DETECTION OF ACCIDENTS UNDER LOW LIGHT CCTV MONITORING CONDITIONS IN TUNNELS

Mr. G. Sekhar Reddy[1], K. Rohith Mithra[2], M. Raja Reddy[3], S. Nitya Sree[4]
[1]Assistant Professor, Dept. of Information Technology, Anurag University, Hyderabad, India
[2,3,4]Student, Dept. of Information Technology, Anurag Group of Institutions, Hyderabad, India
golamari.sekhar@gmail.com [1], rohithmithra16@gmail.com[2] , muskurajareddy25@gmail.com[3] , nityareddy1611@gmail.com[4]

Abstract: The goal of this project is to automatically detect and monitor unexpected events on CCTVs in tunnels. These events include (1) Wrong- Way Driving (WWD), (2) Stop, (3)Person out of vehicle in tunnel, and (4) Fire. To accomplish this, we will introduce and apply an Object Detection and Conventional Object Tracking (ODCT) algorithm. Bounding Box (BBox) results from Object Detection are accepted as input by ODTS, and ID numbers are assigned to each moving and detected object by comparing the BBoxes of the current and previous video frames. The system can keep up with a moving target, which is difficult to do with more traditional object detection frameworks. Average Precision (AP) values of 0.8479, 0.7161, and 0.9085 were obtained for target objects of cars, people, and fires, respectively, when a deep learning model in ODTS was trained with a dataset of images in tunnels. Next, four accident videos were used to evaluate the ODTS-based Tunnel CCTV Accident Detection System's trained deep learning model. Consequently, the system is able to identify any incident within 10 seconds. It was accomplished that ODTS's detection capacity could be increased automatically as the training dataset grew in size and complexity without requiring any changes to the code.

Keywords: CNN, Object detection, framework, IMOT

I. INTRODUCTION

The use of object detection technology has led to accurate measurements and locations of specific objects in still images and moving video. Numerous uses have surfaced, most notably in autonomous vehicles, security and surveillance systems, cancer diagnosis, and other fields. Furthermore, in the field of image processing, object tracking can be accomplished through the use of unique identification and tracking the locations of identified objects over time. However, object detection in a single, static image is a prerequisite to tracking those objects. The success of object tracking, then, should rest heavily on the quality of the object detection used. Successful applications of this object tracking technology include monitoring traffic camera footage for accidents and criminal activity, tracing moving pedestrians, and monitoring a specific area for security purposes. The purpose of this paper is to present the results of a case study on the analysis and control of traffic conditions using automatic object detection within the domain of traffic control. Here are the summaries, in the order given. Based on, engineers at Google created a system to monitor traffic around autonomous vehicles. The Convolutional Neural Network in this system is responsible for both vehicle detection and classification (CNN). The vehicle object is followed by adjusting the centre of tracking to correspond with the location of the vehicle in the image, a technique developed by the vehicle object tracking algorithm.

After the system determines the distance between the driving car and the visualised vehicle objects, a localised image, resembling a bird's-eye view, is displayed on the monitor, incorporating the objects' respective vehicle locations. The system's procedure makes it possible to see the vehicle's position from an outside perspective, which aids the self-driving system. It achieves a vertical tolerance of 1.5 m and a horizontal tolerance of 0.4 m at the camera to precisely localise the vehicle's object. In, researchers combined convolutional neural networks (CNNs) and support vector machines (SVMs) to create a detection system that can track moving vehicles on highways and streets using satellite imagery. This system takes the satellite image as an input value, runs it through a CNN to extract features, and then runs a support vector machine (SVM) binary classifier on the features to detect the vehicle BBox. In addition, a system was created by Arinaldi, Pradana, and Gurusinga to calculate an approximation of vehicle speed, categorise vehicles according to type, and assess traffic density.

The BBox information is gleaned from video or image-based object detection in this system. Gaussian Mixture Model SVM and quicker RCNN were used to evaluate the algorithm used in the system. Then, it seems that R-CNN with a higher processing speed accurately identified the location and type of vehicle. Simply put, deep learning-based object detection is a more effective method than algorithm-based object detection. All of the development scenarios presented in this paper are focused on object data, and they all achieve remarkable results thanks to deep learning. It was challenging for any of them to give the detected objects permanent identifiers and continue tracking them with the same identifier as time went on.

As a result, an effort is made to design an object detection and tracking system in this paper (ODTS), that, by combining an object tracking algorithm with a deep learning-based object detection process, can obtain data on the motion of target objects. In the following paragraphs, we'll provide a comprehensive breakdown of the ODTS process. As part of ODTS, we will also consider a system to identify tunnel accidents. This system monitors CCTV footage in a specific area and tracks moving objects to identify any unforeseen incidents.

A lightweight version of a tracking-by-detection framework for MOT, where objects are detected at each frame and their bounding boxes are used as tracking targets. This work, in contrast to many batch-based tracking approaches, is primarily geared towards online tracking, where the tracker is only shown detections from the immediately preceding and following frames. Real-time tracking is facilitated, and greater adoption of applications like pedestrian tracking for autonomous vehicles is encouraged, with a focus on efficiency. The goal of the MOT problem is to link detections in different frames of a video sequence, so it can be thought of as a data association problem. Trackers use a wide variety of techniques to model the behaviour and appearance of objects in the scene to aid the data association process.

This paper's methods were inspired by data from a recently established visual MOT benchmark. First, many of the best-performing methods on the MOT benchmark—like Multiple Hypothesis Tracking (MHT) and Joint Probabilistic Data Association (JPDA)—are making a comeback. Second, the best-performing tracker is the only one that doesn't employ the Aggregate Channel Filter (ACF) detector, implying that the other trackers' performance may be limited by poor detection quality. Most effective trackers are also too slow for use in real-time situations, suggesting a clear trade-off between the two qualities.

II. BACKGROUND

E. S. Lee et al. The ability to detect vehicles on the road is crucial for understanding driving environments, and pinpointing the location of a detected vehicle can aid in hazard assessment and collision avoidance. However, there are few works on partially visible vehicle detection, and the method for localising partially visible vehicles has not been investigated. This paper proposes a new framework for detecting and localising vehicles with a partial appearance using stereo vision and geometry. A v-disparity map is created by first processing the raw data from the stereo camera. With the help of v-disparity, we can first identify objects in an image, and then use that information to generate candidates for vehicles based on our understanding of where they might be. Validation via deep learning-based methods finishes off vehicle detection.

L. Cao et al. Vehicle detection in satellite images is a new area of study with potential military and commercial uses. However, it remains an unsolved problem, primarily because of the high degree of complexity introduced by the wide range of imaging conditions, the dynamic nature of objects within classes, and the limited resolution of current methods. In light of the recent breakthroughs in deep learning for feature representation, this paper explores the potential for using deep neural features for accurate vehicle detection. In addition, new classification methodology is required to explicitly deal with the intra-class variations, which has become necessary due to the exponential increase in the volume of data.

Arinaldi et al. In this article, we introduce a computer-vision-based system for analysing traffic videos. The system is meant to mechanically collect crucial data for regulators and policymakers. Vehicle counts, vehicle types categorised, estimated speeds derived from video, and lane usage monitoring are all examples of such data. Vehicle recognition and classification in traffic videos constitute the heart of such a system. Both the MoG + SVM system and Faster RCNN, a recently popular deep learning architecture for object detection in images, are implemented as models for this task. We show experimentally that Faster RCNN outperforms MoG in detecting vehicles that are stationary, overlapping, or operating in low light. When comparing performance for the task of visually classifying vehicle types, the faster RCNN also trumps SVM.

III. EXISTING SYSTEM

Vehicles are currently identified and categorised using a Convolutional Neural Network in the current system (CNN). The vehicle object is followed by adjusting the centre of tracking to correspond with the location of the vehicle in the image, a technique developed by the vehicle object tracking algorithm. After the system determines the distance between the driving car and the visualised vehicle objects, a localised image, resembling a bird's-eye view, is displayed on the monitor, incorporating the objects' respective vehicle locations.

IV. PROPOSED SYSTEM

In the system that we have proposed, an effort has been made to generate an object detection and tracking system (ODTS) with yolo that is able to obtain information about the motion of target objects along with their names. This is accomplished by combining an object tracking algorithm with a process that is based on deep learning for object detection. It is taken for granted that ODTS has received sufficient training to carry out object detection on a given image frame in an accurate manner. ODTS acquires new sets of coordinates whenever it gets a certain number of frames of video at a particular time interval c. This results in the detection of n BBoxes. BBoxT of things that were detected by the trained object detection system on the specified image frame at the given time T. The object detection module will simultaneously categorise each detected object according to its associated kind or class, denoted by the notation BBoxT.

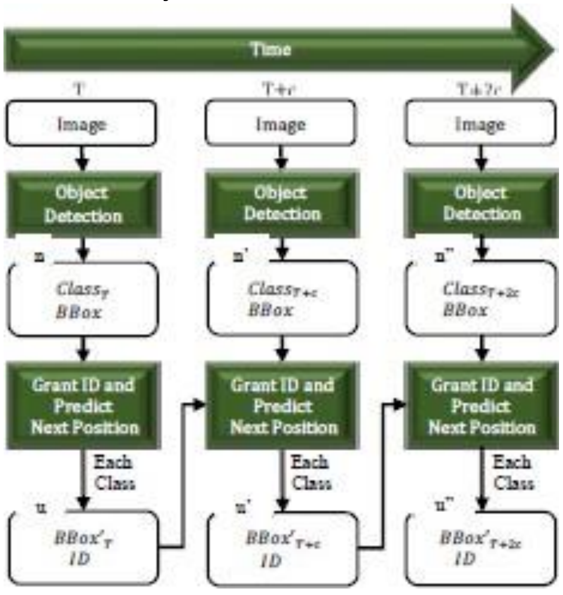
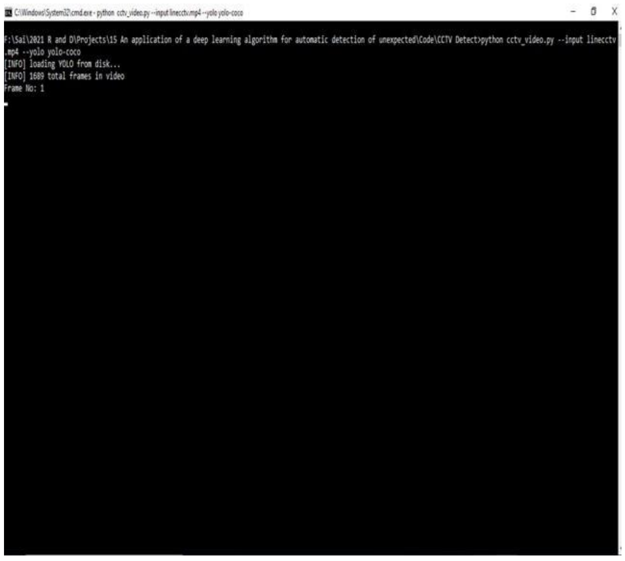


Fig 1: Faster R-CNN (Regional Convolution Neural Network), YOLO model

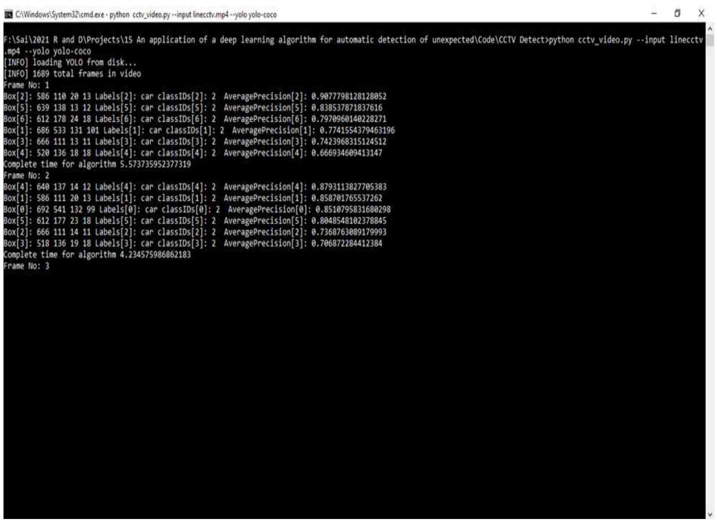
V. RESULTS

An app alerts authorities of noisy tunnel issues. This method will capture tunnel footage and feed it to our system. CCTV and regular cameras record at 25 frames per second, so the video will too. After achieving the frame rate, it is split up to obtain the frame rate for 1 second. The lighting must be adjusted to extract the colour feature. The input image is 800*800. After extracting the colour feature and modifying the lighting, the frame will be presented to a fire detection classifier. We must change our colour. RGB images have yellow fire. Fire detection is difficult if a yellow cloth or car is present. To identify fire, we'll use YCbCr colour modification to extract the Mean, Median, and Standard Deviation.



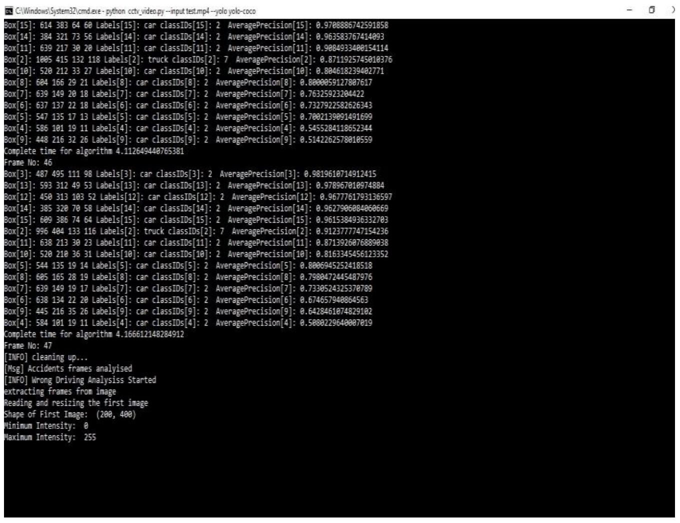
```
C:\Windows\System32\cmd.exe - python cctv_video.py --input linecctv.mp4 --yolo yolo-coco

F:\Sai\2021 R and D\Projects\15 An application of a deep learning algorithm for automatic detection of unexpected\Code\CCTV Detect>python cctv_video.py --input linecctv
.mp4 --yolo yolo-coco
[INFO] loading YOLO from disk...
[INFO] 1689 total frames in video
Frame No: 1
```

Fig 2: Loading Training Model



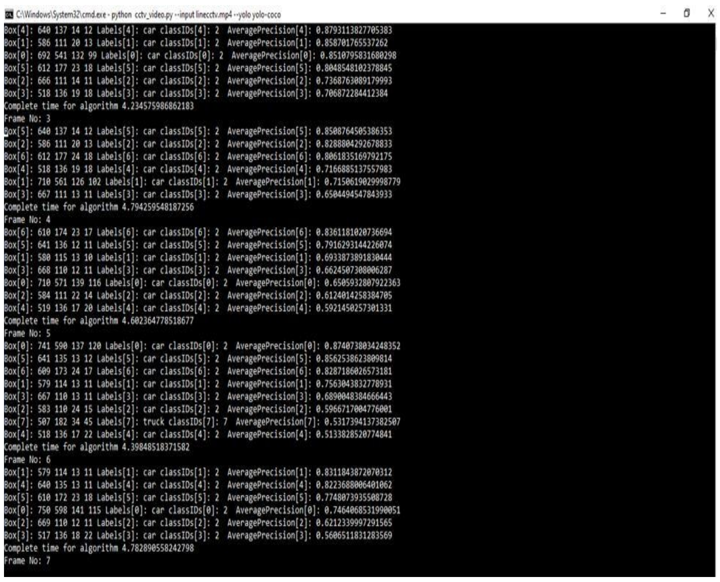
```
C:\Windows\System32\cmd.exe - python cctv_video.py --input linecctv.mp4 --yolo yolo-coco
F:\Sai\2021 R and D\Projects\15 An application of a deep learning algorithm for automatic detection of unexpected\Code\CCTV Detect>python cctv_video.py --input linecctv
.mp4 --yolo yolo-coco
[INFO] loading YOLO from disk...
[INFO] 1689 total frames in video
Frame No: 1
Box[2]: 586 110 20 13 Labels[2]: car classIDs[2]: 2  AveragePrecision[2]: 0.9077798128128052
Box[5]: 639 138 13 12 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.838537871837616
Box[6]: 612 178 24 18 Labels[6]: car classIDs[6]: 2  AveragePrecision[6]: 0.7970960140228271
Box[1]: 686 533 131 101 Labels[1]: car classIDs[1]: 2  AveragePrecision[1]: 0.7741554379463196
Box[3]: 666 111 13 11 Labels[3]: car classIDs[3]: 2  AveragePrecision[3]: 0.7423968315124512
Box[4]: 520 136 18 18 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.666934609413147
Complete time for algorithm 5.573735952377319
Frame No: 2
Box[4]: 640 137 14 12 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.8793113827705383
Box[1]: 586 111 20 13 Labels[1]: car classIDs[1]: 2  AveragePrecision[1]: 0.858701765537262
Box[0]: 692 541 132 99 Labels[0]: car classIDs[0]: 2  AveragePrecision[0]: 0.8510795831680298
Box[5]: 612 177 23 18 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.8048548102378845
Box[2]: 666 111 14 11 Labels[2]: car classIDs[2]: 2  AveragePrecision[2]: 0.7368763089179993
Box[3]: 518 136 19 18 Labels[3]: car classIDs[3]: 2  AveragePrecision[3]: 0.706872284412384
Complete time for algorithm 4.234575986862183
Frame No: 3
```

Fig 3: Video Separated as Frames



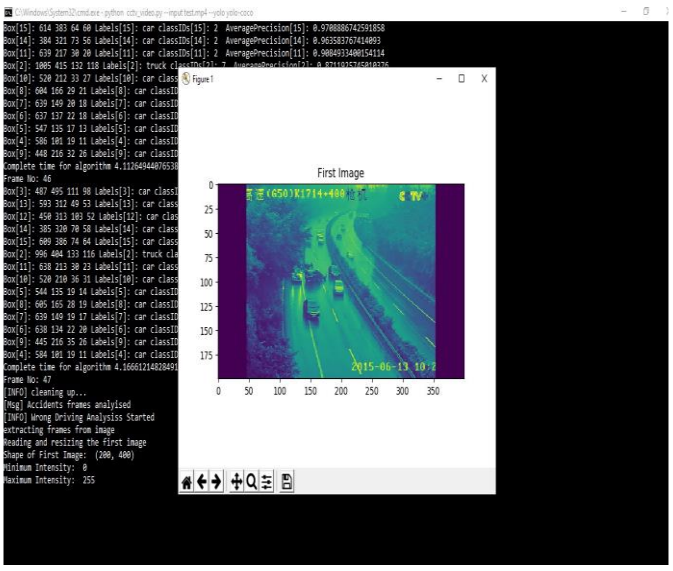
```
C:\Windows\System32\cmd.exe - python cctv_video.py --input test.mp4 --yolo yolo-coco
Box[15]: 614 383 64 60 Labels[15]: car classIDs[15]: 2  AveragePrecision[15]: 0.9708886742591858
Box[14]: 384 321 73 56 Labels[14]: car classIDs[14]: 2  AveragePrecision[14]: 0.963583767414093
Box[11]: 639 217 30 20 Labels[11]: car classIDs[11]: 2  AveragePrecision[11]: 0.9084933400154114
Box[2]: 1005 415 132 118 Labels[2]: truck classIDs[2]: 7  AveragePrecision[2]: 0.8711925745010376
Box[10]: 520 212 33 27 Labels[10]: car classIDs[10]: 2  AveragePrecision[10]: 0.804618239402771
Box[8]: 604 166 29 21 Labels[8]: car classIDs[8]: 2  AveragePrecision[8]: 0.8000059127807617
Box[7]: 639 149 20 18 Labels[7]: car classIDs[7]: 2  AveragePrecision[7]: 0.76325923204422
Box[6]: 637 137 22 18 Labels[6]: car classIDs[6]: 2  AveragePrecision[6]: 0.7327922582626343
Box[5]: 547 135 17 13 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.7002139091491699
Box[4]: 586 101 19 11 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.5455284118652344
Box[9]: 448 216 32 26 Labels[9]: car classIDs[9]: 2  AveragePrecision[9]: 0.5142262578010559
Complete time for algorithm 4.112649440765381
Frame No: 46
Box[3]: 487 495 111 98 Labels[3]: car classIDs[3]: 2  AveragePrecision[3]: 0.9819610714912415
Box[13]: 593 312 49 53 Labels[13]: car classIDs[13]: 2  AveragePrecision[13]: 0.978967010974884
Box[12]: 450 313 103 52 Labels[12]: car classIDs[12]: 2  AveragePrecision[12]: 0.9677761793136597
Box[14]: 385 320 70 58 Labels[14]: car classIDs[14]: 2  AveragePrecision[14]: 0.9627906084060669
Box[15]: 609 386 74 64 Labels[15]: car classIDs[15]: 2  AveragePrecision[15]: 0.9615384936332703
Box[2]: 996 404 133 116 Labels[2]: truck classIDs[2]: 7  AveragePrecision[2]: 0.9123777747154236
Box[11]: 638 213 30 23 Labels[11]: car classIDs[11]: 2  AveragePrecision[11]: 0.8713926076889038
Box[10]: 520 210 36 31 Labels[10]: car classIDs[10]: 2  AveragePrecision[10]: 0.8163345456123352
Box[5]: 544 135 19 14 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.8006945252418518
Box[8]: 605 165 28 19 Labels[8]: car classIDs[8]: 2  AveragePrecision[8]: 0.7980472445487976
Box[7]: 639 149 19 17 Labels[7]: car classIDs[7]: 2  AveragePrecision[7]: 0.7330524325370789
Box[6]: 638 134 22 20 Labels[6]: car classIDs[6]: 2  AveragePrecision[6]: 0.674657940864563
Box[9]: 445 216 35 26 Labels[9]: car classIDs[9]: 2  AveragePrecision[9]: 0.6428461074829102
Box[4]: 584 101 19 11 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.5080229640007019
Complete time for algorithm 4.166612148284912
Frame No: 47
[INFO] cleaning up...
[Msg] Accidents frames analyised
[INFO] Wrong Driving Analysiss Started
extracting frames from image
Reading and resizing the first image
Shape of First Image:  (200, 400)
Minimum Intensity:  0
Maximum Intensity:  255
```

Fig 6: Preprocess done



```
C:\Windows\System32\cmd.exe - python cctv_video.py --input linecctv.mp4 --yolo yolo-coco
Box[4]: 640 137 14 12 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.8793113827705383
Box[1]: 586 111 20 13 Labels[1]: car classIDs[1]: 2  AveragePrecision[1]: 0.858701765537262
Box[0]: 692 541 132 99 Labels[0]: car classIDs[0]: 2  AveragePrecision[0]: 0.8510795831680298
Box[5]: 612 177 23 18 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.8048548102378845
Box[2]: 666 111 14 11 Labels[2]: car classIDs[2]: 2  AveragePrecision[2]: 0.7368763089179993
Box[3]: 518 136 19 18 Labels[3]: car classIDs[3]: 2  AveragePrecision[3]: 0.706872284412384
Complete time for algorithm 4.234575986862183
Frame No: 3
Box[5]: 640 137 14 12 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.8508764505386353
Box[2]: 586 111 20 13 Labels[2]: car classIDs[2]: 2  AveragePrecision[2]: 0.8288804292678833
Box[6]: 612 177 24 18 Labels[6]: car classIDs[6]: 2  AveragePrecision[6]: 0.8061835169792175
Box[4]: 518 136 19 18 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.7166885137557983
Box[1]: 710 561 126 102 Labels[1]: car classIDs[1]: 2  AveragePrecision[1]: 0.7150619029998779
Box[3]: 667 111 13 11 Labels[3]: car classIDs[3]: 2  AveragePrecision[3]: 0.6504494547843933
Complete time for algorithm 4.794259548187256
Frame No: 4
Box[6]: 610 174 23 17 Labels[6]: car classIDs[6]: 2  AveragePrecision[6]: 0.8361181020736694
Box[5]: 641 136 12 11 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.7916293144226074
Box[1]: 580 115 13 10 Labels[1]: car classIDs[1]: 2  AveragePrecision[1]: 0.6933873891830444
Box[3]: 668 110 12 11 Labels[3]: car classIDs[3]: 2  AveragePrecision[3]: 0.6624507308006287
Box[0]: 710 571 139 116 Labels[0]: car classIDs[0]: 2  AveragePrecision[0]: 0.6505932807922363
Box[2]: 584 111 22 14 Labels[2]: car classIDs[2]: 2  AveragePrecision[2]: 0.6124014258384705
Box[4]: 519 136 17 20 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.5921450257301331
Complete time for algorithm 4.602364778518677
Frame No: 5
Box[0]: 741 590 137 120 Labels[0]: car classIDs[0]: 2  AveragePrecision[0]: 0.8740738034248352
Box[5]: 641 135 13 12 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.8562538623809814
Box[6]: 609 173 24 17 Labels[6]: car classIDs[6]: 2  AveragePrecision[6]: 0.8287186026573181
Box[1]: 579 114 13 11 Labels[1]: car classIDs[1]: 2  AveragePrecision[1]: 0.7563043832778931
Box[3]: 667 110 13 11 Labels[3]: car classIDs[3]: 2  AveragePrecision[3]: 0.6890048384666443
Box[2]: 583 110 24 15 Labels[2]: car classIDs[2]: 2  AveragePrecision[2]: 0.5966717004776001
Box[7]: 507 182 34 45 Labels[7]: truck classIDs[7]: 7  AveragePrecision[7]: 0.5317394137382507
Box[4]: 518 136 17 22 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.5133828520774841
Complete time for algorithm 4.39848518371582
Frame No: 6
Box[1]: 579 114 13 11 Labels[1]: car classIDs[1]: 2  AveragePrecision[1]: 0.8311843872070312
Box[4]: 640 135 13 11 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.8223688006401062
Box[5]: 610 172 23 18 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.7748073935508728
Box[0]: 750 598 141 115 Labels[0]: car classIDs[0]: 2  AveragePrecision[0]: 0.7464068531990051
Box[2]: 669 110 12 11 Labels[2]: car classIDs[2]: 2  AveragePrecision[2]: 0.6212339997291565
Box[3]: 517 136 18 22 Labels[3]: car classIDs[3]: 2  AveragePrecision[3]: 0.5606511831283569
Complete time for algorithm 4.782890558242798
Frame No: 7
```

Fig 4: Analyzing Each Frame



```
C:\Windows\System32\cmd.exe - python cctv_video.py --input test.mp4 --yolo yolo-coco
Box[15]: 614 383 64 60 Labels[15]: car classIDs[15]: 2  AveragePrecision[15]: 0.9708886742591858
Box[14]: 384 321 73 56 Labels[14]: car classIDs[14]: 2  AveragePrecision[14]: 0.963583767414093
Box[11]: 639 217 30 20 Labels[11]: car classIDs[11]: 2  AveragePrecision[11]: 0.9084933400154114
Box[2]: 1005 415 132 118 Labels[2]: truck classIDs[2]: 7  AveragePrecision[2]: 0.8711925745010376
Box[10]: 520 212 33 27 Labels[10]: car class
Box[8]: 604 166 29 21 Labels[8]: car classID
Box[7]: 639 149 20 18 Labels[7]: car classID
Box[6]: 637 137 22 18 Labels[6]: car classID
Box[5]: 547 135 17 13 Labels[5]: car classID
Box[4]: 586 101 19 11 Labels[4]: car classID
Box[9]: 448 216 32 26 Labels[9]: car classID
Complete time for algorithm 4.11264944076538
Frame No: 46
Box[3]: 487 495 111 98 Labels[3]: car classI
Box[13]: 593 312 49 53 Labels[13]: car class
Box[12]: 450 313 103 52 Labels[12]: car clas
Box[14]: 385 320 70 58 Labels[14]: car class
Box[15]: 609 386 74 64 Labels[15]: car class
Box[2]: 996 404 133 116 Labels[2]: truck cla
Box[11]: 638 213 30 23 Labels[11]: car class
Box[10]: 520 210 36 31 Labels[10]: car class
Box[5]: 544 135 19 14 Labels[5]: car classID
Box[8]: 605 165 28 19 Labels[8]: car classID
Box[7]: 639 149 19 17 Labels[7]: car classID
Box[6]: 638 134 22 20 Labels[6]: car classID
Box[9]: 445 216 35 26 Labels[9]: car classID
Box[4]: 584 101 19 11 Labels[4]: car classID
Complete time for algorithm 4.16661214828491
Frame No: 47
[INFO] cleaning up...
[Msg] Accidents frames analyised
[INFO] Wrong Driving Analysiss Started
extracting frames from image
Reading and resizing the first image
Shape of First Image:  (200, 400)
Minimum Intensity:  0
Maximum Intensity:  255
```

Fig 7: First Image Compare

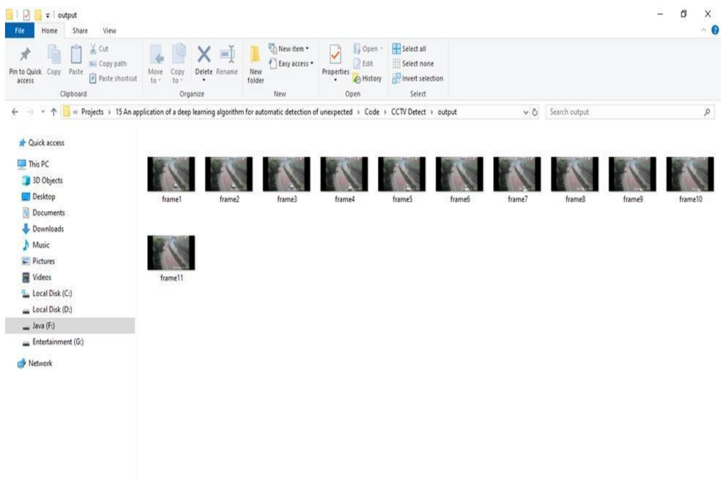
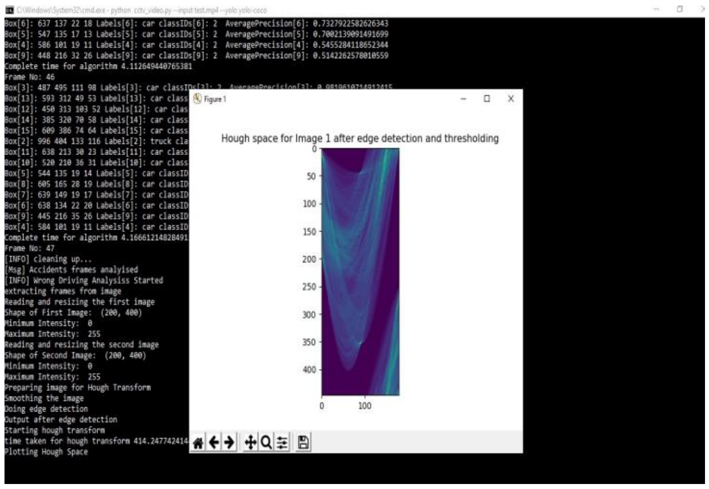


Fig 5: Output Frames


C:\Windows\System32\cmd.exe - python cctv_video.py --input test.mp4 --yolo yolo-coco
Figure 1
Hough space for Image 1 after edge detection and thresholding

```
Box[6]: 637 137 22 18 Labels[6]: car classIDs[6]: 2  AveragePrecision[6]: 0.7327922582626343
Box[5]: 547 135 17 13 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.7002139091491699
Box[4]: 586 101 19 11 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.5455284118652344
Box[9]: 448 216 32 26 Labels[9]: car classIDs[9]: 2  AveragePrecision[9]: 0.5142262578010559
Complete time for algorithm 4.112649440765381
Frame No: 46
Box[3]: 487 495 111 98 Labels[3]: car classIDs[3]: 2  AveragePrecision[3]: 0.98196107149124
Box[13]: 593 312 49 53 Labels[13]: car class
Box[12]: 450 313 103 52 Labels[12]: car clas
Box[14]: 385 320 70 58 Labels[14]: car class
Box[15]: 609 386 74 64 Labels[15]: car class
Box[2]: 996 404 133 116 Labels[2]: truck cla
Box[11]: 638 213 30 23 Labels[11]: car class
Box[10]: 520 210 36 31 Labels[10]: car class
Box[5]: 544 135 19 14 Labels[5]: car classID
Box[8]: 605 165 28 19 Labels[8]: car classID
Box[7]: 639 149 19 17 Labels[7]: car classID
Box[6]: 638 134 22 20 Labels[6]: car classID
Box[9]: 445 216 35 26 Labels[9]: car classID
Box[4]: 584 101 19 11 Labels[4]: car classID
Complete time for algorithm 4.16661214828491
Frame No: 47
[INFO] cleaning up...
[Msg] Accidents frames analyised
[INFO] Wrong Driving Analysiss Started
extracting frames from image
Reading and resizing the first image
Shape of First Image:  (200, 400)
Minimum Intensity:  0
Maximum Intensity:  255
Reading and resizing the second image
Shape of Second Image:  (200, 400)
Minimum Intensity:  0
Maximum Intensity:  255
Preparing image for Hough Transform
Smoothing the image
Doing edge detection
Output after edge detection
Starting hough transform
time taken for hough transform 414.247742414
Plotting Hough Space
```

Fig 8: Second Image Compare

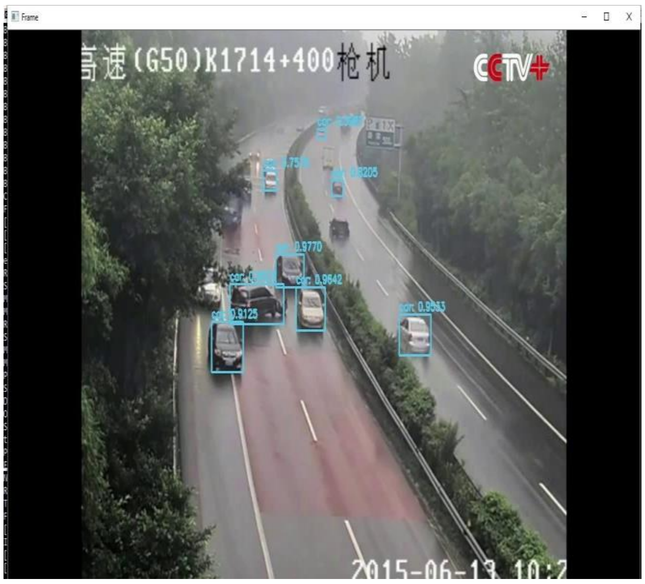
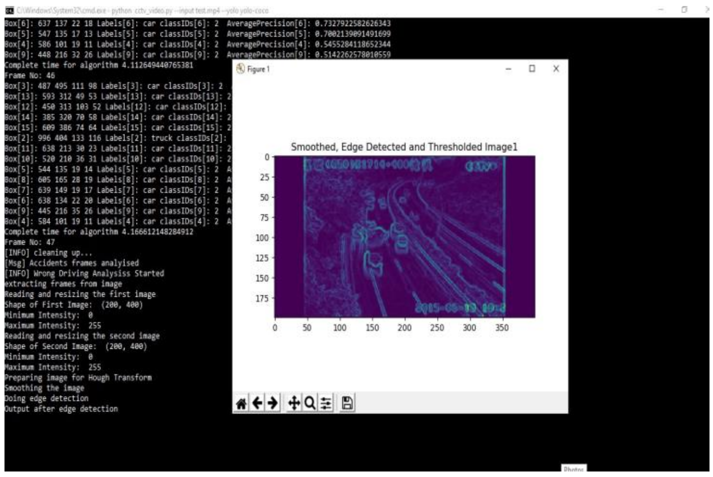


Fig 11: Making Video


C:\Windows\System32\cmd.exe - python cctv_video.py --input test.mp4 --yolo yolo-coco
Figure 1
Smoothed, Edge Detected and Thresholded Image1

```
Box[6]: 637 137 22 18 Labels[6]: car classIDs[6]: 2  AveragePrecision[6]: 0.7327922582626343
Box[5]: 547 135 17 13 Labels[5]: car classIDs[5]: 2  AveragePrecision[5]: 0.7002139091491699
Box[4]: 586 101 19 11 Labels[4]: car classIDs[4]: 2  AveragePrecision[4]: 0.5455284118652344
Box[9]: 448 216 32 26 Labels[9]: car classIDs[9]: 2  AveragePrecision[9]: 0.5142262578010559
Complete time for algorithm 4.112649440765381
Frame No: 46
Box[3]: 487 495 111 98 Labels[3]: car classIDs[3]: 2
Box[13]: 593 312 49 53 Labels[13]: car classIDs[13]: 2
Box[12]: 450 313 103 52 Labels[12]: car classIDs[12]:
Box[14]: 385 320 70 58 Labels[14]: car classIDs[14]: 2
Box[15]: 609 386 74 64 Labels[15]: car classIDs[15]: 2
Box[2]: 996 404 133 116 Labels[2]: truck classIDs[2]:
Box[11]: 638 213 30 23 Labels[11]: car classIDs[11]: 2
Box[10]: 520 210 36 31 Labels[10]: car classIDs[10]: 2
Box[5]: 544 135 19 14 Labels[5]: car classIDs[5]: 2  A
Box[8]: 605 165 28 19 Labels[8]: car classIDs[8]: 2  A
Box[7]: 639 149 19 17 Labels[7]: car classIDs[7]: 2  A
Box[6]: 638 134 22 20 Labels[6]: car classIDs[6]: 2  A
Box[9]: 445 216 35 26 Labels[9]: car classIDs[9]: 2  A
Box[4]: 584 101 19 11 Labels[4]: car classIDs[4]: 2  A
Complete time for algorithm 4.166612148284912
Frame No: 47
[INFO] cleaning up...
[Msg] Accidents frames analyised
[INFO] Wrong Driving Analysiss Started
extracting frames from image
Reading and resizing the first image
Shape of First Image:  (200, 400)
Minimum Intensity:  0
Maximum Intensity:  255
Reading and resizing the second image
Shape of Second Image:  (200, 400)
Minimum Intensity:  0
Maximum Intensity:  255
Preparing image for Hough Transform
Smoothing the image
Doing edge detection
Output after edge detection
```

Fig 9: Space Count

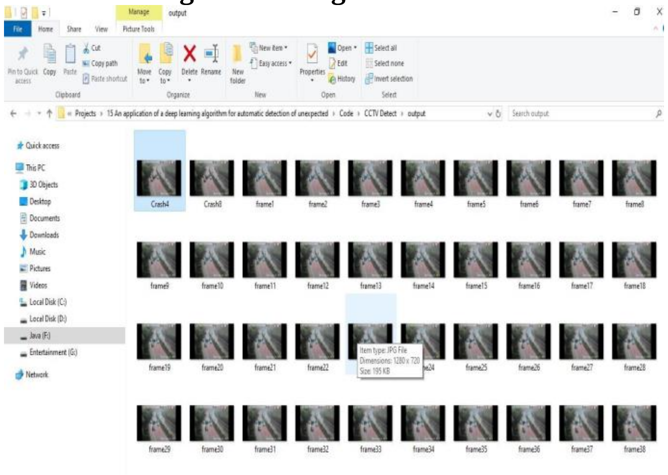
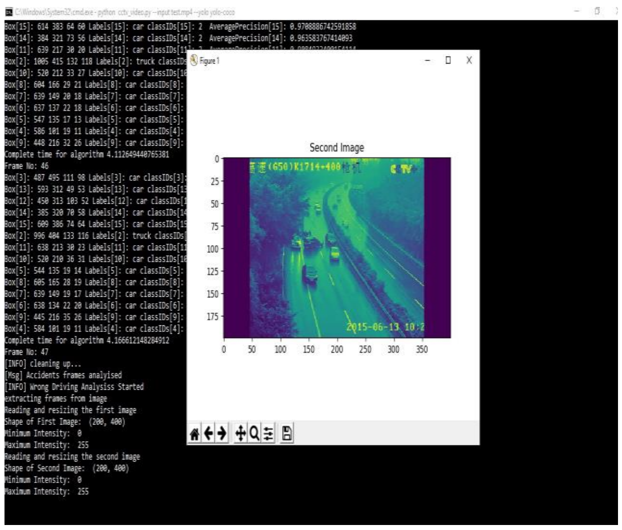
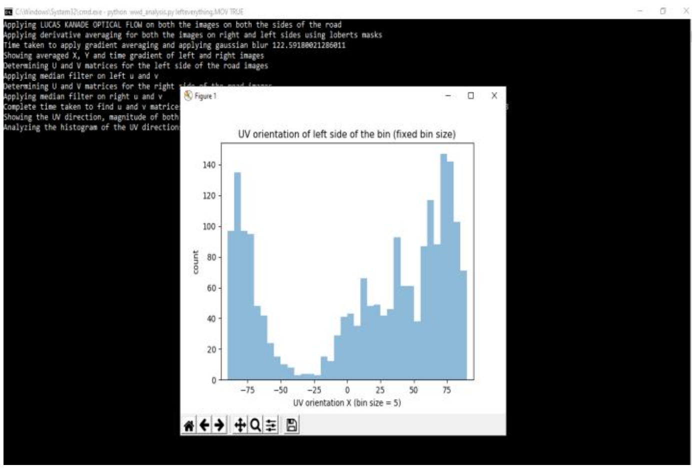


Fig 12: Crash Detected


C:\Windows\System32\cmd.exe - python cctv_video.py --input test.mp4 --yolo yolo-coco
Figure 1
Second Image

```
Box[15]: 614 383 64 60 Labels[15]: car classIDs[15]: 2  AveragePrecision[15]: 0.9708886742591858
Box[14]: 384 321 73 56 Labels[14]: car classIDs[14]: 2  AveragePrecision[14]: 0.963583767414093
Box[11]: 639 217 30 20 Labels[11]: car classIDs[1
Box[2]: 1005 415 132 118 Labels[2]: truck classIDs
Box[10]: 520 212 33 27 Labels[10]: car classIDs[10
Box[8]: 604 166 29 21 Labels[8]: car classIDs[8]:
Box[7]: 639 149 20 18 Labels[7]: car classIDs[7]:
Box[6]: 637 137 22 18 Labels[6]: car classIDs[6]:
Box[5]: 547 135 17 13 Labels[5]: car classIDs[5]:
Box[4]: 586 101 19 11 Labels[4]: car classIDs[4]:
Box[9]: 448 216 32 26 Labels[9]: car classIDs[9]:
Complete time for algorithm 4.112649440765381
Frame No: 46
Box[3]: 487 495 111 98 Labels[3]: car classIDs[3]:
Box[13]: 593 312 49 53 Labels[13]: car classIDs[13
Box[12]: 450 313 103 52 Labels[12]: car classIDs[1
Box[14]: 385 320 70 58 Labels[14]: car classIDs[14
Box[15]: 609 386 74 64 Labels[15]: car classIDs[15
Box[2]: 996 404 133 116 Labels[2]: truck classIDs[
Box[11]: 638 213 30 23 Labels[11]: car classIDs[11
Box[10]: 520 210 36 31 Labels[10]: car classIDs[10
Box[5]: 544 135 19 14 Labels[5]: car classIDs[5]:
Box[8]: 605 165 28 19 Labels[8]: car classIDs[8]:
Box[7]: 639 149 19 17 Labels[7]: car classIDs[7]:
Box[6]: 638 134 22 20 Labels[6]: car classIDs[6]:
Box[9]: 445 216 35 26 Labels[9]: car classIDs[9]:
Box[4]: 584 101 19 11 Labels[4]: car classIDs[4]:
Complete time for algorithm 4.166612148284912
Frame No: 47
[INFO] cleaning up...
[Msg] Accidents frames analyised
[INFO] Wrong Driving Analysiss Started
extracting frames from image
Reading and resizing the first image
Shape of First Image:  (200, 400)
Minimum Intensity:  0
Maximum Intensity:  255
Reading and resizing the second image
Shape of Second Image:  (200, 400)
Minimum Intensity:  0
Maximum Intensity:  255
```

Fig 10: Smooth Detection



```
C:\Windows\System32\cmd.exe - python wwd_analysis.py lefteverything.MOV TRUE
Applying LUCAS KANADE OPTICAL FLOW on both the images on both the sides of the road
Applying derivative averaging for both the images on right and left sides using loberts masks
Time taken to apply gradient averaging and applying gaussian blur 122.59180021286011
Showing averaged X, Y and time gradient of left and right images
Determining U and V matrices for the left side of the road images
Applying median filter on left u and v
Determining U and V matrices for the right
Applying median filter on right u and v
Complete time taken to find u and v matrice
Showing the UV direction, magnitude of both
Analyzing the histogram of the UV direction
```

Figure 1
UV orientation of left side of the bin (fixed bin size)
count
0
20
40
60
80
100
120
140
−75
−50
−25
0
25
50
75
UV orientation X (bin size = 5)

Fig 13: UV orientation

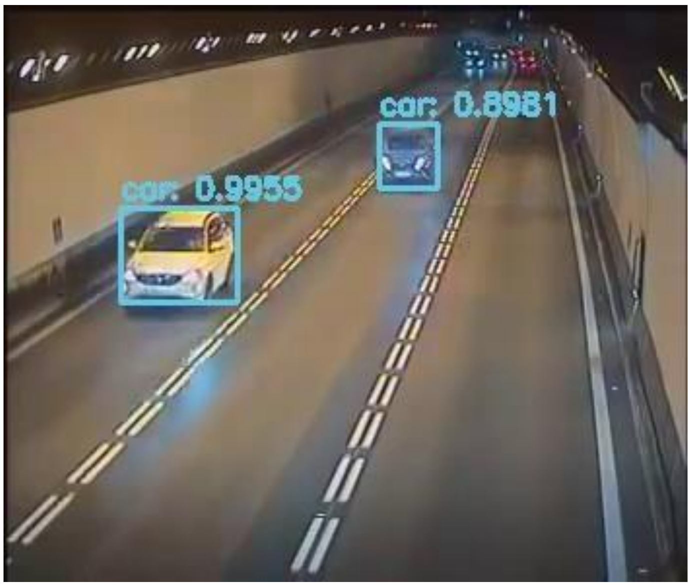


Fig 14: Object & Crash Detection (Tunnel)

Table 1: Test Cases

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

VI. Conclusion

This research presents a new ODTS process using deep learning-based object detection network and object tracking algorithm to acquire and use dynamic information for a certain object class. ODTS object tracking employs SORT, which uses BBox information without an image, hence object identification performance is crucial. Unless the object tracking algorithm relies on object identification performance, constant object detection performance may not be necessary. ODTS-based Tunnel CCTV Accident Detection System was created. Deep learning object detection network training and system accident detection were tested. CADA distinguishes every cycle based on automobile object dynamics in this system. Experimenting with each accident photograph allowed detection in 10 seconds. Nevertheless, deep learning training improved Vehicle object detection and decreased Human object detection. Due to a lack of Fire objects, untrained films have a significant false detection rate. Yet, teaching No Fire objects reduces erroneous detections.

Further Enhancement

Securing the Fire picture should improve the deep learning object detection network's fire object identification performance. The ODTS can be utilised as a Tunnel CCTV Accident Detection System or to estimate vehicle speed or monitor illegal parking. Securing photos and fire and person objects improves system reliability. Moreover, applying and monitoring the tunnel management site could increase system reliability.

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

[13] Y. Min and J. Yunde, “Temporal Dynamic Appearance Modeling for Online Multi-Person Tracking,” oct 2015.

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

[15] K. Bernardin and R. Stiefelhagen, “Evaluating Multiple Object Tracking Performance: The CLEAR MOTMetrics,” Image and Video Processing, , no. May, 2008.