Convolution Theorem which states that correlation becomes an element-wise multiplication in the Frequency domain. Using the $\cdot$ symbol to explicitly denote element-wise multiplication and $*$ to indicate the complex conjugate, correlation takes the form:-

$$G = F \cdot H^* \tag{3.16}$$

The correlation output is transformed back into the spatial domain using the inverse fast Fourier transform. The max value around the center is then obtained which will indicate the new position for the car with the help of psr (peak to sidelobe ratio) which indicates the goodness of the tracking filter at the moment. Peak to sidelobe ratio can be used to detect occlusions or tracking failure, to stop the online update and to reacquire the track if the object reappears with a similar appearance.

Kalman Filter Based Trackers

Kalman filter is used to establish an object motion model, using the current object's information to predict the object's position, so that the search scope may be reduced. The search time of moving objects is also reduced and one achieves fast tracking. It also allows for establishing a relationship through moving object features to deal with separation after objects are merged [13].

3.4.6 Variance measure algorithms

Violent Flow Descriptor (VIF) [14] considers the statistics of magnitude changes of flow vectors over time. In order to get these vectors, an optical flow algorithm is used like Lucas-Kanade or Horn-Schunck. VIF depends heavily on the magnitude of optical flow vectors, these vectors are calculated for each pixel in two consecutive frames, these vectors could represent the motion of objects in a video scene, where the bigger vectors represent the objects with more movement. The VIF descriptor has a very low computational cost and acceptable accuracy.

Given a video sequence S of frames f1, f2, . . . two related but different tasks are considered. The first is anomaly classification, The video S is assumed to be segmented temporally, containing T frames portraying either anomaly or non-anomaly event behavior. The goal is to classify S accordingly. The second is anomaly detection, assuming an input stream of frames the goal is to detect the change from anomaly to non-anomaly behavior, with the shortest delay from the time (frame) in which the change occurred.

The algorithm proceeds as follows, for a sequence of frames "S", VIolence Flows (VIF) descriptor is produced by first estimating the optical flow between pairs of consecutive frames. This provides for each pixel $P_{x,y,t}$ where t is the frame index, a flow vector ($u_{x,y,t}$, $v_{x,y,t}$), matching it to a pixel in the next frame t + 1. Here, we consider only the magnitudes of these vectors:

$$m_{x,y} = \sqrt{(u_{x,y,t}^2 + v_{x,y,t}^2)} \tag{3.17}$$

The rationale is that although flow vectors encode meaningful temporal information, their magnitudes are arbitrary quantities; they depend on frame resolution, different motions in

different Spatio-temporal locations, etc. By comparing magnitudes, meaningful measures of the significance of observed motion magnitudes in each frame compared to its predecessor can be obtained. Hence, for each pixel in each frame, a binary indicator $b_{x,y,t}$ is obtained reflecting the significance of the change of magnitude between frames:

$$b_{x,y,t} = \begin{cases} 1 & \text{if } |m_{x,y,t} - m_{x,y,t-1}| > \theta \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

Where θ is a threshold adaptively set in each frame to the average value of $|m_{x,y,t} - m_{x,y,t-1}|$. Doing so provides us with a binary, magnitude-change, significance map b_t for each frame ft. Next, the mean magnitude-change map is computed by simply averaging these binary values, for each pixel, overall the frames $f_t \in S$:

$$b_{x,y,t} = \frac{1}{T} \sum b_{x,y,t} \tag{3.19}$$

In its simplest form, the VIF descriptor is a vector of frequencies of quantized values $b_{x,y}$. The VIF descriptor is therefore produced by partitioning b into M $\times$ N non-overlapping cells and collecting magnitude change frequencies in each cell separately. The distribution of magnitude changes in each such cell is represented by a fixed-size histogram as shown in figure 3.6. These histograms are then concatenated into a single descriptor vector

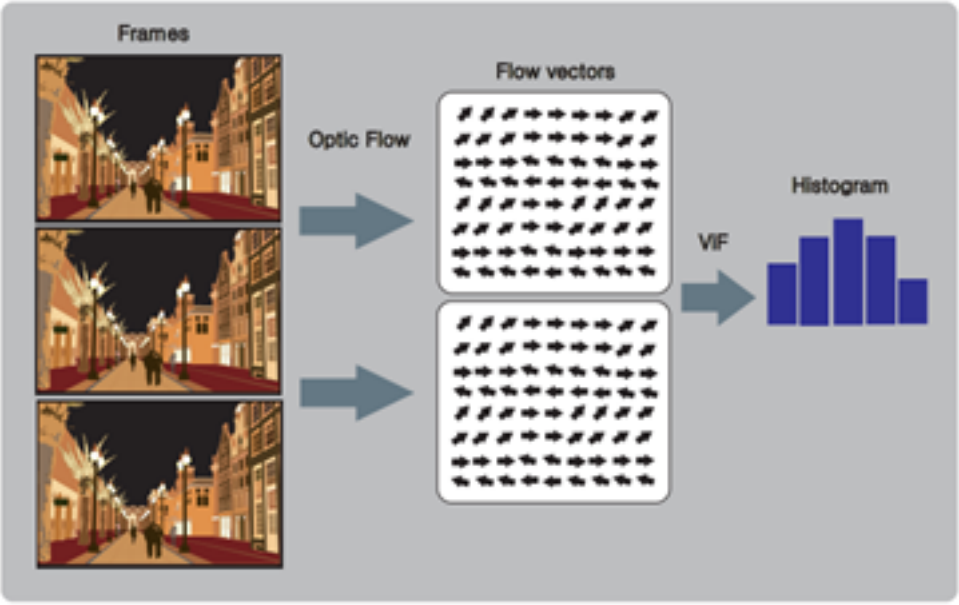

Figure 3.6: Explaining VIF

Support vector machine (SVM) is later used as a classifier with a linear kernel, taking as input the result of a VIF descriptor (feature vector with 336 values.) .The SVM classifiers can help in deducing the presence of a crash. .

3.4.7 Implemented approach

Our comparative study shows that the most suitable architecture for a real-time task is the You Only Look Once (Yolo) architecture. The YOLOV3 is chosen since it achieves a good trade-off between accuracy and speed, settling in a position between YOLO and YOLOv2. YOLOV3 will be used for detecting and tracking vehicles in a real-time manner. It may be used to detect vehicles or pedestrians. A transfer learning approach is used by exploiting the pre-trained weights of YOLOV3 trained on Large data sets. This is very useful since a lot of capabilities needed for training like storage and GPU power are not available. After detection MOOSE tracker is used for vehicle tracking because of its suitability for real-time tasks. Mosse tracker uses correlation filter so it can be used in complex problems unlike Lucas-Kanade which had poor real-time performance when running it through rotations, occlusions, and other distractions, Mosse filter produces stable correlation filters when initialized using a single frame and it's robust to variations in lighting, scale, pose and non-rigid deformations. Collision estimation techniques inspired by gaming engines is used to filter out the videos that are bad candidates for accidents to save time and enhance the performance. Finally, the VIF descriptor is used to detect the variations of flow vectors computed from the frames captured by the tracker for every vehicle individually because of the very low cost and acceptable accuracy. Those vectors are fed to the SVM classifier for crash classification.

4. System Design and Architecture

The project design is modular, hence the implementation phase was performed in a parallel manner, each member was either working on the processing side including: tracking and detection modules, or the client side including the backed development and GUI simultaneously In addition to the simulation camera simulation environment.

4.1 Overview and Assumptions

The system is assumed to be deployed based on the client -server methodology. Most of the processing is assumed to be on the server side .The Camera is assumed to be a normal CCTV camera with no special hardware. The server is assumed to be GPU accelerated and the client is a normal web client.

4.2 System Architecture

Our system is composed of 3 main modules depicted in Figure 4.1 These are namely; the Processing Module, the server module and the client module

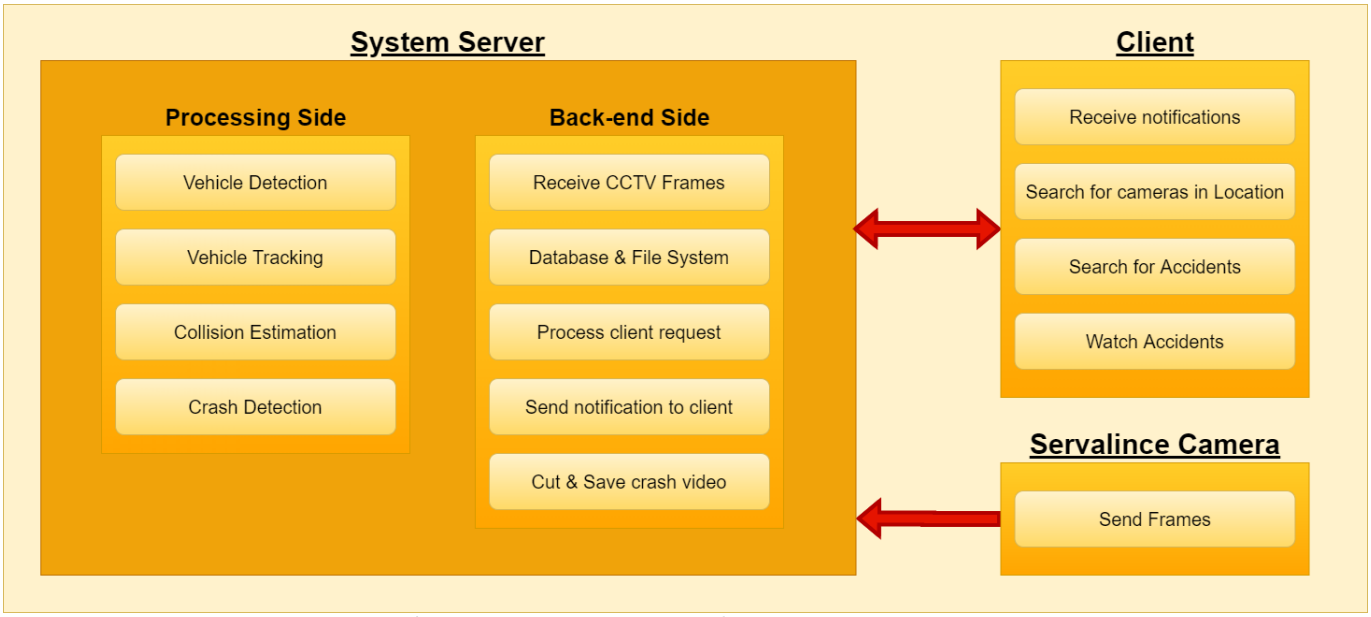

Figure 4.1: system architecture

4.3 Surveillance Camera and Image processing Module

As its name depicts, the surveillance camera and image processing module is concerned with the surveillance camera, reading, decoding frames received from it and sending them to the processing server. The module is also responsible for sending groups of frames - 30 frames - per second.The current implementation is a software simulation specifics to follow.

4.3.1 Functional Description

The module receives the camera stream, decodes them and sends them to the server side.

4.3.2 Modular Decomposition

This module is just one component to receive the camera stream.it is simple; it just compresses the camera stream every 30 frames and sends them to the server directly every second without any other functionalities.

4.3.3 Design Constraints

The Design constraints are choosing a suitable camera in the deployment stage. The challenge is creating a true simulation environment for the behavior of the camera. The module Also needs to be as simple as possible; so that there is no need of high computational power,thus any simple microcontroller can do the job. Beside it should send the footage very fast so detection can happen in real time.

4.4 System Server

This module is responsible for most processing of the system as will be illustrated in the coming sub-sections

4.4.1 Functional Description

Generally this module receives a stream of frames from the camera, and processes the frames. Then it sends a notification to the client if an accident occurred,the client can also send a query to the server to display an accident video or view old accidents.

4.4.2 Modular Decomposition

The system server is divided into two main parts; the processing part and the web server part. The processing part receives the stream of frames from from the server side, It performs the computer vision algorithms for detecting and tracking vehicles. It then detects any signs of a crash in the frame. After processing it sends its results (crash/non-crash)to the web server.

The web server's job is to receive the stream from CCTV camera and send it to the processing side then receives the results from the processing side and notifies the clients and also provides them with video streams from the cameras. The clients can be hospitals, police stations,..etc. It also saves the stream as a video in it's database in case it was needed by the authorities, in case of any investigation held by the authorities.

4.4.3 Design Constraints

To ensure real time performance the server has to be GPU enabled and the code of frame processing should be written in a way that can be computed by a GPU.also it must be able to process 30 frames per second to ensure real time performance.

4.5 The client Module

This module is responsible for receiving notifications and sending client requests to the server

4.5.1 Functional Description

The web client should handle the alerts received from the web server and receive the video streams.This is achieved by receiving notification on its port and sending its requests to the server using json on the server port,it can also conduct a search using location or time -or both- information to get some data from the server side database.

4.5.2 Modular Decomposition

The client server is divided into 2 main modules; the the alert handling module and the searching module.The Alert handling module is responsible for receiving alerts from the server and notifying the client through a suitable GUI .The searching module s responsible for allowing the client to request any old data from the server, or search for accidents based on city, location, date, time or even all of these together.

4.5.3 Design Constraints

This module on one hand, needs to be intuitive, user friendly and organized in terms of the graphical user interface.On the other hand, it should provide a stable and robust connection to the server.since it always listens, receives notifications from the server and displays them to the client as early as possible.

.

5. System Testing and Verification

The testing of the project was conducted in an agile manner,each module was tested individually after completing its implementation,Its functionality was then tested against existing modules to check if the integration between different modules was successful.Unit tests were done on almost all the modules. The verification of the project was through testing it with a test set representing different types of crashes and checking its accuracy in detecting them.

5.1 Testing Setup

For testing the project a suitable hardware and software were needed.A Nvidia GPU with at least 2GB was needed to be able to run our model,check the accuracy and measure the speed. The GPU has to be Nvidia because it is the only one that supports CUDA.There were no strict requirements considering the software setup.Pytest, A Python unit test testing framework, was used for testing.Testing was conducted on devices with both windows and Linux Operating systems.

5.2 Testing Plan and Strategy

As previously stated,Testing was conducted in an agile manner along with the development process.Our strategy was testing any finished module before integrating it with other modules.Integration testing is then conducted to ensure no integration failures existed.

5.3 Module Testing

In this section module testing will be elaborated for each module independently.we shall start by quickly recapping the modules in our system:Surveillance Camera module,System Server,client Module.

5.3.1 Testing the camera module

Testing the camera module was concerned with two main functionalities:

- Receiving the Camera steam
- Sending the stream to the server

Unit testing was used to test each functionality.The tests were as follows:

Table 5.1: Receiving the camera stream test

TC-ID1	Receiving the camera stream
Purpose	Checking for Camera Stream Delivery
Prerequisite	1.The server and the Camera should be connected
Priority	High Priority
Steps	1.Sending Frames from camera to camera module
Expected Results	Frame Received and Decoded correctly
Actual Results	Frames Received and Decoded correctly
State	Pass
Issue	-
Execution Date	5/3/2020

Table 5.2: Sending the camera stream to server test

TC-ID2	Sending the camera stream to server
Purpose	Checking that Camera Stream is sent to server module
Prerequisite	1.The server and the Camera should be connected
Priority	High Priority
Steps	1.Sending Frames from camera
Expected Results	Frames Received correctly at server module
Actual Results	Frames Received correctly at server module
State	Pass
Issue	-
Execution Date	5/3/2020

5.3.2 Testing Server module

Testing server module concentrates on testing the vehicle detection, vehicle tracking, and collision detection modules on a sequence of frames, it also tests sending a notification to the client in case of a crash. The tests were as follows.

Table 5.3: Checking detecting objects accuracy test

TC-ID3	Checking detecting objects accuracy
Purpose	Checking detecting important objects in the frames,mainly:cars,vehicles and pedestrians.
Prerequisite	1.Model weights should be uploaded on server
Priority	High Priority
Steps	1.Provide a frame to the detection function
Expected Results	Objects should be detected with an accuracy ranging between (95-100%)
Actual Results	Objects are detected with an accuracy ranging between (96-98%)
State	Pass
Issue	-
Execution Date	15/4/2020

Table 5.4: Checking detecting objects speed test

TC-ID4	Checking detecting objects speed
Purpose	Checking detecting important objects in the frames,mainly:cars,vehicles and pedestrians within a strict timing requirement
Prerequisite	1. Model weights should be uploaded on server 2. A GPU should be available
Priority	High Priority
Steps	1.Provide a frame to the detection function
Expected Results	Objects should be detected with an accuracy ranging between (95-100%) with a requirement of 5fps.
Actual Results	Objects are detected with an accuracy ranging between (96-98%) with a requirement of 5fps.
State	Pass
Issue	-
Execution Date	15/4/2020

Table 5.5: Checking detection module behaviour with a invalid input

TC-ID5	Checking detection module behaviour with a invalid input
Purpose	Checking that the detection module can handle invalid inputs
Prerequisite	1. Model weights should be uploaded on server 2. A GPU should be available
Priority	High Priority
Steps	1.Provide a Null input or input of a wrong type to the detection function.
Expected Results	The module should not proceed and terminate immediately
Actual Results	The module should not proceed and terminate immediately
State	Pass
Issue	-
Execution Date	15/4/2020

Table 5.6: Checking the method of loading pre-saved YOLO results handles invalid paths

TC-ID6	Checking the method of loading pre-saved YOLO results handles invalid paths
Purpose	This module is used to load pre-saved YOLO results. This test verifies it can handle an invalid path
Prerequisite	-
Priority	Medium Priority
Steps	1.Function should be called with one argument, the file name
Expected Results	File does not exist. So, it terminates and returns None
Actual Results	Function terminates and returns None.
State	Pass
Issue	-
Execution Date	13/4/2020

Table 5.7: Checking the method of loading pre-saved YOLO loads the correct files

TC-ID7	Checking the method of loading pre-saved YOLO loads the correct files
Purpose	This module is used to load pre-saved YOLO results. The function should load the saved data of the correct video, not just any data.
Prerequisite	-
Priority	High Priority
Steps	1. Function should be called with one argument, the file name. 2. Compare results to the expected results.
Expected Results	Both returned results and expected results should be identical.
Actual Results	Both results are identical.
State	Pass
Issue	-
Execution Date	13/4/2020

Table 5.8: Checking initialization of tracker for detected objects.

TC-ID8	Checking initialization of tracker for detected objects.
Purpose	Checking tracking an object in a frame
Prerequisite	1. Model weights should be uploaded on server 2. A GPU should be available 3. Object detection function should be working properly
Priority	High Priority
Steps	1. Provide detected objects from detection function along with the bounding boxes,labels and confidences to tracking function 2. Tracker should be called for every detected object.
Expected Results	For every detected object, an instance of the tracker class should be initialized with the object parameters.
Actual Results	For every detected object, an instance of the tracker class is initialized with the object parameters.
State	Pass
Issue	-
Execution Date	23/4/2020

Table 5.9: Checking optical flow results between two frame.

TC-ID9	Checking optical flow results between two frame
Purpose	Horn Scunck method is used to estimate the optical flow between two frames.
Prerequisite	-
Priority	High Priority
Steps	1. Call the method process from the class Horn Schunk with two arguments, the Two frames.
Expected Results	The returned result should be equal to the pre-calculated result.
Actual Results	The returned result is equal to the pre-calculated result.
State	Pass
Issue	-
Execution Date	20/2/2020

Table 5.10: Checking the results of the derivatives between two frame .

TC-ID10	Checking the results of the derivatives between two frame
Purpose	Verify that the calculated derivatives are correct.
Prerequisite	-
Priority	High Priority
Steps	1. Call the method derivative from the class Horn Schunk with two arguments, the Two frames.
Expected Results	The returned result should be equal to the pre-calculated result.
Actual Results	The returned result is equal to the pre-calculated result.
State	Pass
Issue	-
Execution Date	20/2/2020

Table 5.11: Checking crash detection model accuracy

TC-ID11	Checking crash detection model accuracy
Purpose	1.Dataset loaded and divided to training set and test set.
Prerequisite	1. Train the model on the training set. 2. Estimate accuracy on the test set.
Priority	High Priority
Steps	1. Call the method derivative from the class Horn Schunk with two arguments, the Two frames.
Expected Results	Crashes should be detected within an accuracy range (80-100%)
Actual Results	Crashes are detected within an accuracy range (70-85%
State	Pass
Issue	-
Execution Date	25/4/2020

Table 5.12: Sending crash notification to client

TC-ID12	Sending crash notification to client
Purpose	Send crash notification to client
Prerequisite	1. Model weights should be uploaded on server 2. A GPU should be available 3. Object detection ,tracking function and VIF functions should be working properly 4. A crash occurred
Priority	High Priority
Steps	1. A crash is detected by The VIF descriptor and feedback is sent to the server back-end. 2. server sends an alert to the client back-end.
Expected Results	1. A notification is sent to client
Actual Results	1. A notification is sent to client
State	Pass
Issue	-
Execution Date	5/5/2020

5.3.3 Testing client module

The client should handle the alerts received from the web server and receive the video streams. The testing was as follows.

Table 5.13: Receiving crash notification from server test

TC-ID13	Receiving crash notification from server
Purpose	Receive crash notification to client
Prerequisite	1. server and client should be connected to internet 2. A crash occurred
Priority	High Priority
Steps	1. A notification is sent from server 2. A notification is received by the client. 3. A notification appears on the client interface.
Expected Results	1. A notification is sent from server and displayed to the client.
Actual Results	1. A notification is sent from server
State	Pass
Issue	-
Execution Date	5/5/2020

Table 5.14: Displaying crash footage on client side test

TC-ID14	Displaying crash footage on client side
Purpose	Display the crash stream on the client side
Prerequisite	1. server and client should be connected to internet 2. A crash occurred 3. client is notified by crash
Priority	Medium Priority
Steps	1. A request is sent to the server. 2. The server replies with the required footage.
Expected Results	1. The footage should be received from the server.
Actual Results	1. The footage should be received from the server.
State	Pass
Issue	-
Execution Date	25/5/2020

5.4 Integration testing

As elaborated in the previous illustration the functionalities of modules are tested with respect to others needing them in other correlated modules.This supported integrating different modules together.The whole system functionality was tested after testing each module independently as illustrated in the previous section. The following tests are the integration tests conducted to assure correct integration between modules.

These tests test the integration between the camera and server modules.

Table 5.15: Checking detection module behaviour after the integration

TC-ID15	Checking detection module behaviour after the integration
Purpose	Checking that the detection module behaves and returns results in a certain format as agreed upon in the designing phase.
Prerequisite	1. Model weights should be uploaded on server 2. A GPU should be available
Priority	High Priority
Steps	A notification is sent from server
Expected Results	1. function returns the coordinates then the accuracy then the label of the detected object where all of them except the label are float. 2. Displayed results correctly bounds the object
Actual Results	1. function returned the expected order with the expected types 2. Displayed results are correct
State	Pass
Issue	-
Execution Date	1/5/2020

Table 5.16: Checking the method of loading pre-saved YOLO after the integration

TC-ID16	Checking the method of loading pre-saved YOLO after integration
Purpose	This module is used to load pre-saved YOLO results. This test verifies it can handle an invalid path
Prerequisite	1. Program is running in the mode of using pre-saved results.
Priority	High Priority
Steps	1.Function should be called with one argument, the file name
Expected Results	1. Results of correct type and order should be returned. 2. Program should keep running normally.
Actual Results	1. Results returned in correct order with correct types. 2. Program kept running as usual.
State	Pass
Issue	-
Execution Date	1/5/2020

Table 5.17: Checking object tracking behavior after the integration

TC-ID17	Checking object tracking behavior after the integration
Purpose	Tracker should be updated with every frame the model receives to identify the new position of the object and save results of the changes over the sequence of the frames.
Prerequisite	1. Model weights should be uploaded on server 2. A GPU should be available 3. Object detection function should be working properly 4. .Instances of the tracker for every object should be initialized
Priority	High Priority
Steps	1. Provide the class with the new frame.
Expected Results	1. .Tracker should estimate the new position of the object. 2. Tracker should save the results of the new frame. 3. Tracker should update its estimation for the object speed.
Actual Results	1. .Tracker estimates the new position of the object. 2. Tracker save the results of the new frame. 3. Tracker updates its estimation for the object speed.
State	Pass
Issue	-
Execution Date	1/5/2020

Table 5.18: Checking VIF descriptor behaviour after the integration

TC-ID18	Checking VIF descriptor behaviour after the integration
Purpose	Verify that after the integration instance if the class VIF is initialized correctly and the sequence of frames is passed to be processed and results are returned as expected.
Prerequisite	1. Model weights should be uploaded on server 2. A GPU should be available 3. Object detection and tracking function should be working properly
Priority	High Priority
Steps	1. Provide tracked objects from the tracking function for the sequence of frames.
Expected Results	1. .Instance of the class should be initialized correctly. 2. A result should be returned indicating the probability of the
Actual Results	1. Instance of the class is initialized correctly. 2. A result is returned indicating the probability of the existence of an accident.
State	Pass
Issue	-
Execution Date	5/5/2020

5.5 Testing Schedule

The testing schedule was set in order to fulfill the agile strategy stated previously.The schedule focused on unit testing of each function inside a module directly after finishing its implementation. Then testing them again after the integration.

Table 5.19: Testing schedule

Test	Start Date	End Date
Checking optical flow results between two frame	20/2/2020	20/2/2020
Checking the results of the derivatives between two frame	20/2/2020	20/2/2020
Receiving camera stream	5/3/2020	5/3/2020
Sending camera stream	5/3/2020	5/3/2020
Checking the method of loading pre-saved YOLO results handles invalid paths	13/4/2020	13/4/2020
Checking the method of loading pre-saved YOLO loads the correct files	13/4/2020	13/4/2020
Checking detecting object accuracy	15/4/2020	15/4/2020
Checking detecting object speed	15/4/2020	15/4/2020
Checking detection module behaviour with a invalid input.	15/4/2020	15/4/2020
Checking initialization of tracker for detected objects	23/4/2020	23/4/2020
Checking crash detection model accuracy	25/4/2020	25/4/2020
Checking detection module behaviour after integration	1/5/2020	1/5/2020
Checking the method of loading pre-saved YOLO after the integration	1/5/2020	1/5/2020
Checking object tracking behavior after the integration	1/5/2020	1/5/2020
Checking VIF descriptor behaviour after the integration	5/5/2020	5/5/2020
Sending crash notification to client	5/5/2020	5/5/2020
Receiving crash notification from server	5/5/2020	5/5/2020
Displaying crash footage on client side	25/5/2020	25/5/2020

5.6 Comparative Results to Previous Work

Compared to previous work, we achieved very good results with respect to accuracy. The VIF descriptor associated model was trained using 65 videos and calculated its accuracy using 10 unseen videos. The accuracy of our system was measured using the results of 31 videos with different lighting conditions and with different camera angles. The following table illustrates this.

Table 5.20: Comparative Results to Previous Work

Module	Accuracy	Our Accuracy
YOLO	96-98%	96-98%
Vif descriptor	81% - [15]	78%
Collision Estimation	No Previous Work	93%
Whole System	81% - [15]	95%

6. Conclusions and Future Work

In this Chapter the conclusion ,results and recommendations are provided. An illustration of all the faced challenges throughout the project is presented and a brief of any future work can be found as well.

6.1 Faced Challenges

The most obvious and the hardest challenge was the fact that the project had to meet the real time requirement.There was no other option except complying with this requirement or the project would lose its value.We faced challenges in the development process like dealing with multithreading.Special hardware like GPUs was needed for our models to run so we had to search for free sessions on cloud servers like google colab.Finding a dataset addressing our problem was hard as well,so we had to overcome the shortage of data in a clever way like adding the collision detection module , search for pretrained weights and depend on more classical approaches.

6.2 Gained Experience

After working in this project we gained a lot of experience in some computer vision domains like object detection and tracking .We learned more about some machine learning frameworks like pytorch,how they represent the data , the graph theory behind these frameworks and how they deal with different hardware platforms whether CPUs or GPUs in deployment phase.We experienced multithreading and parallel programming to achieve a real time performance.Finally we learned some new GUI libraries and frameworks like PYQT5.

6.3 Conclusions

Vehicle tracking problem is an area of continuous enhancement and research,Many algorithm can be of great accuracy,however poor real time performance.It depends on the context or the application for one to decide whether to choose accuracy or speed.As for us we made this decision when choosing our tracker where we chose MOOSE rather Lucas-Knade (refer to section three).Crash detection is hard due to Occlusion,blur and other distortion,However based on our own experiment VIF descriptors and SVMs (refer to section three). can achieve acceptable results in detecting them.

6.4 Future Work

Tracking and other computer vision problems are computationally intensive.They require special hardware with sophisticated memory requirements for acceptable and real time performance.Nowadays many optimization techniques are arising to address this problem due to the increasing need of deploying such models on IOT devices with power and memory

constrains.As a future work we may consider optimizing our models to a more compact ones that can run directly on small boards attached with the CCTV cameras to reduce the overhead of sending the camera footage to servers and achieve a more quick response thus saving more lives.This can also help in deserted roads where internet connection may be weak and unreliable or even not found.We may also consider adding OCR for recognizing car plates.This module might be integrated with the processing side on our server to apply OCR on car plates after detecting them.We can also update our database with information about cars' crashes history based on the car plates.Adding this functionality may not be very hard.The system architecture is extendable and no severe modifications will be conducted.The presence of such functionality can be of great importance for the authorities and car owners as well.

Bibliography

[1] *https://www.asirt.org/safe-travel/road-safety-facts/* 2

[2] Asadi-Aghbolaghi, M., Clapes, A., Bellantonio, M., Escalante, H. J., Ponce-López, V., Baró, X., ... & Escalera, S. (2017, May). *A survey on deep learning based approaches for action and gesture recognition in image sequences* IEEE international conference on automatic face and gesture recognition . 8

[3] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. (2017-05-24). *"ImageNet classification with deep convolutional neural networks"*. Communications of the ACM. 60 (6): 84–90. doi:10.1145/3065386. ISSN 0001-0782. 10

[4] Redmon, Joseph (2016) *You Only Look Once: Unified, Real-Time Object Detection.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 12

[5] Redmon, J., & Farhadi, A. (2017) *YOLO9000: better, faster, stronger.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271). 14

[6] Redmon, J., & Farhadi, A. (2018) *Yolov3: An incremental improvement.* . arXiv preprint arXiv:1804.02767. 15

[7] Tsung-Yi Lin ; Priya Goyal ;Ross B. Girshick ;Kaiming He ;Piotr Doll *"Focal Loss for Dense Object Detection"* 16

[8] Andry Maykol G. Pinto, A. Paulo Moreira , Paulo G. Costa and Miguel V. Correia, *"Revisiting Lucas-Kanade and Horn-Schunck"* Journal of Computer Engineering and Informatics, Vol. 1, Iss. 2, PP. 23-29, 2013 17

[9] Andry Maykol G. Pinto, A. Paulo Moreira , Paulo G. Costa and Miguel V. Correia, *"Revisiting Lucas-Kanade and Horn-Schunck"*, Journal of Computer Engineering and Informatics, Vol. 1, Iss. 2, PP. 23-29, 2013 19

[10] Chris Harris and Mike Stephens . *"A Combined Corner and Edge Detector*, 1988 19

[11] Shi, Jianbo and Tomasi, Carlo *Good Features to Track*, Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition,2000 19

[12] David S. Bolme, J. Ross Beveridge, Bruce A. Draper and Yui Man Lui, *"Visual Object Tracking using Adaptive Correlation Filters"*, 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, June 2010 20

[13] Xin Li, Kejun Wang,Wei Wang and Yang Li, *"A Multiple Object Tracking Method Using Kalman Filter"*, International Conference on Information and Automation, Harbin, China, June 2010 21

[14] T. Hassner, Y. Itcher, and O. Kliper-Gross, *"Violent flows: Real-time detection of violent crowd behavior"* , Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference, Providence, Rhode Island, USA., June 2012 21

[15] V. Machaca Arceda and E. Laura Riveros, *"Fast car Crash Detection in Video,"*, 2018 XLIV Latin American Computer Conference (CLEI), São Paulo, Brazil, 2018, pp. 632-637, doi: 10.1109/CLEI.2018.00081.

Appendices

A. Development Platform and Tools

This appendix is referencing the software and hardware requirements used in this project.

A.1 Hardware Platforms

A GPU is required on the server side to maintain accurate and real time performance of deployed algorithms. other than that the simplest computer with the following specifications will be more than enough and already has been used in developing and testing the program:

- Intel core i3 processor (or equivalent)
- 6 Gigabytes of memory
- A Nvidia GPU of at least 4GB RAM
  - The graphics card should support CUDA, so far only Nvidia supports it.
  - A 2GB Nvidia GeForce 920mx was used during development but it is not recommended.

A.2 Software Tools

- Pytorch framework (version 0.14) is used for running YOLO architecture.
- OpenCV Library was used in some algorithms for real time performance.
- Scikit-learn library was used in building a SVM model.
- ZMQ package was used for communications between the three modules.
- SQLite was used for the database.
- PYQT5 toolkit was used for developing the user interface.

No certain operating system is required. Although the project wasn't tried on MACos it's expected to work just fine.

B. Use Cases

The following appendix illustrates all possible use cases.

1. Normal case
   User is trying to do nothing. Camera feed is received by the server and when an accident happens the user is notified.

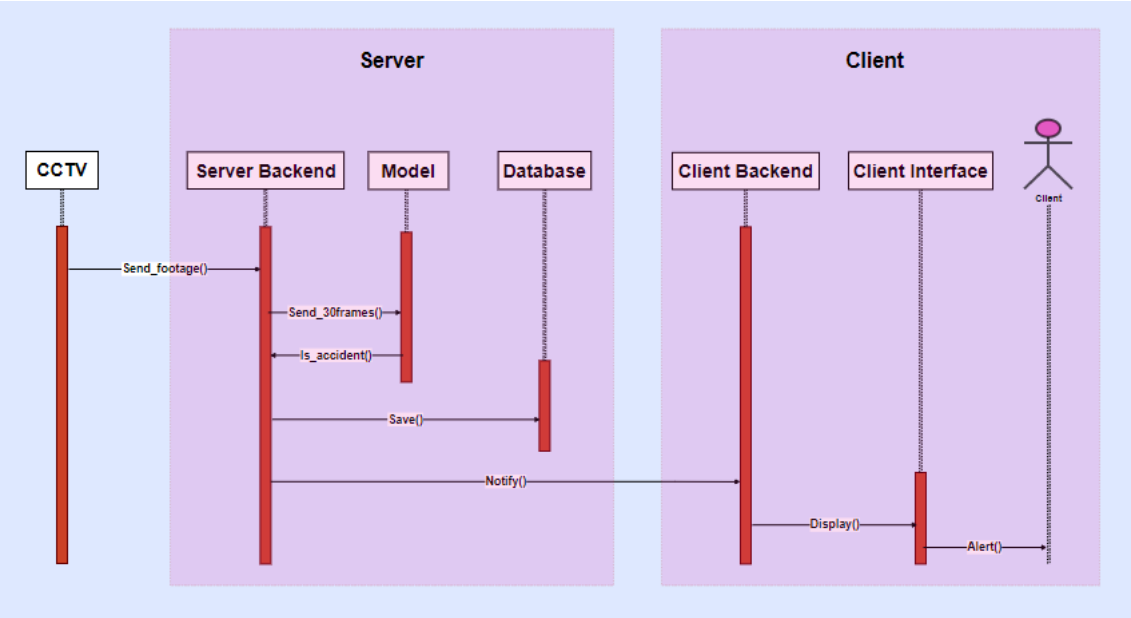

Figure B.1: normal case

2. Request data
   When a user is searching for certain accidents through the provided search mechanism. A request is sent to a server that fetches the data and sends it back to the client.

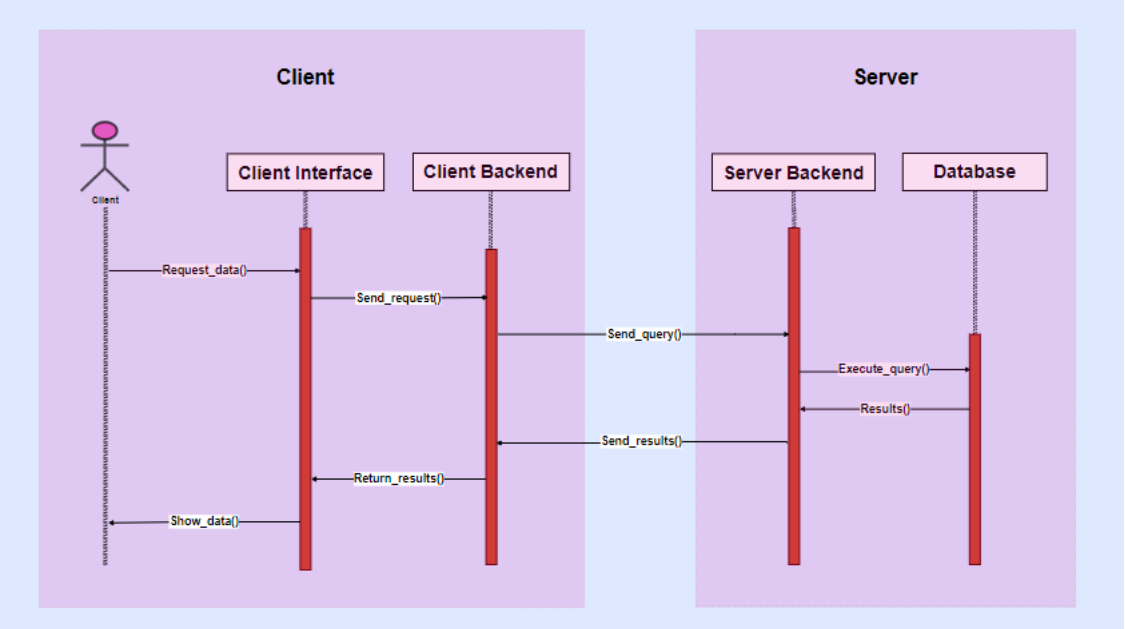

Figure B.2: request data case

3. Play video
   Normally the accident footage is located on the server. So, When a user wants to inspect an accident through watching its footage it sends a request to the server which provides it with the required footage.

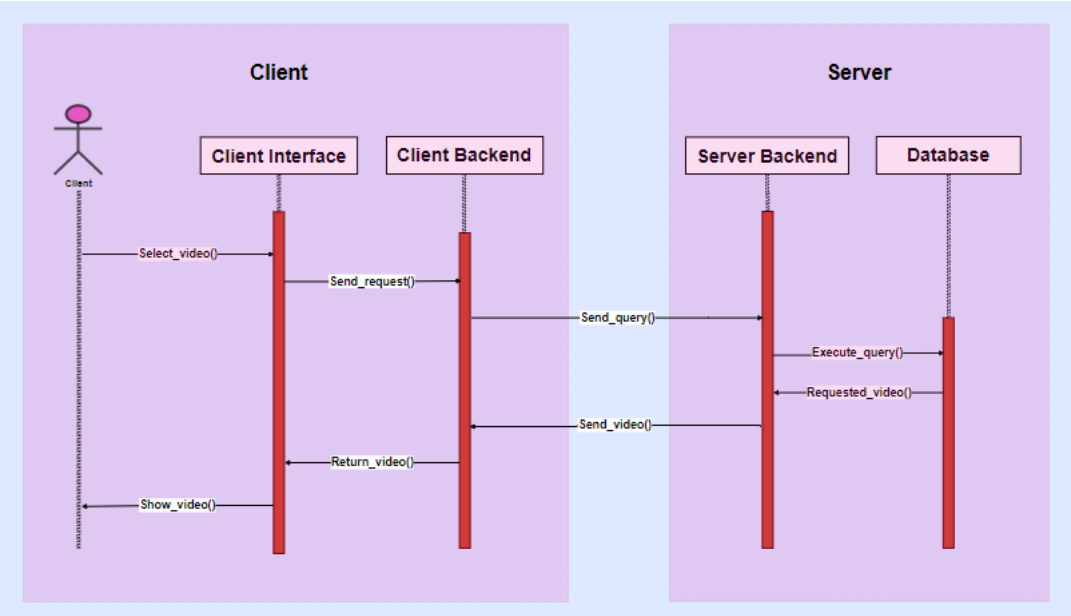

Figure B.3: play video case

4. Camera simulation
   This special form provides the user with the possibility of examining old videos that are not coming from CCTV, by simulating the camera operation. The user selects the video to be processed then the module starts communicating with the server as a camera sending its feed.

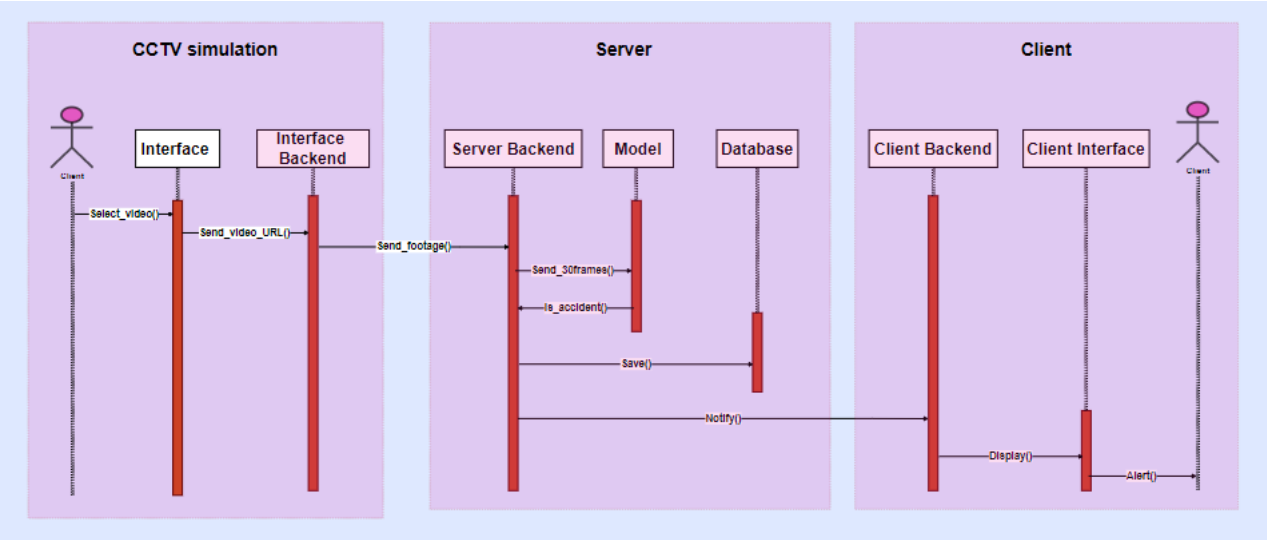

Figure B.4: camera simulation case

C. User Guide

C.1 Client interface

This is the guide of the ordinary user interface. It includes a complete guide on how to use the interface.

- Searching These are the steps you should take to search for accidents. Note that date, time and location are not mandatory; you can leave them unchanged.

  1. Start the program
     Normally it will look like this

Figure C.1: Normal program UI

     If the search results were empty that means that either you have a problem in your database connections and you should contact the database administrator or you just installed the system so there are no accidents to display.

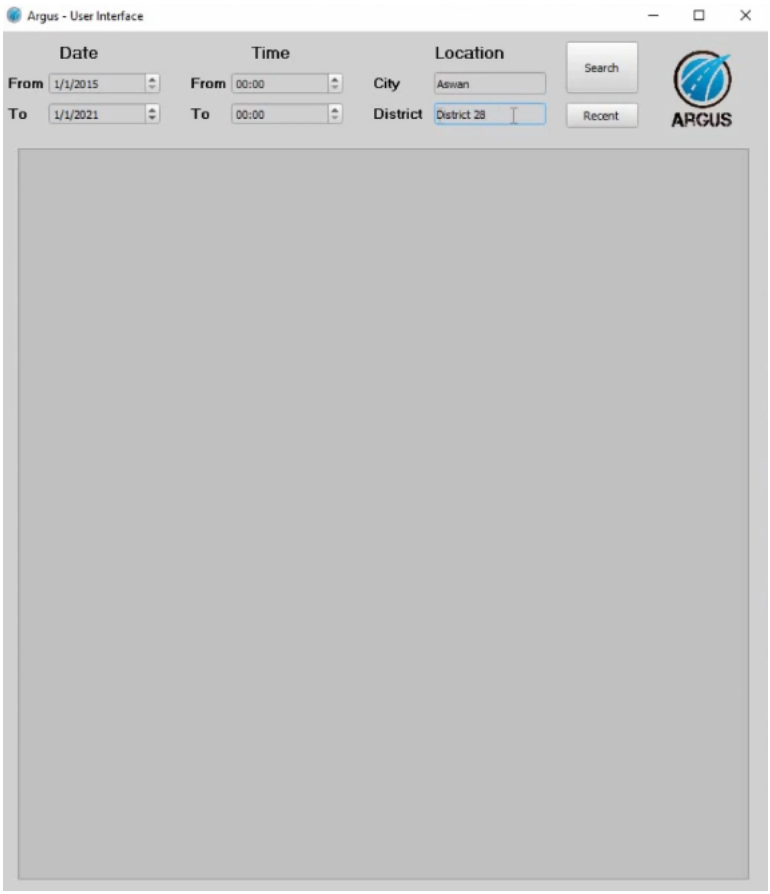

Figure C.2: No displayed accidents

2. Select the date interval you want to search through

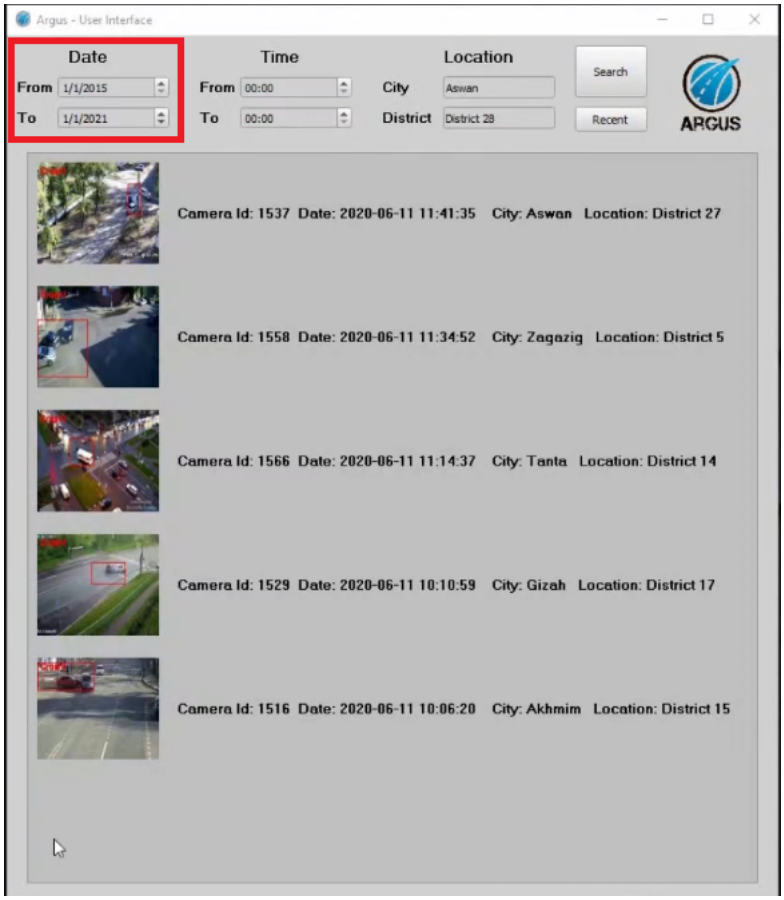

Figure C.3: selecting date

3. Select the time interval you want to search through

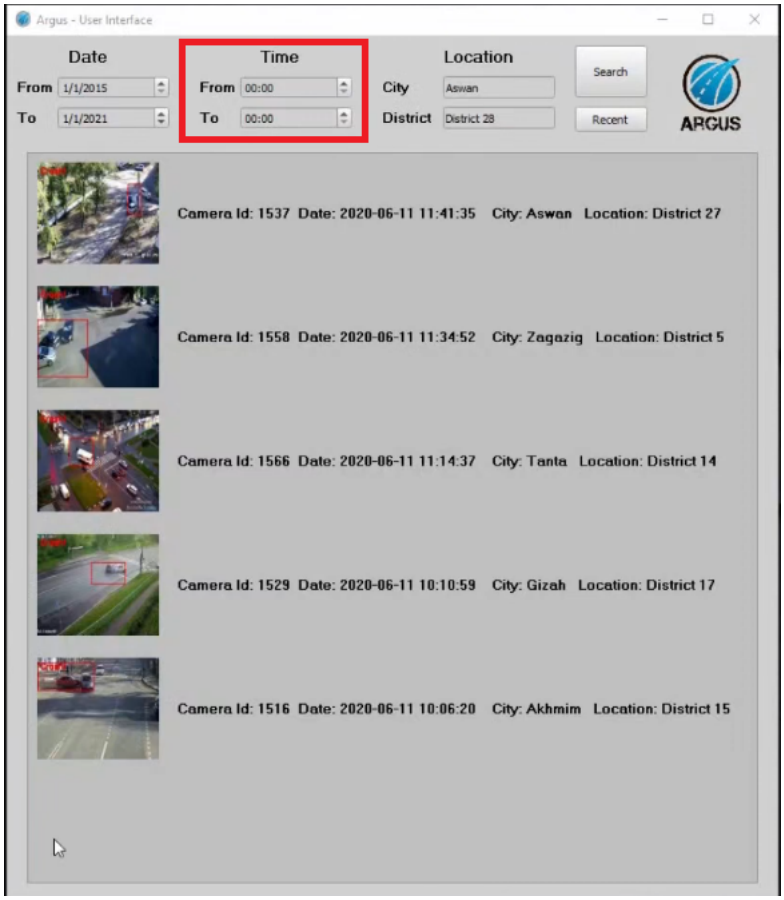

Figure C.4: selecting time interval

4. select the location you want to search

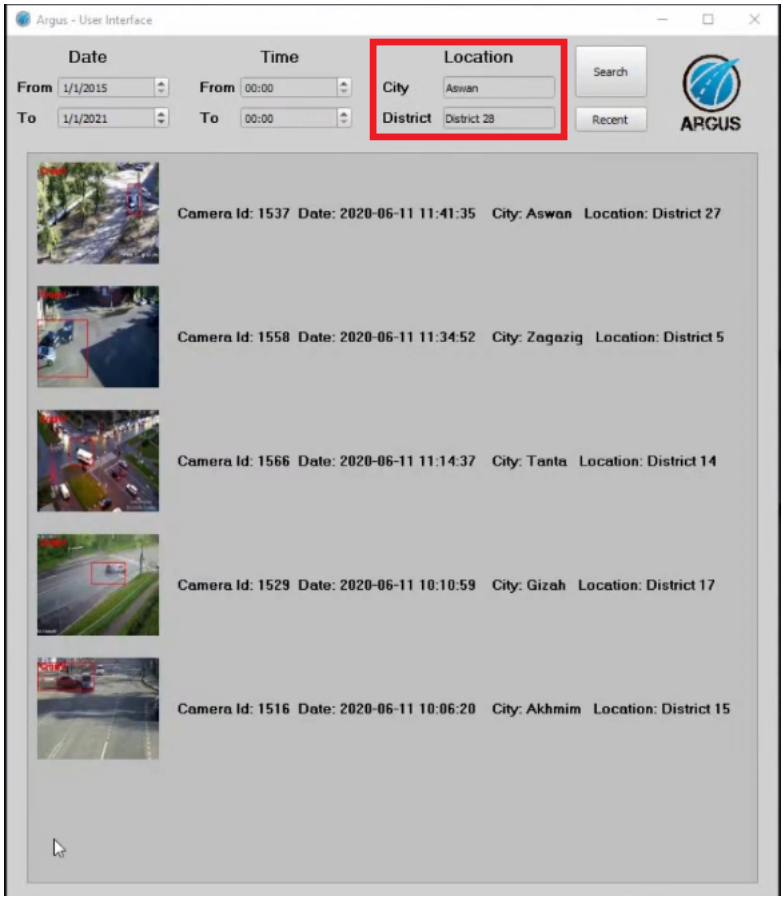

Figure C.5: selecting location

5. press search

Figure C.6: press search

6. Review results
   Results will be displayed in the box. For example, these are the results for searching for accidents that happened in 'Tanta' in the past five years.

Figure C.7: Review results

- Get most recent accidents
  Normally the system displays the most recent ten accidents when it opens up. But after the search for a certain query you might want to go back to default view.

  1. After executing a query, press the recent button to go to the default view

Figure C.8: Get most recent accidents

2. Then the system will display the most recent accidents.

Figure C.9: Get most recent accidents

- Display the video of an accident
  To play the video where the accident is annotated so you can examine it.

  1. Double click on any video

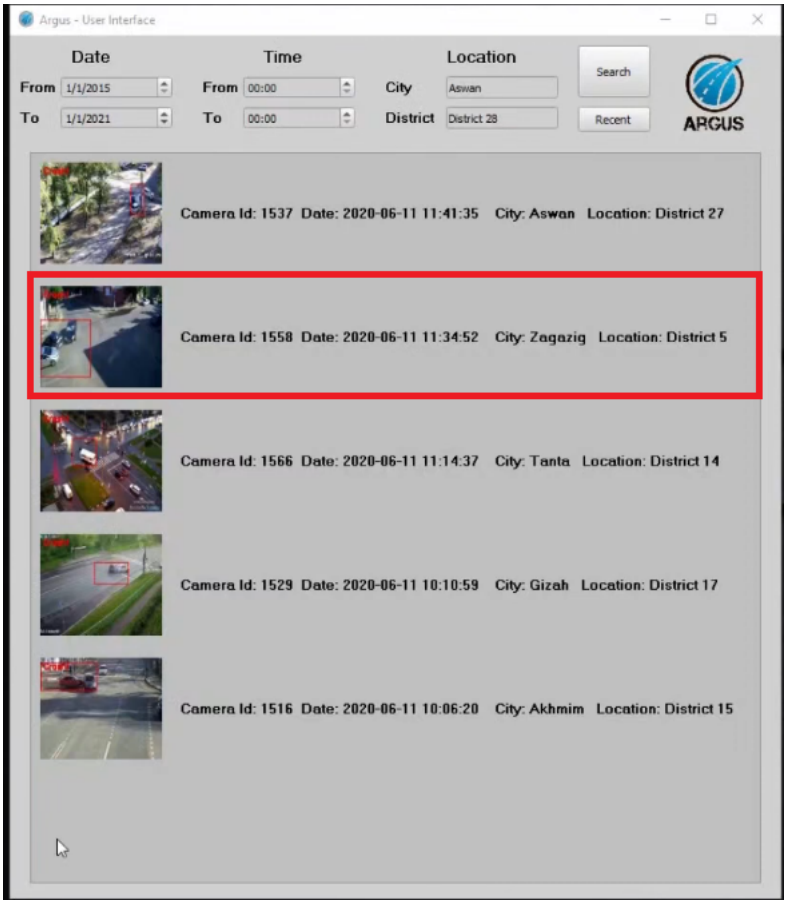

Figure C.10: Display the video of an accident

  2. Then the video should start playing

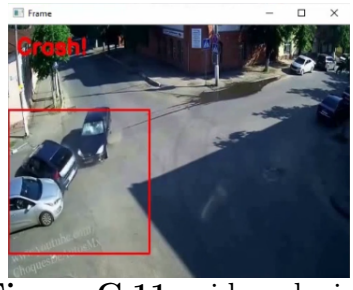

Figure C.11: video playing

- Get notifications about accidents when it happens
  You are required to do nothing. While the system is running, once an accident takes

place within two seconds maximum the program will alert you and the accident will be added to the view highlighted in green.

Figure C.12: Get notifications about accidents

C.2 Surveillance camera simulation interface

This view or form will give you the ability to pass old videos or footage to the server as if you are sending this from a CCTV to check for accidents in it. Also, As we deploy the system you might want to test it offline without using CCTVs.

- Load a video
  You have two options here either drag and drop or use file browser.

  1. press 'select video' button

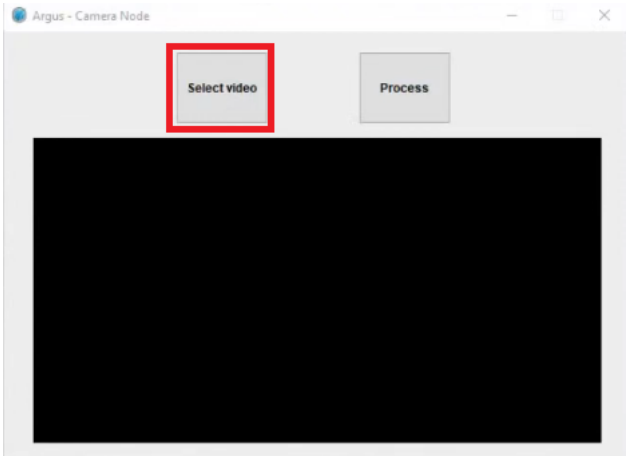

Figure C.13: load video

  2. browse and choose the video you want

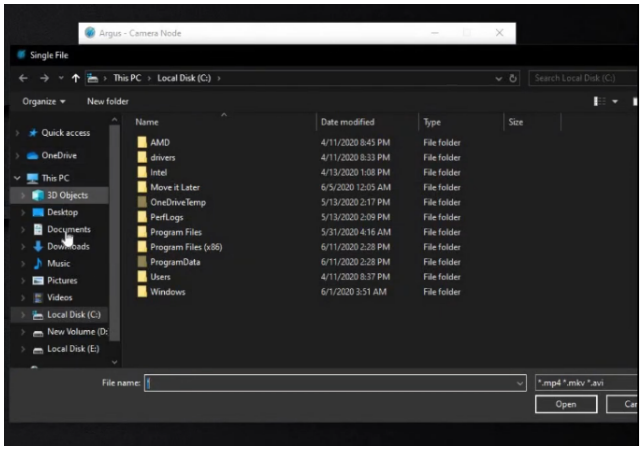

Figure C.14: browse and choose the video you want

  3. Display the video (Optional)
     you can display the video you just loaded by clicking on it. and it start playing anyway once you pressed process

Figure C.15: Display the video

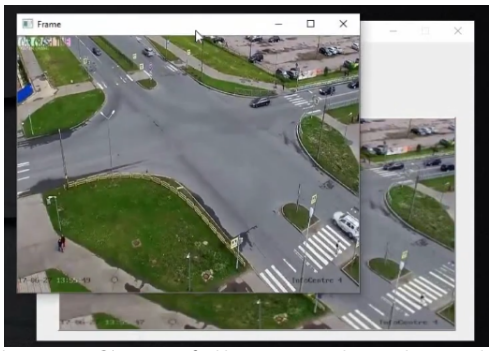

Figure C.16: follow-Display the video

4. Press ‘Process’ button to send it to the server
   Once the ‘Process’ button is pressed the video will start streaming to the server and results will appear on the client view. A gif will appear at the left top corner of the screen to verify that video is streaming to the server.

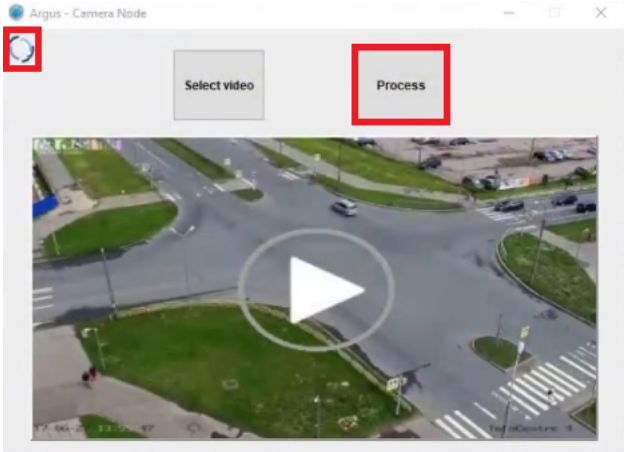

Figure C.17: sending to the server

D. Feasibility Study

Argus is targeting the governments as a target customer which is good because no other application that tackles the same problem is targetting the governments. The only competition is google automatic car crash detection system and SOSmart, both are mobile applications. Mobile applications have advantages, they are cheaper and yield accurate results. But, They have disadvantages also; they depend on other peoples which in many cases led to a big latency gap between the accident and when the authorities are informed. anything could happen to the phone before it can report the accident like running out of battery or being destroyed in the accident before sending the alert. Besides, it requires mobile network availability to be effective. So, in remote places, it will be cheaper to install a camera and use wired communication rather than having to provide network coverage to the whole area. Finally, we need to estimate the price of the product and set marketing strategy; which will be discussed in this section.

D.1 Total cost

In this section we will give a complete estimation to the cost of running the business for a whole year. So that in the next section we can estimate the price of the product.

D.1.1 Establishing the company

On starting the company the following will be needed:

- Company office Most of the company work will be in foregien countries and remote so there is no need for a fancy office at the start of the company. A good office in a workspace will be very sufficient at least for the first year or two. Such an office in a decent workspace is estimated to cost about 1000 EGP a month so for the whole year the office will cost about 12000 EGP.

- Employees The company will have at least 3 call center employees(salary will be about 2500 EGP per employee), 2 senior software engineers with good experience in distributed systems and machine intelligence(salary will be about 16000 EGP per engineer), 4 junior software engineers to fix the system's bugs and update it(salary will be about 8000 EGP per engineer), quality assurance engineer(salary will be about 8000 EGP), Marketing department head(salary will be about 10000 EGP), two marketing department employees(salary will be about 4000 EGP per employee) and human resources manager(salary will be about 5000 EGP). In total employees will cost about 1,200,000 EGP a year.

All the previous numbers are based on estimation from glassdoor.com website.

D.2 Marketing

This is the most crucial department in a starting company because it will promote the company and let people and the government know about the company. Conventional marketing methods are no good in our situation banners and TV commercials will be useless. The main marketing strategy is communicating with Embassies in our country and sending representatives to our company to meet with city governors and ministers. A Round trip for one employee to a forgein country will cost about 30000 EGP (on average 15000 for flight ticket and 15000 lodging and insurance). Say the company will get in touch with four countries. So, in total marketing will cost about 120,000 EGP a year.

D.3 Estimated product price

Considering the costs in the previous section we can estimate a price that is both affordable by governments and profitable to the company at the same time.. So, the cost of running the company based on the previous section will be 1,350,000 EGP. Adding a profit margin to the total cost to divide among the customers company will get a year so in total cost will be 1,500,000 EGP a year. From the business study in section three we have come to the conclusion that the company can serve about ten customers in five years. Therefore, on average the company will serve two customers a year so the product price will be 750,000 EGP.

One thing to consider that the number of customers is limited and main customer service will be the sustainable source of income to the company. So, the company has two options: either charge customers more for free customer service or use annual subscription system.