TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Khwopa College Of Engineering
Libali, Bhaktapur
Department of Computer Engineering

A REPORT ON

VOICE ASSISTANT FOR VISUALLY IMPAIRED PEOPLE

Submitted in partial fulfillment of the requirements for the degree

BACHELOR OF COMPUTER ENGINEERING

Submitted by

Anusha Bajracharya	KCE074BCT011
Luja Shakya	KCE074BCT022
Niranjan Bekoju	KCE074BCT025
Sunil Banmala	KCE074BCT045

Under the Supervision of
Er. Ramesh Marikhu

Khwopa College Of Engineering
Libali, Bhaktapur
February 9, 2021

Certificate of Approval

The undersigned certify that the sixth semester project entitled **"Voice Assistant for Visually Impaired People"** submitted by Anusha Bajracharya, Luja Shakya, Niranjan Bekoju and Sunil Banmala to the **Department of Computer Engineering** in partial fulfillment of requirement for the degree of **Bachelor of Engineering in Computer Engineering**. The project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled, bona fide and ready to undertake any commercial and industrial work related to their field of study and hence we recommend the award of Bachelor of Computer Engineering degree.

.....
Er. Ramesh Marikhu
(Project Supervisor)

.....
Er. Shiva Kumar Shrestha
Head of Department
Department of Computer Engineering, KhCE

Copyright

Acknowledgement

We take this opportunity to express our deepest and sincere gratitude to our HoD Er. Shiva Kumar Shrestha, for his insightful advice, motivating suggestions for this project and also for his constant encouragement and advice throughout our Bachelor's program.
We are deeply indebted to our DHoD Er. Dinesh Man Gothe for boosting our efforts and moral by his valuable advises and suggestion regarding the project, giving us realization of the practical scenario of the project and supporting us in tackling various difficulties.
Also, we would like to thank Er. Bindu Bhandari for providing valuable suggestions and for supporting the project.

Anusha Bajracharya	KCE074BCT011
Luja Shakya	KCE074BCT022
Niranjan Bekoju	KCE074BCT025
Sunil Banmala	KCE074BCT045

Abstract

Visually impaired people struggle a lot in their life, being unable to see clearly is a major disorder for keeping their life in balance.Thus, for assisting those people in their daily life, we've decided to make a portable system that can be easily accessible to all visually impaired people through their mobile phones. In this system ,we'll detect the surrounding objects as from the user's camera utility and recognize those objects then, provide guidance about the objects to user with commanding speech.

Keywords: *YOLO, Deep SORT, Non-Maximum Suppression, Intersection over union*

Contents

List of Figures

List of Abbreviation

Abbreviations	Meaning
CNN	Convolutional Neural Network
COCO	Common Object in Context
CSP	Cross-Stage Partial Connections
CUDA	Compute Unified Device Architecture
DCT	Detection Classification and Tracking
FAR	False Alarm Ratio
FM	Number of Fragmentations
FN	False Negative
FP	False Positive
FPS	Frames Per Secon
GNU	GNU is not UNIX
GPU	Graphical Processing uni
GT	Number of Groundtruth Trajectory
GUI	Graphical User Interface
IDF1	ID F1 Score
IDP	ID Precision
IDR	ID Recall
IDs	Number of Identification switch
IOU	Intersection over union
LTS	Long Term Support
ML	Number of Mostly Lost Trajectory
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
MT	Number of Mostly Tracked Trajectory
NMS	Non Maximum Suppression
OpenCV	Open Computer Vision
PAN	Path Aggregation Network
PT	Number of Partially Tracked Trajectory
Prcn	Precision
RCNN	Region Based Convolutional Neural Network
RTX	Ray Tracing Texel eXtreme
Rcll	Recall

Abbreviations	**Meaning**
SORT	Simple Online Real time Tracking
SSD	Single shot Detection
TN	True Negative
TP	True Positive
TTS	Text To Speech
UI	User Interface
UML	Unified Modelling Language
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Background Introduction

People are facing problems in their daily life. They are facing problem to reach their destination because of traffic, obstacle and many more.But, in our society there are visually impaired people as well. These people find it difficult to even complete their daily routine. Even, the simple job like playing, walking, moving becomes more difficult for these people. The physically challenged people may be visually impaired people, deaf. There may be people who have diseases like polio, Parkinson's, Amputee Cancer. Among them, blind people face a lot problem. They cannot see the beautiful world. They have lost the opportunity to live a happy life with the nature. They cannot enjoy with the beauty of the earth. Besides, it's very difficult for them to travel from one place to another on their own. Some people use white cane for their help. It may provide some relief, some direction to them. But, They want some one who can help them, instruct them to reach their destination.

So, here we have come with an instructor to instruct those blind people. We are here with the system that will instruct a blind or visually impaired people with the current state environment.

In this project, we use object detection method, detect and recognize the object and give audio/vocal information about it.

1.2 Motivation

There is no doubt that the world for visually impaired people is difficult to live than normal people. Visually impaired people find it difficult to do their regular activities like finding materials, getting things on their own, arranging clothes, cooking, etc. But, The biggest challenge for a blind person, especially the one with the complete loss of vision, is to navigate around places. Obviously, blind people roam relatively easily around their house without any help because they know the position of everything in the house. But, it is not possible while going out where visually impaired people haven't been before.
Thus, our system provides Voice Assistant for guiding and instructing blind people and visually impaired people so that they can navigate around places by themselves.

1.3 Problem Definition

Blind people face several problems in their daily life. They lack their vision totally or partially due to neurological or physiological factors. In recent years, blind mobility has become an important issue since a large number of people are visually impaired and partially sighted. Globally, at least 2.2 billion people have a vision impairment or blindness, of whom at least 1 billion have a vision impairment that could have been prevented or has yet to be addressed. This 1 billion people includes those with moderate or severe distance vision impairment or blindness due to unaddressed refractive error, as well as near vision impairment caused by unaddressed presbyopia. Globally, the leading causes of vision impairment are uncorrected refractive errors and cataracts. The majority of people with vision impairment are over the age of 50 years. Navigation for visually impaired person is a great challenge as these people has to rely on other. The simplest and most widely used travelling aid used by all blinds is the white cane. It has provided those people with a better way to reach destination and detect obstacles on ground, but it cannot give them a high guarantee to protect themselves from all level of obstacles. Sometimes it happens that blind people are lost and their guardians get tensed about them. There have been many efforts but even now, it is not easy for the blind people to move. To solve this great problem it has been studied by many researches about support instruments for eye-sight. According to the problems stated earlier we have prepared a system in this paper which aid blind person by informing about the obstacles around them which help them to move around without the help of other.

1.4 Goals and Objectives

The main objective of this project is:

- To instruct the visually impaired people using voice assistant.

1.5 Scope and Applications

The major scope of the project is to assist visually impaired people and the system is able to:

- be used by visually impaired people worldwide.
- provide a voice assistance for guiding and instructing Visually Impaired People.

1.6 Report Organization

1.6.1 Introduction

The main purpose of the introduction is to set the scene for our readers so that they can know about the problems that visually impaired people have to go through and how our purposed system can help them.

1.6.2 Literature Review

The literature review is there to familiarize ourselves with the current state of knowledge on assistance for visually impaired people and ensure us to not repeat what others have already done.

1.6.3 Feasibility Study

It is used to determine the viability of an idea, such as ensuring our project is legally and technically feasible as well as economically justifiable or not.

1.6.4 Methodology

It critically helps us to analyze and select correct method for our project to avoid unnecessary hurdles.

1.6.5 Requirement Analysis

It gives us a clear vision about the necessary programming languages and software required to build voice assistance.

1.6.6 System Design and Architecture

The main purpose of this is to evaluate the contribution of each component for overall performance of the system using different diagrams.

1.6.7 Block Diagram and Description of Proposed System

It provides us quick and high-level view of different topics which enhances our ability to understand that topic.

1.6.8 Expected Outcome

It helps to ensure that our set goal is achieved and not to repeat what others have done already in this topic.

1.6.9 Actual Outcome

It provides clear view about the accomplishments we have achieved and comparison can be made with expected outcome to note down deviations.

1.6.10 Conclusion and Future enhancements

The conclusion section summarizes our main thoughts on project and future enhancements provides us scope to upgrade our project.

Chapter 2

Literature Review

2.1 You Only Look Once: Unified, Real-Time Object Detection

In this paper [1], Redmon et. al. presented a new approach to object detection. Authors compared their model with other Real-Time Systems namely Deformable Parts Model(DPM), R-CNN, Fast Detectors, Deep MultiBox, OverFeat, MultiGrasp. Authors took the best part from those models and build their own model which was faster, better and worked in real time. This model was fast as it only looked once in the entire image and then detect the object in the image as the regression problem. The base YOLO model processed images in real-time at 45 fps. A smaller version processed an astounding 155 fps still achieving double the mAP if the other real time detectrs.
In this paper, the authors also included the architecture of the model, and the process the authors used for training. The final output of the network was the 7*7*30 tensor of predictions which could predict at most 98 object.

2.2 YOLO9000:Better, Faster, Stronger

Redmon et. al. presented about the YOLO v1 in [1]. YOLO model made signifant no. of localization error. It had very low recall. So, the authors presented a new model better than YOLO v1 and created YOLOv2 with the improvement in object localization and recall. In paper [2], Authors not only presented about the YOLOv2 also about the YOLO9000 a realtime detection system. YOLO9000 was a realtime framework for detection more than 9000 object categories by jointly optimizing detection and classification. Authors used WordTree to combine data from various source and the authors had joint optimization technique to train simultaneously in ImageNet and COCO dataset. YOLO9000 was a strong step that closes the gap between the classification and detection. Authors made a better model by introducing Batch Normalization that significantly improved in convergence of model by eliminating regularization. Authors added BatchNormaliztion in all Convolutional Layer that increased mAP by 2%.
Authors created a model faster than ever by using darknet-19 as their base framework. Darknet-19 has 19 Convolutional layer and 5 max pooling layes with only 5.58 billion operation. This figure was less that the no. of operation on VGG-16.

VGG-16 has 30.69 billion floating point operation for a single pass over a single image. So, it was obvious the YOLO to be faster.
YOLO9000 was not only faster and better, but was stronger too. As, the authors purposed a mechanism for jointly training on classification data as well as detection data. The proposed method used image labelled for detection to learn detection-specific information like bounding box prediction, objectness score calculation and classify common objects. Also it used imagse with only class lables to expand the number of categories it can detect.

2.3 YOLOV3: An Incremental Model

Redmon et. al. [3] presented some update to YOLO. Authors made the model little bigger than last time but more accurate. It was still fast and run in 22ms at 28.2 mAP for 320 * 320 image. It was as accurate as SSD but three times faster that SSD. Here, Authors did not use softmax activation function in last layer instead simply used independed logistic classifier. Also, During training Binary cross entropy was used for the class predicitions. This formulation helps when we move to more complex domain with many overlapping labels (i.e. woman and person).

2.4 YOLOv4: Optimal Speed and Accuracy of Object Detection

YOLOv4 is a one-stage object detection model that improves on YOLOv3 with several bags of tricks and modules introduced in the literature. [4] In this section, the authors elaborated the details of YOLOv4. YOLO v4 was developed by three developers Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4 consists of:

- Backbone: CSPDarknet53
- Neck: SPP , PAN
- Head: YOLOv3

YOLO v4 uses:

- Bag of Freebies (BoF) for backbone: CutMix andMosaic data augmentation, DropBlock regularization,Class label smoothing.
- Bag of Specials (BoS) for backbone: Mish activation, Cross-stage partial connections (CSP), Multiinput weighted residual connections (MiWRC).
- Bag of Freebies (BoF) for detector: CIoU-loss CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for a single ground truth, Cosine annealing scheduler , Optimal hyperparameters, Random training shapes.
- Bag of Specials (BoS) for detector: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

2.5 Deep-Sort/nwokje

Previously a research paper for deepsort by Nicolai Wojke, Alex Bewley and Dietrich Paulus started integration appearance information to improve the performance of SORT. [5] Due to this extension, the authors were able to track objects through longer periods of occlusions, effectively reducing the number of identity switches. In spirit of the original framework the authors placed much of the computational complexity into an offline pre-training stage where the authors learned a deep association metric on a large-scale person re-identification dataset. During online application, the authors established measurement-to-track associations using nearest neighbor queries in visual appearance space. Experimental evaluation showed that the extensions reduced the number of identity switches by 45%, achieving overall competitive performance at high frame rates.

2.6 Deep SORT and YOLO v4 for people tracking and detection with Tensorflow backend/ LeonLok

In this project,YOLO v3 swapped out for YOLO v4 and added the option for asynchronous processing, which significantly improved the FPS. However, FPS monitoring was disabled when asynchronous processing was used since it wasn't accurate. In addition, the algorithm from the paper of Deep-Sort by Nwokje was used for tracking of objects. The original method for confirming tracks was based simply on the number of times an object has been detected without considering detection confidence, leading to high tracking false positive rates when unreliable detections occur (i.e. low confidence true positives or high confidence false positives). The track filtering algorithm reduced that significantly by calculating the average detection confidence over a set number of detections before confirming a track. This code only detects and tracks people, but could be changed to detect other objects by changing lines so it could also be implemented for detecting different objects including person.

2.7 Maximum correntropy Kalman filter

Traditional Kalman filter (KF) was derived under the well-known minimum mean square error (MMSE) criterion, which was optimal under Gaussian assumption. [6] The best part of the Kalman filter is that it is recursive, meaning where we take current readings, to predict the current state, then use the measurements and update our prediction.Traditional Kalman filter works well under Gaussian noises, but its performance may deteriorate significantly under non-Gaussian noises, especially when the underlying system is disturbed by impulsive noises. The main reason for this was that KF was developed based on the MMSE criterion, which captures only the second order statistics of the error signal and was sensitive to large outliers. To address this problem, the authors proposed in this work to use the MCC criterion to derive a new Kalmanfilter, which may perform much better

in non-Gaussian noise environments, since correntropy contains second and higher order moments of the error. Following problem were solved:

- When object was partially occluded, place weight or reliance on both motion and sensor measurement data on that object.
- if its fully occluded, shift a lot of weight on motion data of an object.

Chapter 3

Feasibility Study

3.1 Technical Feasibility

We have used YOLO weights file as pretrained detection model and DEEP SORT for tracking so technically, we have no problems with model and datasets. Since we have used a smartphone to get video from camera as input, process them using GPU reinforced PC and used speaker for output so, this project is feasible.

3.2 Operational Feasibility

Operation was done by using the pretrained model from YOLO v4 weights file and then using the Deep sort tracking mechanism for object tracking. We intended to deploy using the Android app in our next iteration of the project but in this first iteration we deployed our project into the desktop application using python GUI framework where, for testing phase we can use webcamera or any other IP camera or video then apply the project.
The main parts of our system are:

3.2.1 Object Detection and Tracking

Deployment of inference system on PC was quite easy due to use of YOLO-V4 pre trained module for detection and for tracking part, we have used Deep Sort algorithm which was pre defined and easy to implement. Integration and synchronization of these modules were quite difficult, but we manage to fix those difficulties.

3.2.2 Text To Speech Module

In context for the desktop application developed in python using the pyttsx3 package proved to be best and the engine used for this operation is espeak in ubuntu 18.04 LTS. Without using multi threading programming there was a slight problem associated with the TTS module as it takes more time to speak the given text and at that time instance, GPU has to remain idle therefore, we implemented multi thread programming to address this problem. Use of multi thread programming made it possible for simultaneously doing TTS operations as well as video processing for the proper object detection and tracking as well.

With all the modules merged and connected to work together, our project for developing the Desktop application has been completed.

3.3 Economic Feasibility

The total expenditure of the project is in computational power. For computational power we are using YOLO V4 weights file and for computational power, we are using high end PC with RTX 2080Ti provided by College and also our own laptop. So, the project is economically feasible.

3.4 Scheduling Feasibility

Schedule we have managed to meet the deadline of project is mentioned in Gantt Chart at 6.5 Schedule. Here is proper management of time as we can for this project is done. We have given almost top priority for the optimum analysis of the algorithm to get higher accuracy and testing of models. Documentation remains as important as any other task so throughout the whole project documenting each and every work will be continued. For this in every week we have developed the weekly report and in every meeting, minutes of that particular meeting is taken for the works to be done in next week. Thus, within our time management skill we are ensured this project will be schedule feasible.

Chapter 4

Methodology

4.1 Agile Method as Software Development Model

Building overall project in python and implementing in android smartphone was a big challenge which can be made easy by building project with a small section and developing it to a fully functioned project. Implementing the whole project inside android app was a challenging part as the processing power and efficiency of mobile device could be compromised. Therefore Agile model was used as Software Development model as it facilitied us to start our project from a small section and launch it as beta version which helped us to visualize where the project was going and which part it had defect.

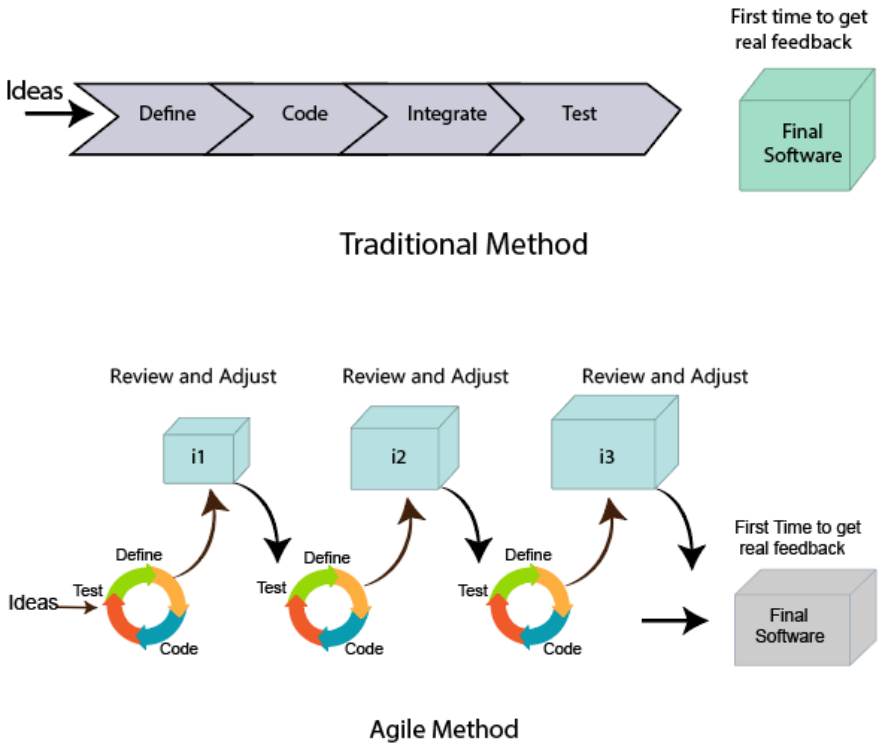


Figure 4.1: Agile Model as Software Development Model

4.2 Asana as Kanban Board

Asana was used as Kanban Board to organize, distribute, manage and track all the works and tasks assigned to each members of our team. All the works done in base level as well as the meeting minutes were recorded in Asana.
The Project was managed in asana in five sessions:

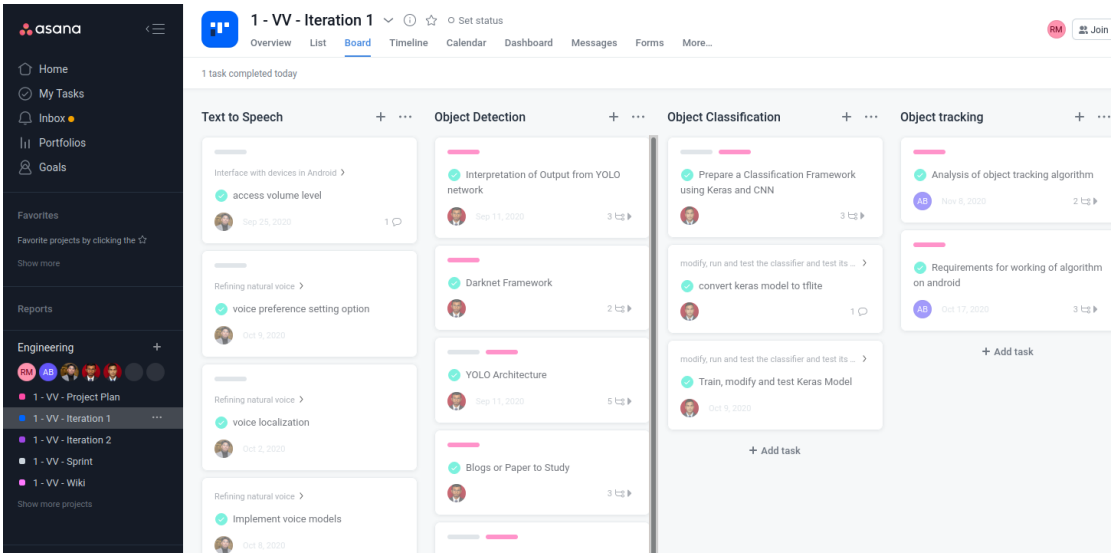


Figure 4.2: Asana

4.2.1 Project Plan

Planning for further steps were done in detail in this session. All the tasks to be done, their sub tasks, deadline of the tasks and subtasks were recorded here. Then the task was linked to respective iteration session by the assigned member before doing the task.

4.2.2 Sprint

The works decided to be done, being done and the one that were done recently were recorded here.

4.2.3 WIKKI

Utility tasks that are useful and informative but unrelated to the actual project were listed here. Forexample: tutorials and guidelines to use gitlab, asana, etc were listed here.

4.2.4 Iteration1

Everything related to the first iteration of our project were recorded here including meeting minutes and reports.

4.2.5 Iteration2

Everything related to the second iteration of our project were recorded here including meeting minutes and reports.

4.3 Overall Phase Followed

The overall project has been completed in three main phase which are:

1. Planning Phase
2. Development Phase
3. Integration

4.3.1 Planning Phase

In planning phase, necessary works to be done and planning of overall project were discussed and the decision were documented for further procedures.
First the project was divided into four parts:

1. Image detection
2. Image classification
3. Imge Tracking
4. Text to Speech

These parts were then assigned to each project members who then studied about the respective parts in detail.

4.3.2 Development Phase

In this phase, the divided parts were well studied and developed. Then each of those developed parts were tested separately.

4.3.3 Integration

In this phase, the separately developed parts were integrated to form a system and integration testing was done.

4.4 Task Workflow

Each tasks of every session were done and recorded in a procedural manner and the programmes were recorded and stored in gitlab.
The workflow of the tasks follow following steps:

1. Get task assigned in Asana

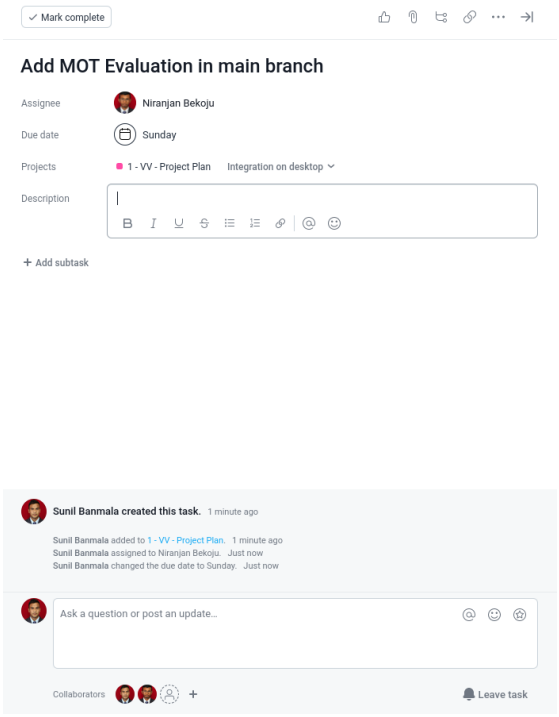


Figure 4.3: Create task in asana

2. Create issue in gitlab for each tasks assigned to self

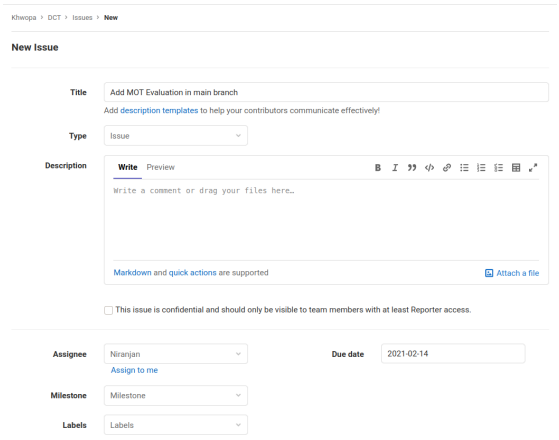


Figure 4.4: Create issue in gitlab

3. Create corresponding branch in gitlab and fetch the branch in local
4. Checkout to the branch and do assigned tasks there
5. Then push the task done, to the gitlab
6. Mark the task assigned in Asana to: Done

Chapter 5

Requirement Analysis

5.1 Software Requirement

Software requirement for our prepared system includes:

1. Python
2. Shell
3. Star UML
4. Visual Studio Code
5. Google Colaboratory
6. CUDA
7. Gitlab
8. Slack
9. Texmaker
10. Beamer
11. Asana
12. Instagantt
13. Microsoft Team
14. Google Drive
15. Google Calender

5.2 Hardware Requirement

Our prepared system of virtual assistant requires following hardware requirements:

1. Android Mobile for video capturing.
2. Desktop with NVIDIA Graphics Card (11 GB,RTX 2080 Ti) and 16 GB RAM.

5.3 Functional Requirement

The functional requirement for the prepared systems are:

1. The system must be able to capture image/video in real time.
2. It must instruct visually impaired people in real time.
3. It must not disturb and produce unwanted commands instructions.

5.4 Non-Functional Requirement

These are essential for the better performance of the system. The points below focus on the non-functional requirement of the system prepared.

5.4.1 Reliability

Different test metrices must be conducted to test the accuracy of the system.

5.4.2 Maintainability

The System is breakdown into multiple sub module so that we could know where the problem is and maintain the sub module.

5.4.3 Performance

The object detection and only single process is required. So, it will be able to provide better performance.

5.4.4 Frame Per Sec

For the smoothnesss of the system, we required at least 4 to 7 frame per sec.

Chapter 6

System (or Project) Design and Architecture

6.1 Use Case Diagram

The Use Case Diagram of the prepared inference system.

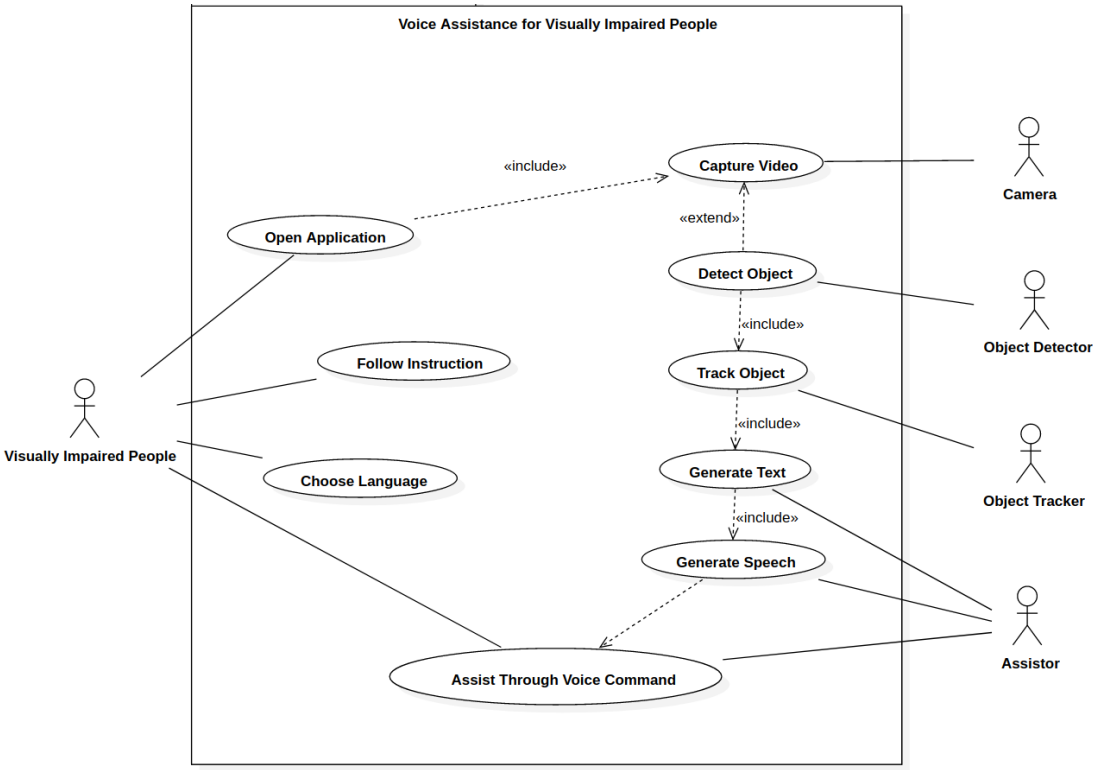


Figure 6.1: Use case Diagram

6.2 Context Diagram

The Context Diagram shows the top level picture of the system.

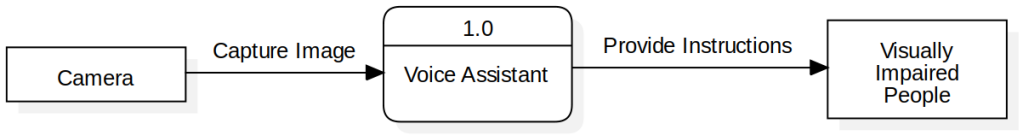


Figure 6.2: Context Diagram of Inference System

6.3 Data Flow Diagram

The Data Flow Diagram shows the flow of the data between the subsystem of the inference system. The Data flow Diagram of the inference system is shown below:

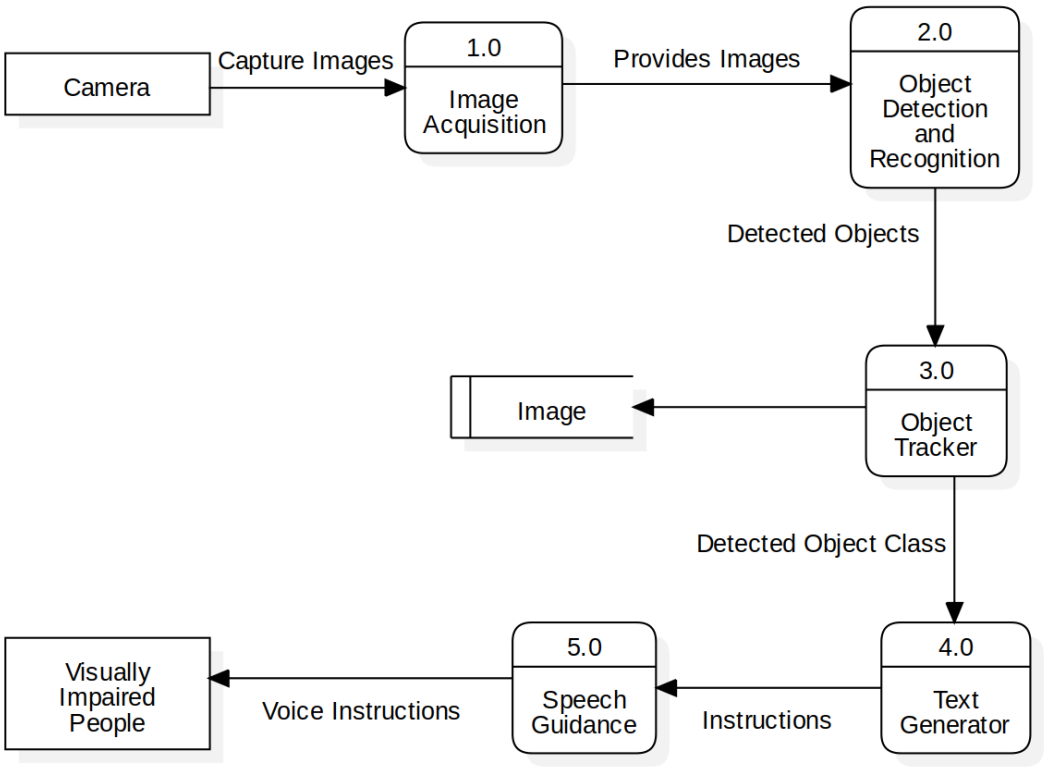


Figure 6.3: Data Flow Diagram of Inference System

6.4 Sequence Diagram

The Sequence Diagram shows the flow of the process in between the subsystem. And the state when the subsystem are active and when the sub system are passive. The Sequence Diagram of the inference system is shown below:

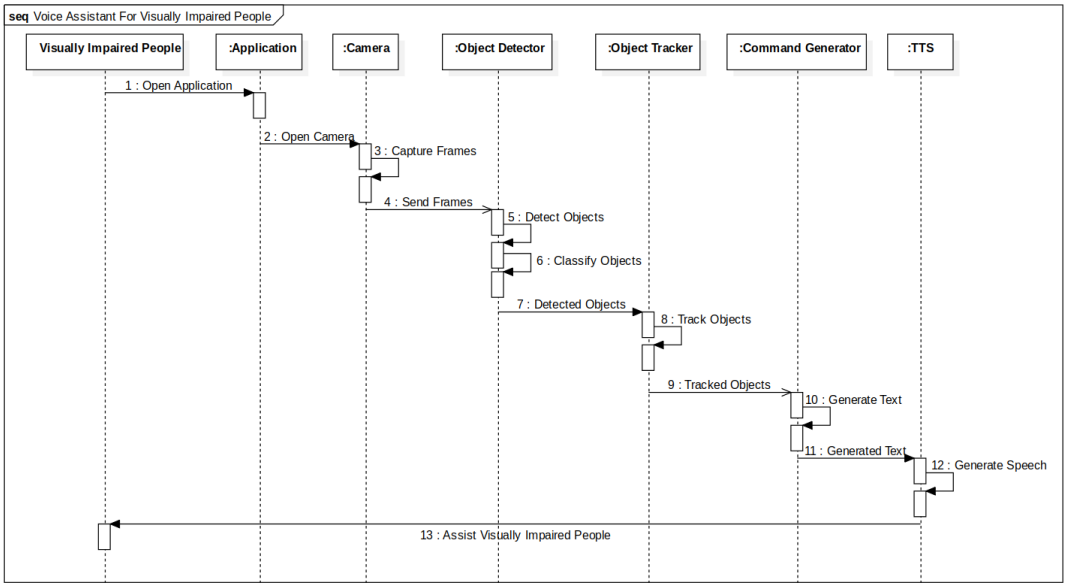


Figure 6.4: Sequence Diagram of Inference System

6.5 Schedule (Gantt Chart)

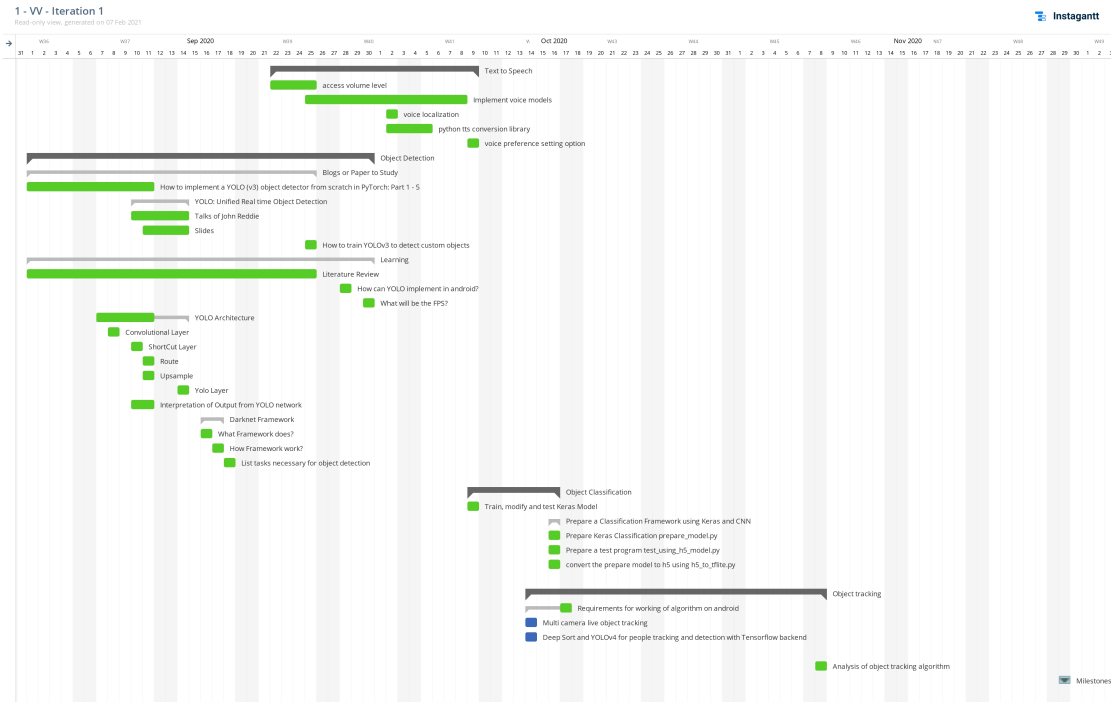


Figure 6.5: Gantt Chart

Chapter 7

Block Diagram and Description of Prepared System

7.1 System Block Diagram

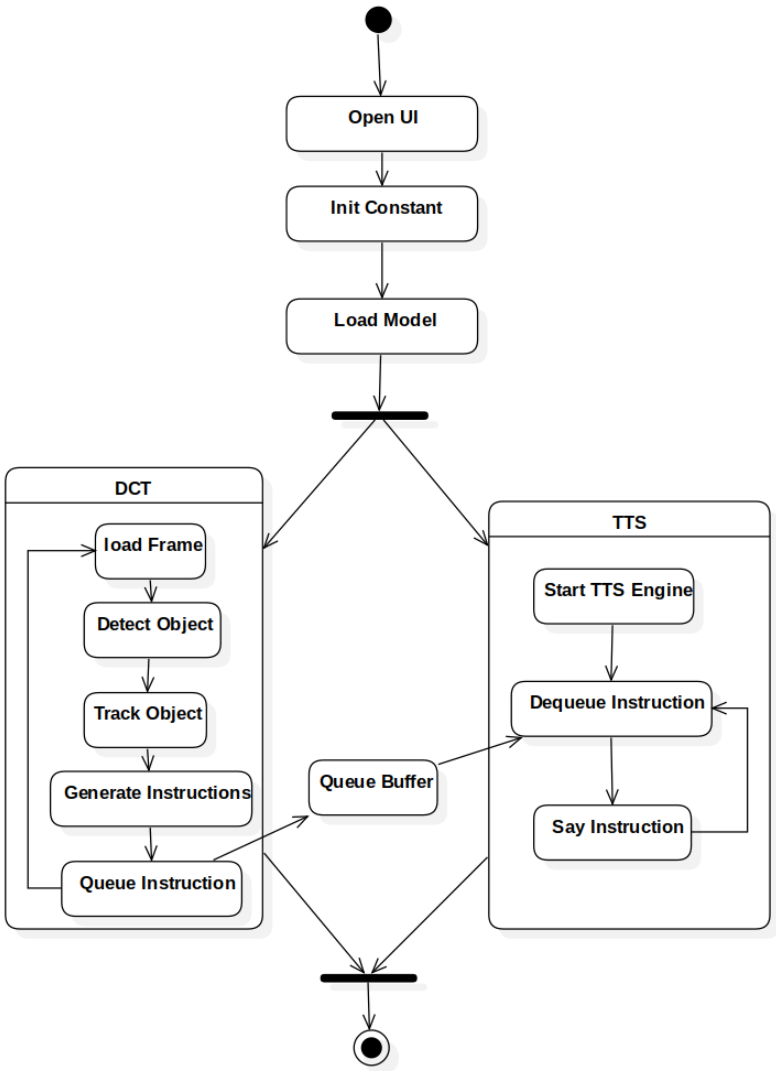


Figure 7.1: System Block Diagram

The person will carry a systemin android phone . When the application is opened, it will access the camera. The camera will capture the video in real time with certain frame per second. Objects in each selected frame will be detected and recognized with the help of YOLO(You Only Look Once) algorithm. It will give what object and where the obect is located in the image. The object is tracked using deepsort algorithm for tracking same object over following frames. Then, according to the object detected and their movement, a voice guidance is provided so that the person is made known about the object and be able to dodge the object instead of colliding with it.

7.2 YOLO - You Only Look Once

7.2.1 What is YOLO?

YOLO stands for You Only Look Once. It's a object detector that uses features learned by a Deep Convolutional Network(DFN) to detect an object. YOLO makes use of only Convolutional Layers.

7.2.2 Feature of YOLO

Some of the features of YOLO is given below:

- It has 75 convolutional layer.
- It does not have any form of pooling.
- YOLO is invariant to size of input image yet, we stick on constant input size.
- The network downsample image by a factor called stride of network.

  for eg. If the stride of network is 32.

  input size = 416*416

  then, Output size = 13*13

  Output size is stride times smaller than input size.

7.2.3 Interpreting the Result

The output of the YOLO object detection is a feature map. Each cell in the output image can predict a fixed number of bounding boxes. The YOLOV3 produces 3 bounding box for each cell. In feature map, we've B *(5+C) entries in feature map.

where,

B represents the number of bounding box each cell can predict.
Each bounding box have 5+C attributes,
here, C represents class confidence for each bounding box.
Extra 5 is for
t_x for box co-ordinate
t_y for box co-ordinate

t_w for box co-ordinate
t_h for box co-ordinate
p_o for Objectness score

Each cell of a feature map predict an object through one of it's bounding box. One Bounding box is responsible for detecting only one given objects.

7.2.4 How we do?

Consider,

Image Size = 416*416
stride = 32
Output Image = 13 * 13 is Image of 13*13 cells

Each cell gives feature map of B * (5+C) entries.

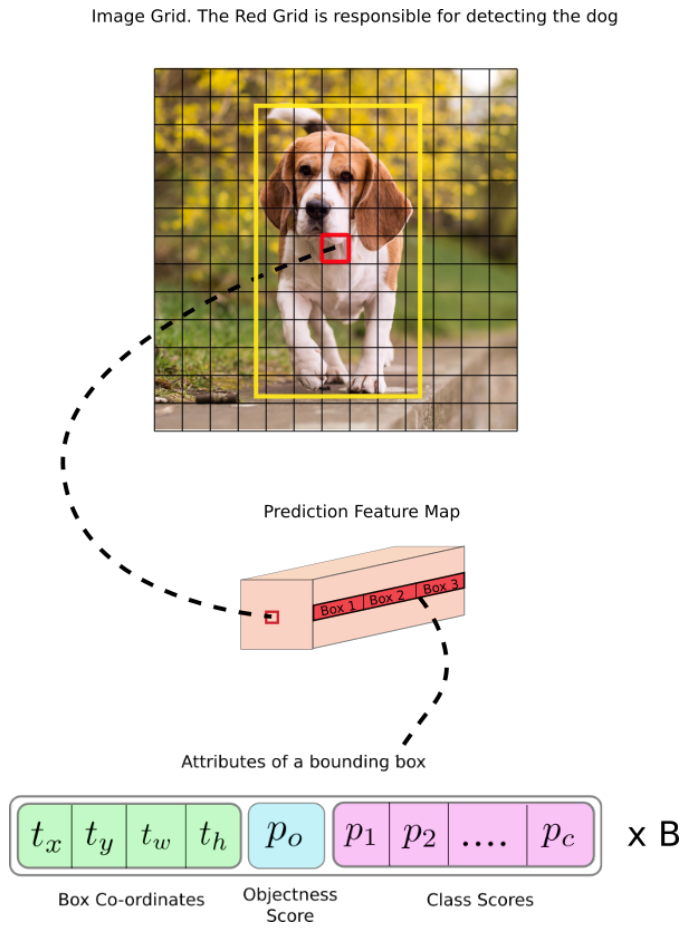


Figure 7.2: Feature map of each cell

Objectness Score

One of the 5 attribute P_o is probability that the object is contained inside a bounding box.

P_o is nearly equal to 1 if the object is contained. It mainly occurs at the center of ground truth box.

P_o is nearly equal to 0 if the cell contain no object. It mainly occurs at the corners of ground truth box.

Class Confidence

Class Confidence in the bounding box are the probabilities of detected object belong to particular class. Before YOLOV3, soft max was used for the class score. In V3, Soft max was dropped and Sigmoid was adopted instead. It is because Soft max assumes classes are mutually exclusive. i.e. If an object belongs to one class, then it cannot belongs to another class(True only for COCO data set).It is not true when we've classes like women and person which are mutually inclusive.

7.2.5 Benefits of YOLO

- Fast and Good for real time processing
- Predictions are made from one single network.

7.2.6 YOLO Outputs

Figure 7.3: Test image feeded to YOLO network

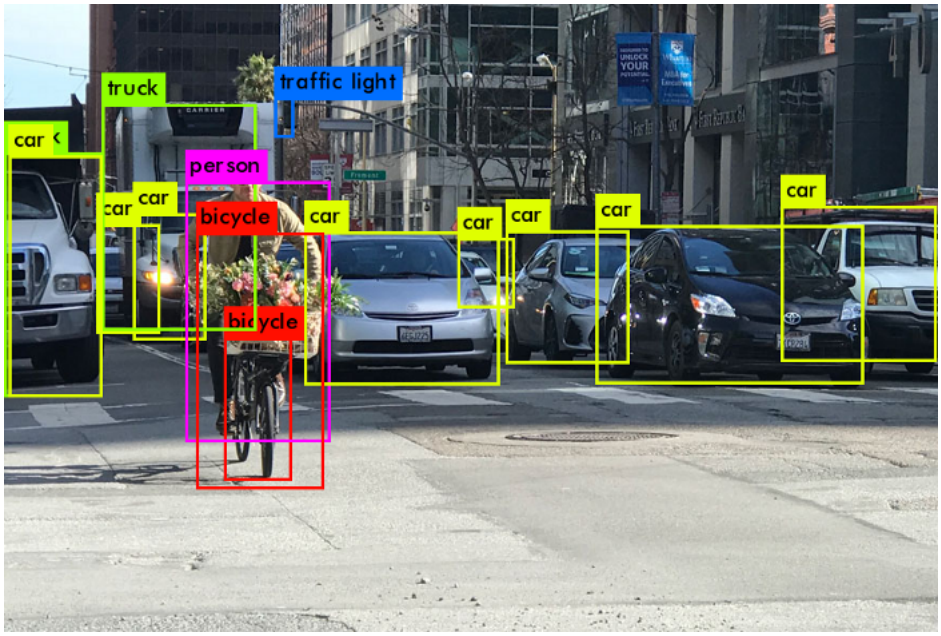


Figure 7.4: Output image from YOLO network

7.2.7 YOLO v1 – CNN Design

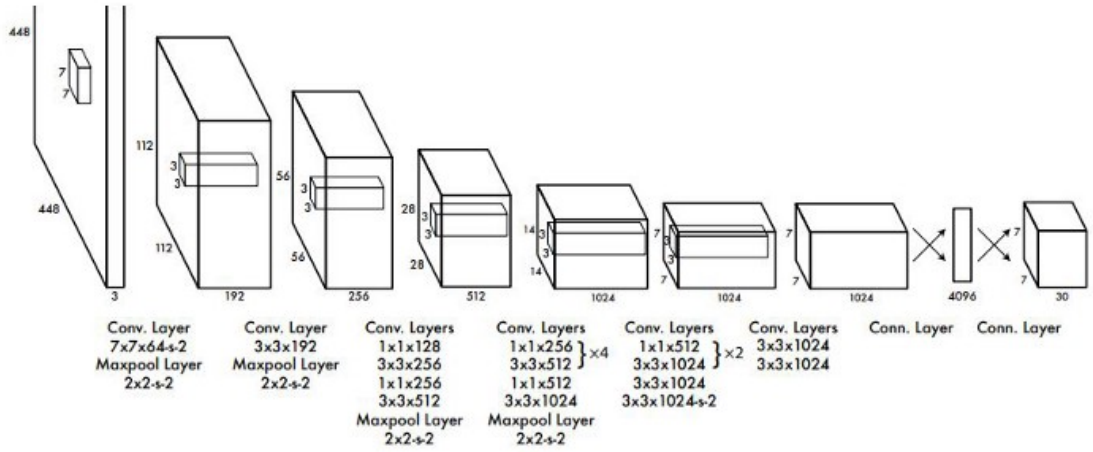


Figure 7.5: YOLO v1 CNN

YOLO v1 CNN is shown in fig 7.5. [1] It has 24 convolution layer followed by 2 fully connected layers that are responsible for classification of objects and regression of bounding boxes. Here, convolution layers acts as a feature extractor. The final output is 7 x 7 x 30 tensor. Leaky RelU activation function is used for all layers except the final layer. The final layer uses a linear activation function.

7.2.8 Loss Design

YOLO design uses sum squared error as the backbone of loss. [1] Since, multiple grid in the output do not contain any objects and their confidence score is zero. They overpower the gradients from a few cells that contain the objects. To avoid such overpower that leads to training divergence and model instability, YOLO increases the weight ($\lambda_{coord} = 5$) for prediction from bounding box containing object and reduces the weight $\lambda_{noobj} = 0.5$ for predictions from bounding boxes that do not contain any objects.

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2] \tag{7.1}$$

The equation 7.1 shows the first part of YOLO loss. It calculates the error in the prediction of center coordinates of bounding box. The loss function penalizes error in center coordinate of the bounding box, if the predictor is responsible for the ground truth box.

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \tag{7.2}$$

The equation 7.2 shows the second part of YOLO loss. It calculates the error in prediction of bounding box width and height. Here, if the magnitude of error in small bounding box is same as that of large bounding box. It is more wrong for small bounding box that a large bounding box. Hence, a square root of those values is used to calculate the loss. The loss function penalizes error in width and height of the bounding box, if the predictor is responsible for the ground truth box.

$$\sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \tag{7.3}$$

The equation 7.3 shows the third part of YOLO loss. It calculates the error in prediction of object confidence score for bounding boxes that have an object. The loss function only penalize object confidence score, if that predictor is responsible for the ground truth box.

$$\lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \tag{7.4}$$

The equation 7.4 shows the fourth part of the YOLO loss which calculates the error in prediction of object confidence score for bounding boxes that do not have an object.

$$\sum_{i=0}^{s^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{7.5}$$

This equation 7.5 shows fifth part of YOLO loss. It calculates the error in prediction of class probabilities for grid cells that have an object. The loss funciton only penalizes class probabilities error, if an object is present in that grid cell.

The loss function is:

$$
\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2] \\
+\lambda_{coord} &\sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
&+ \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \\
&+\lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \\
&+ \sum_{i=0}^{s^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{7.6}
$$

$\mathbb{1}_{i,j}^{obj}$ is 1 when j^{th} bounding box of i^{th} grid has object, otherwise 0.

7.3 Intersection Over Union

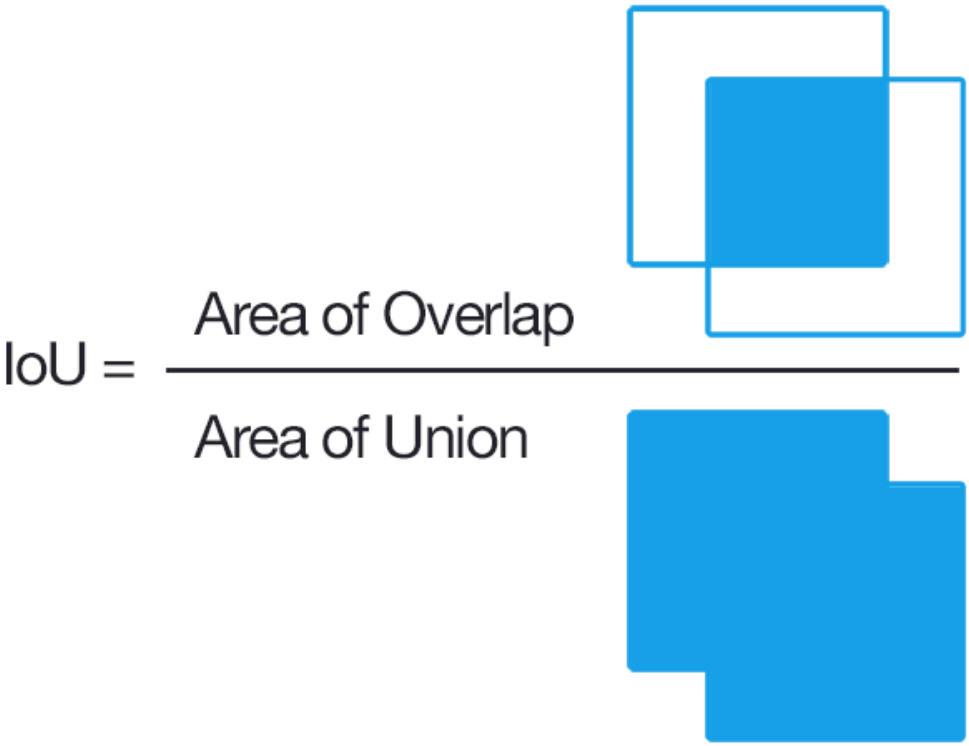


Figure 7.6: Intersection Over Union

Intersection over union(IOU) is a measure of overlap between two bounding box. In computer vision it is used for correctly detecting an object. To know object detection first you have to know about object localization. Object localization refers to figuring out where is the object in the picture and showing it with rectangular box. IOU is known to be a good metric for measuring overlap between two bounding boxes or masks.
Following pictures shows what is goodness measure of IOU
Informally, IOU measures how equal two areas are. In terms of size and location

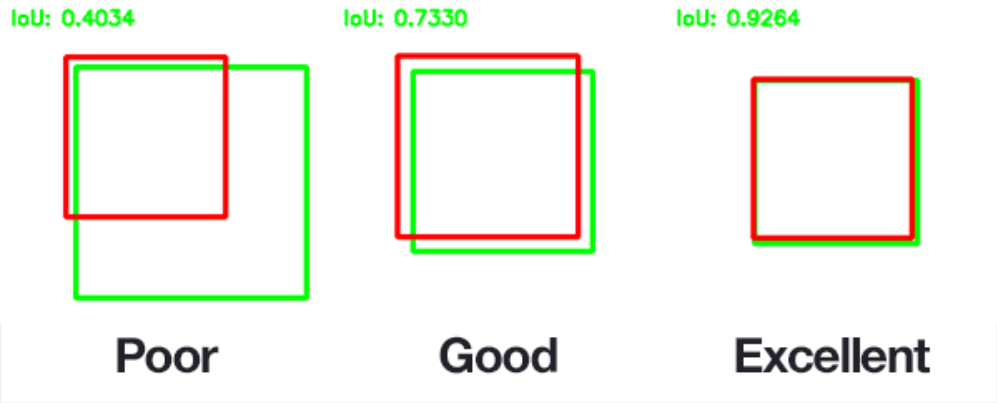


Figure 7.7: Goodness measure of IOU

of the area. If two areas are exactly equal, IOU will be 1. If two areas are far

apart, even if their shape is same, they will have IOU 0. And if two areas lie at the same location but their size differs a lot, then also IOU will be a small value. [7]

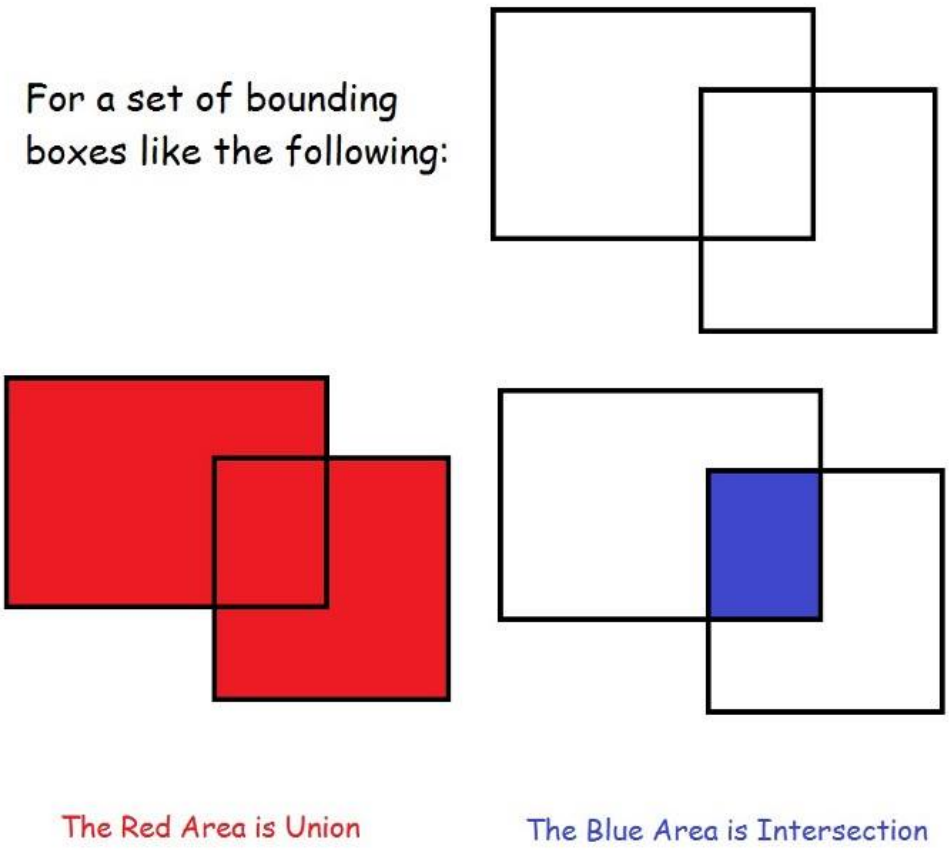


Figure 7.8: Bounding Box Union and Intersection

$$IOU(Box1, Box2) = \frac{Intersection_Size(Box1, Box2)}{Union_Size(Box1, Box2)}$$

7.4 Non-Maximum Suppression

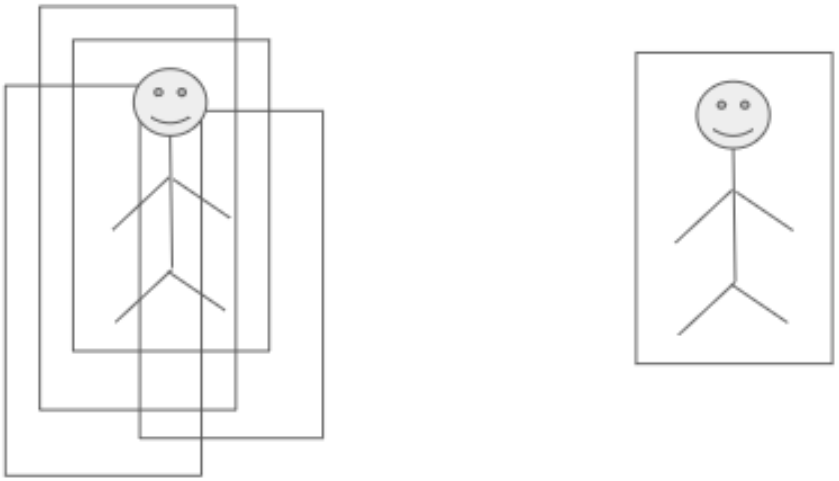


Figure 7.9: Non Maximum Suppression

Non Maximum Suppression (NMS) is a technique used in many computer vision algorithms. It is a class of algorithms to select one entity (e.g. bounding boxes) out of many overlapping entities. The selection criteria can be chosen to arrive at particular results. Most commonly, the criteria is some form of probability number along with some form of overlap measure (e.g. IOU).

In object detection, there is a problem that an algorithm detects multiple bounding boxes for a single object. To solve this problem there, is a technique called non-max supression.Non-max suppression cleans up the multiple detection and end with just one detection per object. For this it chooses the bounding box with highest probability and suppressed all the other bounding boxes who's IOU with it is greater, so in last only one bounding box is left which is more accurate. [8]
Input: A list of Proposal boxes B, corresponding confidence scores S and overlap threshold N.
Output: A list of filtered proposals D.
Algorithm:

1. Select the proposal with highest confidence score, remove it from B and add it to the final proposal list D. (Initially D is empty).

2. Now compare this proposal with all the proposals — calculate the IOU (Intersection over Union) of this proposal with every other proposal. If the IOU is greater than the threshold N, remove that proposal from B.

3. Again take the proposal with the highest confidence from the remaining proposals in B and remove it from B and add it to D.

4. Once again calculate the IOU of this proposal with all the proposals in B and eliminate the boxes which have high IOU than threshold.

5. This process is repeated until there are no more proposals left in B

Figure 7.10: Result before and after Non-max Suppression

Most object detection algorithms use NMS to whittle down a large number of detected rectangles to a few. The need for NMS arises because of the way object detection works in most cases (e.g. Faster RCNN, Yolo V3, SSD, etc). At the most basic level, most object detectors do some form of windowing. Many, thousands, windows of various size and shapes are generated either directly on the image or on a feature of the image (e.g. after a deep CNN such as ResNet). These windows supposedly contain only one object, and a classifier is used to obtain a probability/score for each classes. Once the detector outputs the large number of bounding boxes, it is necessary to pick the best ones. NMS is the most commonly used algorithm for this task. In essence it is a form of clustering algorithm. Attempts have been made to use standard clustering algorithms such as k-means, Nearest Neighbor, DB Scan, etc. in object detection.

7.5 Kalman Filter in Deepsort

The most popular and one of the most widely used, elegant object tracking framework is Deep SORT, an extension to SORT (Simple Real time Tracker).

The Kalman filter keeps track of the system and the variance or uncertainty of the estimate. The estimate is updated using a state transition model and measurements.

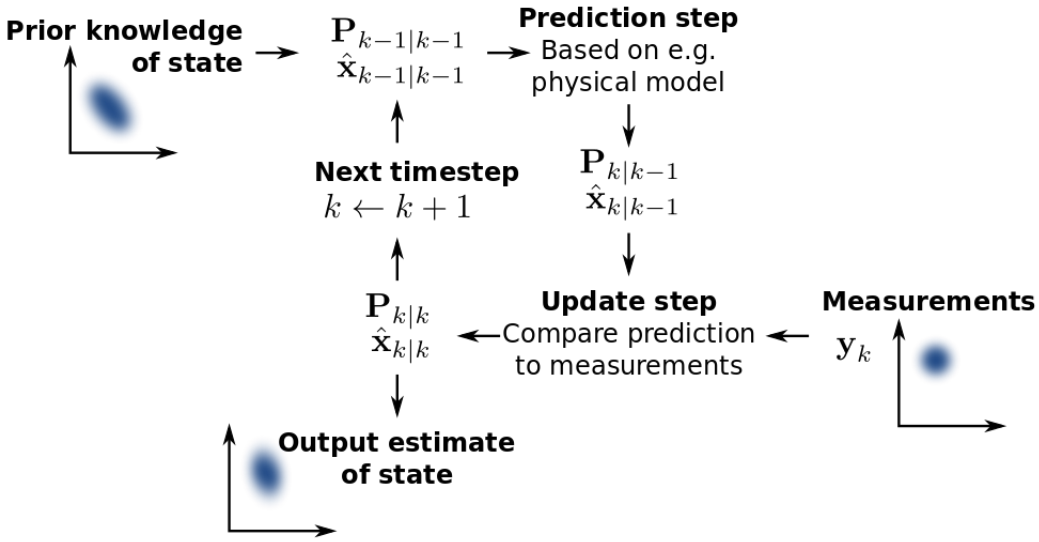


Figure 7.11: Kalman Filter

$\hat{x}_{k|k-1}$ denotes the estimate of the system's state at time step k before the k-th measurement y_k has been taken into account;$P_{k|k-1}$ is the corresponding uncertainty.

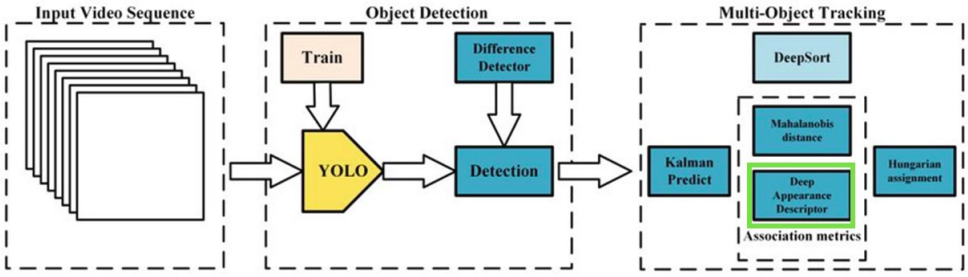


Figure 7.12: YOLO with Kalman Filter

7.6 Threads Implementation in Desktop

There were basically two modules in the inference system which were DCT module and TTS module. To bring synchronization between those two modules, threads was implemented which let those two modules run side by side.
The main logic was the use of a buffer variable: command list in which the DCT module stores the information to be given to the user and the TTS module actually executes the information.
In actual programme, the main module is DCT module which calls TTS modules in a threads. In DCT module:

```
self.thread = TTS()
self.thread.start()
self.thread.setLanguage(False)
```

To test the threads, a test program was made to make sure it works in which simply two audio files were played side by side:

```
import threading
import time
import playsound
song = "1.mp3"
class myThread (threading.Thread):
    def __init__(self, song_name):
        threading.Thread.__init__(self)
        self.song_name = song_name
    def run(self):
        playsound.playsound(self.song_name)
thread1 = myThread("1.mp3")
thread2 = myThread("2.mp3")
thread1.start()
thread2.start()
print("Exiting Main Thread")
```

As output, we get two audio files playing side by side at a same time.

7.7 Region Detector

Region detector gets ROI and Frame from the inference system and returns the region(left, center or right) on which the ROI lies. The steps involved are:

- Region detector gets: Frame width , Frame height and ROI(Region Of Interest) coordinates from the inference system.
- Then the detector calculates the coordinates for left, center and right region.

```
right = [0,0,int(self.width/3),self.height]
center = [int(self.width/3),0,int(self.width/3*2),self.height]
left = [int(self.width/3*2),0,int(self.width),self.height]
```

- Then it finds intersection of ROI with each of the left, right and center region and select the miximum one.

```
l = [ ]
l.append(self.intersection(center,self.roi))
l.append(self.intersection(left,self.roi))
l.append(self.intersection(right,self.roi))
return l.index(max(l))
```

- The maximum intersection gives the region on which the ROI lies.

7.8 Command Index Generation(CIG) for TTS

Command index generation generates the index of the command to be executed for the detected object.
The object classes as recognized by the class index are:

```
['car','truck','bus','person','bicycle','motorbike']
```

And the commands as recognized by command index are:

```
['car ahead',
'car in your left',
'car in your right',
'Truck ahead',
'Truck in your left',
'Truck in your right',
'Bus ahead',
'Bus in your left',
'Bus in your right',
'person ahead',
'person in your left',
'person in your right',
'Bicycle ahead',
'Bicycle in your left',
'Bicycle in your right',
'Bike ahead',
'Bike in your left',
'Bike in your right']
```

After getting region value from Region Detector and class index value from inference system, CIG now calculates the actual index of the command using the formula:

```
command_index = class_index * 3 + regionValue()
```

Now, with command index value, TTS module can execute the required command.

7.9 Text to Speech (pyttsx3)

Text to speech conversion is simply done by using pyttsx3 library which uses pre-installed text to speech synthesizer engine to formulate text given as input then generating audio file as output. here, Included TTS engines:

- sapi5
- espeak
- libspeak
- festival

The text-to-speech (TTS) synthesis procedure consists of two main phases. Namely:

Generation of linguistic representation: The input text is transcribed into a phonetic or some other linguistic representation

Generation of speech waveforms: The output is produced from this phonetic and prosodic information

The character string is then pre-processed and analysed into phonetic representation which is usually a string of phonemes with some additional information for correct intonation, duration, and stress. Finally the audio file is generated as the calculated parameters of speech from the TTS engines for the audible result to user.

7.10 Detection Evaluation

Evaluation for output of Detection is done using F1 Score analysis. It is determined from confusion matrix as:

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Figure 7.13: Confusion Matrix

True Positive (TP):
Number of Correct Detections in each frame.

False Positive (FP):
Number of Wrong Detection in each frame.

False Negative (FN):
Number of Missed Detection in each frame.

Precision:
Precision is a good measure to determine, when the costs of False Positive is high. Precision talks about how precise/accurate a model is out of those predicted positive, how many of them are actual positive. The denominator is the Total Predicted Positive.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{TotalPredicatedPositive}$$

Recall:
Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive).

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{TotalActualPositive}$$

F1 Score:
F1 is a function of Precision and Recall. F1 Score is needed when you want to seek a balance between Precision and Recall.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

7.11 MOT Evaluation

7.11.1 Metrics

Typical evaluation format is shown as:
IDF1— IDP— IDR— Rcll— Prcn— FAR— GT— MT— PT— ML— FP— FN— IDs— FM— MOTA— MOTP— MOTAL

The meaning of each alias is
IDF1(ID F1 Score):

$$IDF1 = \frac{2 * IDTP}{2 * IDTP + IDFP + IDFN}$$

IDP(ID Precison):

$$IDP = \frac{IDTP}{IDTP + IDFP}$$

IDR(ID Recall):

$$IDR = \frac{IDTP}{IDTP + IDFN}$$

IDTP:
The longest associated trajectory matching to a groundtruth trajectory is regarded as the gt's true ID.
Then other trajectories matching to this gt is regarded as a 'IDFP'.

Rcll(Recall):
The ratio of TP boxes to GT boxes.

$$Recall = \frac{TP}{TP + FN}$$

Prcn(Precision):
The ratio of TP boxes to all detected boxes.

$$Precision = \frac{TP}{TP + FP}$$

FAR(False Alarm Ratio):
The ratio of FP boxes to frame number.

$$FAR = \frac{FP}{\sum_t 1}$$

GT(Number of GroundtruthTrajectory):
The number of groundtruth trajectories.

MT(Number of Mostly Tracked Trajectory):
The number of trajectories that have over 80% target tracked.

PT(Number of Partially Tracked Trajectory):
The number of trajectories that have 20% to 80% target tracked.

$$PT = GT - MT - ML$$

ML(Number of Mostly Lost Trajectory):
The number of trajectories that have less than 20% target tracked. Total false positive number among all frames.

$$FP = \sum_t \sum_i fp_{i,t}$$

FN(Number of False Negatives):
Total false negative number among all frames.

$$FN = \sum_t \sum_i fn_{i,t}$$

IDs(Number of IDSwitch):
ID switch number, indicating the times of ID jumps.

$$IDs = \sum_t ids_{i,t}$$

FM(Number of Fragmentations):
ID switch is the special case of fragmentation when ID jumps. Fragmentation reflects the continousity of trajectories. When trajectories are determinated, it counts all missed target in each frame.

MOTA(Number of Multiple Object Tracking Accuracy):
A metric reflects the tracking accuracy. It has intergrated consideration of FN, FP, and IDS.

$$T = \sum_t \sum_i gt_{i,t}$$

$$MOTA = 1 - \frac{FN + FP + IDS}{T}$$

MOTP(Number of Multiple Object Tracking Precision):
A metric reflects the tracking precision.

$$MOTP = \frac{\sum_{i,t} IoU_{t,i}}{TP}$$

MOTAL(MOTA Log):

$$MOTAL = 1 - \frac{FN + FP + \log_{10} IDS}{T}$$

Chapter 8

Expected Outcome

8.1 Desktop Outcome Expected

We've expected outcome in following scenario:
Consider, a visually impaired people walking on a street using our system (Desktop App Active). S/he does not know what is going in front of her. But, can listen to voice. Suddenly, a car arrive in front of him. Now, Our system will detect the object that is appeared on camera module. Calculate distance. Then it will give a instruction something like. A car is at a distance 100m coming towards us with the speed of 10m/s. Be Alert, car will arrive within 10s.
Also, Our System will detect any object (Used during training) in front like ditch, dogs, poll and alert people about the environment.
The expected outcome for our Desktop App with GUI for the testing as well as debugging phase in starting of project is as follow:

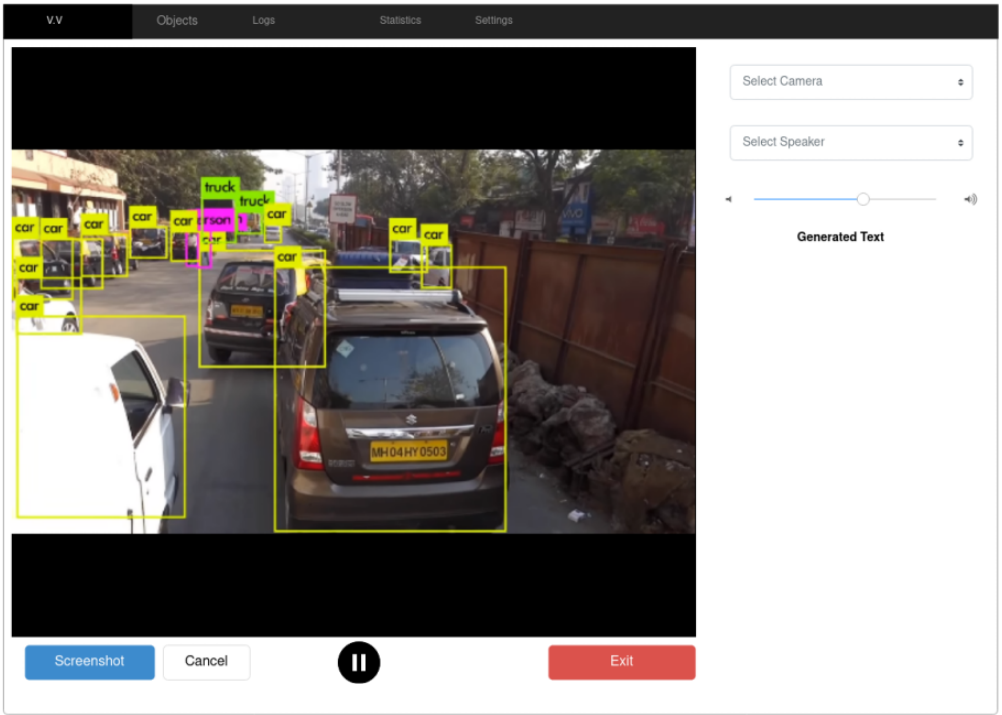


Figure 8.1: Mockup Design main window for desktop App

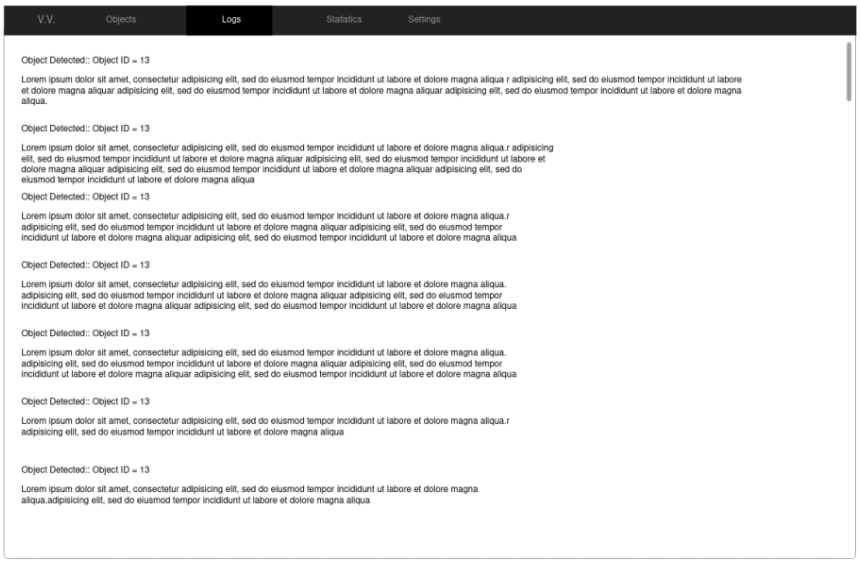


Figure 8.2: Mockup Design Logs window for desktop App

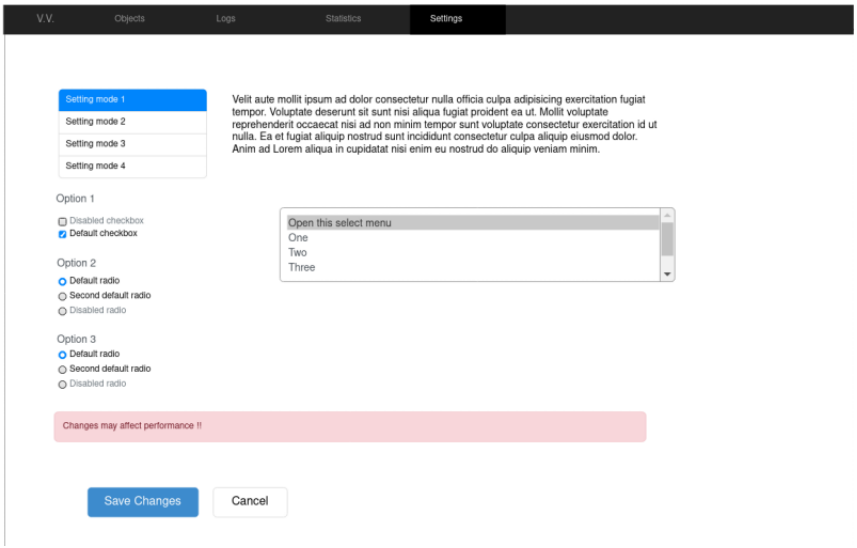
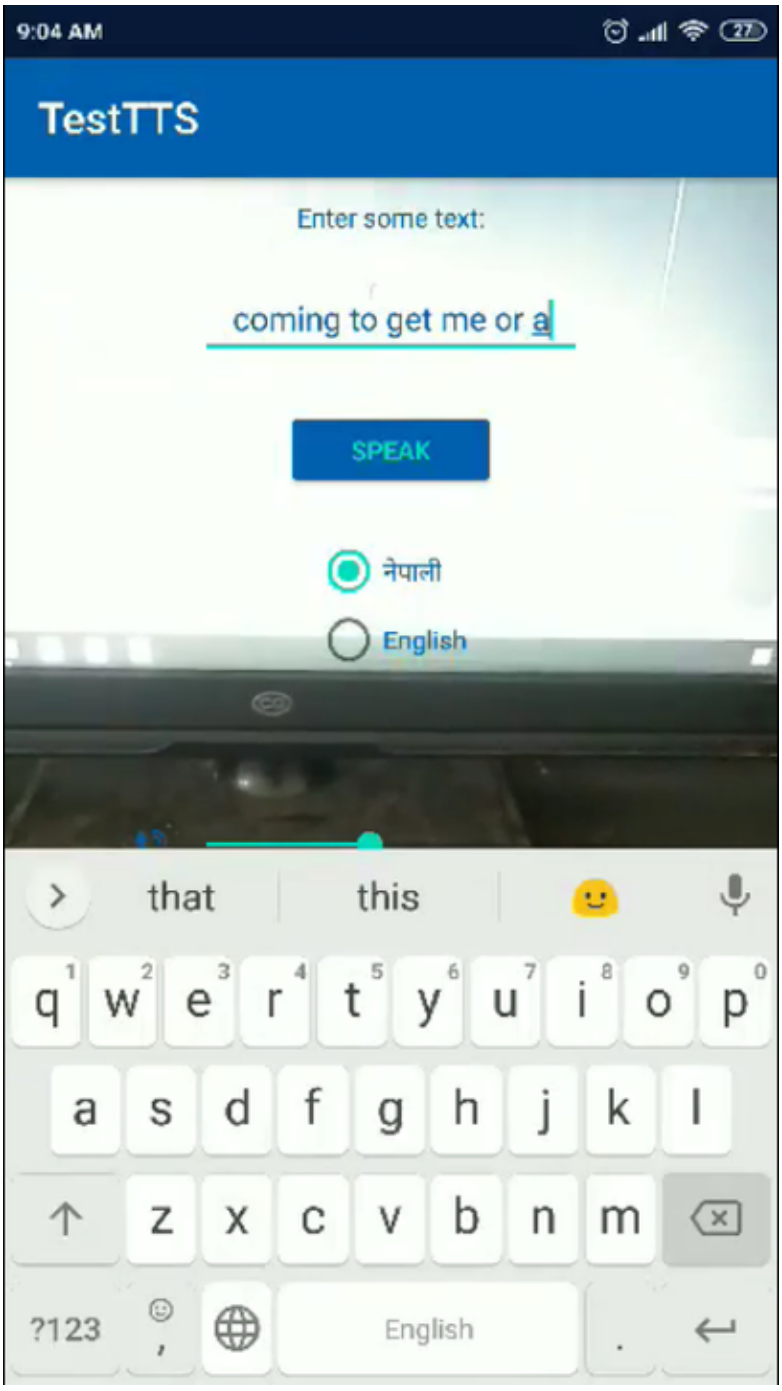


Figure 8.3: Mockup Design Settings window for desktop App

As per our expectations in our desktop app we intended to develop good UI with smooth selection of desired objects to be detected which unfortunately proved to be inefficient interms of performance so the extra tabs were removed afterwards. Also with some logs and statistics tabs we thought of keeping each and every generated texts and detected objects throughout the program.

8.2 Mobile Outcome Expected



Chapter 9

Actual Outcome

9.1 Desktop Outcome

Our Actual outcome is some what inconsistent with what we expected in previous chapter. There was unnecessary lagging of processing and rendering with some of the extra buttons and tabs specified earlier in mockup design so, we decided to remove those from our UI. Actual outcome has become much faster as we implemented multi thread programming to handle our TTS and DCT dillema. As for the screenshot of actual outcome main screen for viewing the ditection is shown below:

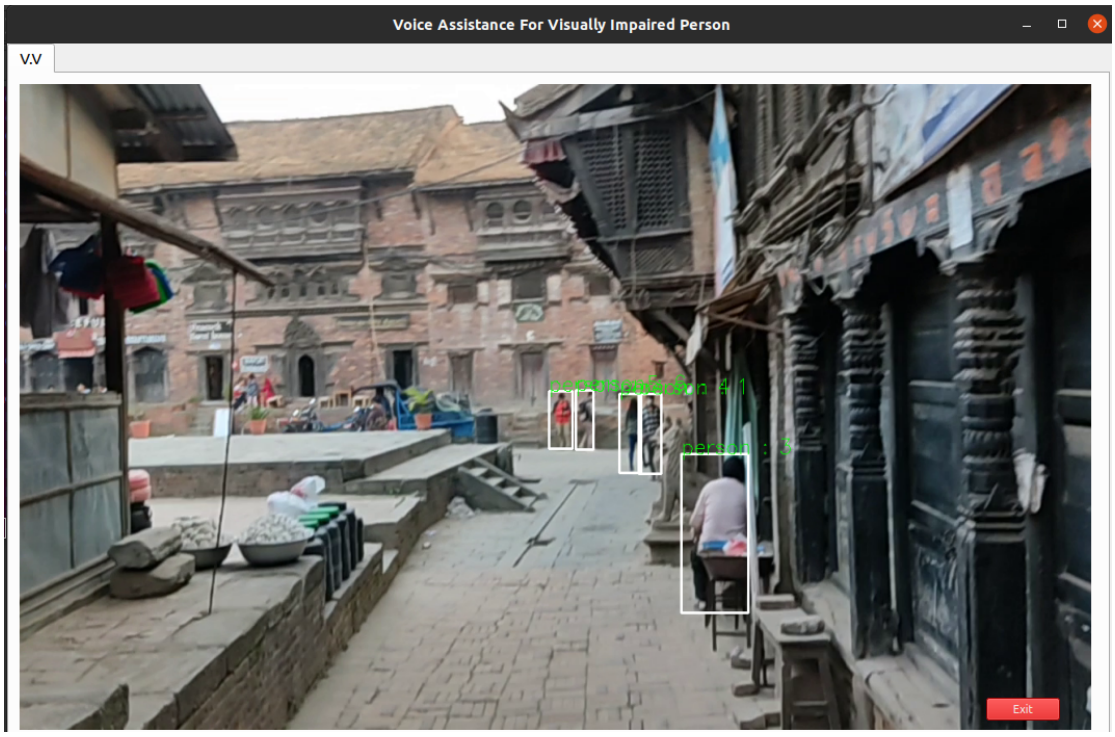


Figure 9.1: Actual outcome of desktop App for dattatraya.mp4

Here, We have only placed preview of each and every frames that has been equipped with bounding boxes and labeled with object name and track ID so that we can verify the detection as well as tracking progress.

9.2 Detection Evaluation Result

Frame No.	TP	FP	FN	Precision	Recall	F1 Score
1	4	0	0	1	1	1
2	4	0	0	1	1	1
3	4	0	0	1	1	1
4	5	0	1	1	0.8333333333	0.9090909091
5	5	0	1	1	0.8333333333	0.9090909091
6	5	0	1	1	0.8333333333	0.9090909091
7	5	0	1	1	0.8333333333	0.9090909091
8	5	0	1	1	0.8333333333	0.9090909091
9	4	1	1	0.8	0.8	0.8
10	4	2	1	0.6666666667	0.8	0.7272727273
11	4	1	2	0.8	0.6666666667	0.7272727273
12	5	1	1	0.8333333333	0.8333333333	0.8333333333
13	4	0	2	1	0.6666666667	0.8
14	4	1	2	0.8	0.6666666667	0.7272727273
15	5	1	1	0.8333333333	0.8333333333	0.8333333333
16	5	1	1	0.8333333333	0.8333333333	0.8333333333
17	4	2	2	0.6666666667	0.6666666667	0.6666666667
18	4	0	1	1	0.8	0.8888888889
19	4	2	1	0.6666666667	0.8	0.7272727273
20	4	0	1	1	0.8	0.8888888889
21	5	1	1	0.8333333333	0.8333333333	0.8333333333
22	3	0	0	1	1	1
23	4	0	0	1	1	1
24	4	0	1	1	0.8	0.8888888889
25	3	1	0	0.75	1	0.8571428571
26	3	0	1	1	0.75	0.8571428571
27	3	0	1	1	0.75	0.8571428571
28	4	0	0	1	1	1
29	3	0	1	1	0.75	0.8571428571
30	5	1	0	0.8333333333	1	0.9090909091
31	4	0	0	1	1	1
32	5	0	0	1	1	1
33	5	0	0	1	1	1
34	5	0	0	1	1	1
35	4	2	1	0.6666666667	0.8	0.7272727273

Frame No.	TP	FP	FN	Precision	Recall	F1 Score
36	5	0	0	1	1	1
37	2	0	2	1	0.5	0.6666666667
38	3	0	1	1	0.75	0.8571428571
39	3	0	1	1	0.75	0.8571428571
40	3	0	0	1	1	1
41	4	0	2	1	0.6666666667	0.8
42	4	0	3	1	0.5714285714	0.7272727273
43	3	1	3	0.75	0.5	0.6
44	3	0	3	1	0.5	0.6666666667
45	3	0	3	1	0.5	0.6666666667
46	4	0	1	1	0.8	0.8888888889
47	3	1	3	0.75	0.5	0.6
48	4	0	2	1	0.6666666667	0.8
49	4	0	3	1	0.5714285714	0.7272727273
50	4	0	3	1	0.5714285714	0.7272727273
51	2	0	4	1	0.3333333333	0.5
52	3	0	3	1	0.5	0.6666666667
53	4	0	2	1	0.6666666667	0.8
54	4	0	2	1	0.6666666667	0.8
55	2	0	1	1	0.6666666667	0.8
56	3	0	1	1	0.75	0.8571428571
57	2	0	1	1	0.6666666667	0.8
58	2	0	1	1	0.6666666667	0.8
59	3	1	0	0.75	1	0.8571428571
60	3	1	0	0.75	1	0.8571428571
61	2	1	1	0.6666666667	0.6666666667	0.6666666667
62	2	0	1	1	0.6666666667	0.8
63	2	0	1	1	0.6666666667	0.8
64	3	0	1	1	0.75	0.8571428571
65	3	0	1	1	0.75	0.8571428571
66	3	0	1	1	0.75	0.8571428571
67	3	0	1	1	0.75	0.8571428571
68	3	0	1	1	0.75	0.8571428571
69	3	0	1	1	0.75	0.8571428571
70	4	0	0	1	1	1
71	5	0	0	1	1	1
72	5	0	0	1	1	1
73	4	0	0	1	1	1
74	5	0	0	1	1	1
75	4	0	0	1	1	1
76	3	0	1	1	0.75	0.8571428571
Total				0.9427631579	0.7898809524	0.8470513405

9.3 MOT Evaluation Result

```
[GT PREPROCESSING]: Removing non-people classes, remaining 92/92 boxes
[TRACK PREPROCESSING]: remove distractors and low visibility boxes,remaining 80/80 computed boxes
Distractors:
[GT PREPROCESSING]: Removing distractor boxes, remaining 92/92 boxes

**************************** dattatraya Evaluation ****************************
IDF1  IDP  IDR  | Rcll  Prcn  FAR   | GT   MT   PT   ML   | FP    FN    IDs   FM    | MOTA  MOTP  MOTAL
 86.0 92.5 80.4 |  80.4  92.5  0.30 |    6    4    2    0 |     6    18     1     0 |  72.8  96.5  73.6


[GT PREPROCESSING]: Removing non-people classes, remaining 175/175 boxes
[TRACK PREPROCESSING]: remove distractors and low visibility boxes,remaining 79/79 computed boxes
Distractors:
[GT PREPROCESSING]: Removing distractor boxes, remaining 175/175 boxes

**************************** dattatraya2 Evaluation ****************************
IDF1  IDP  IDR  | Rcll  Prcn  FAR   | GT   MT   PT   ML   | FP    FN    IDs   FM    | MOTA  MOTP  MOTAL
 60.6 97.5 44.0 |  45.3  97.5  0.10 |   11    4    0    7 |     2    93     0     2 |  44.1  96.1  44.1



**************************** Summary Evaluation ****************************
IDF1  IDP  IDR  | Rcll  Prcn  FAR   | GT   MT   PT   ML   | FP    FN    IDs   FM    | MOTA  MOTP  MOTAL
 70.9 95.0 56.6 |  57.6  95.0  0.20 |   17    8    2    7 |     8   111     1     2 |  54.2  96.3  54.5
```

Figure 9.2: MOT evaluation result

9.4 Mobile Outcome

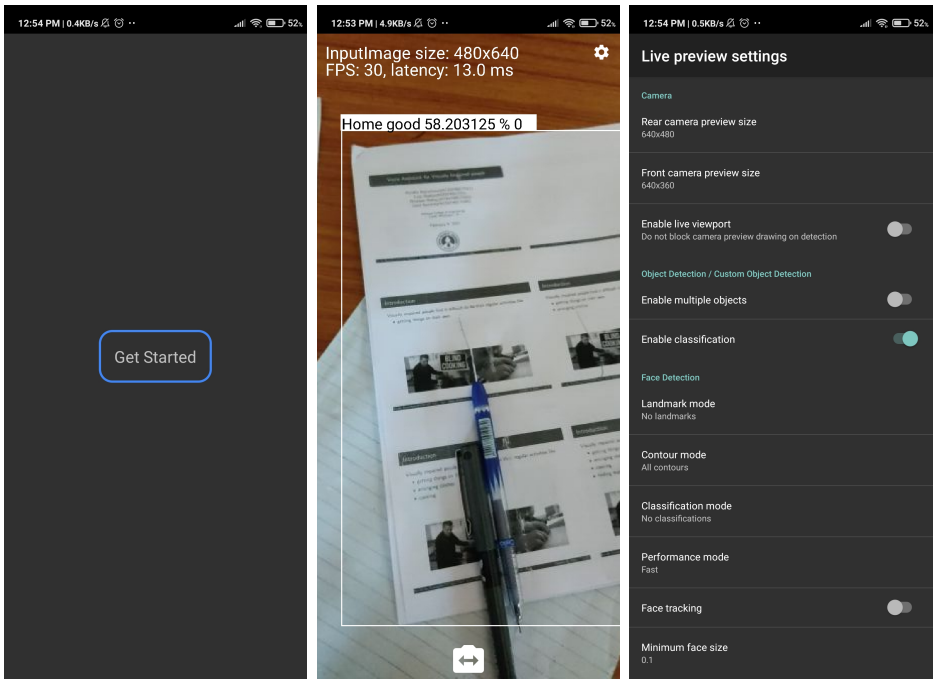


Figure 9.3: Mobile Outcome

Chapter 10

Conclusion and Future Enhancements

In conclusion, Iteration 1 for the project Voice Assistance for the visually impaired people is completed as we succesfully deployed the project as a desktop application. As for Future Ehancements for our project we could implement these:

- Generating our own models using custom dataset.
- A finer classification like for example: people could be subclassed into walking, running, sitting, eating, talking on the phone, jumping, lying on the ground etc.
- Any objects that may of interest to the user like statues, monuments etc.
- Natural speaker can be assigned to TTS module as native speakers may have calm and good supportive accents.
- Use of image captioning or labeling for much more detailed information about surroundings.

Appendix

10.1 Images

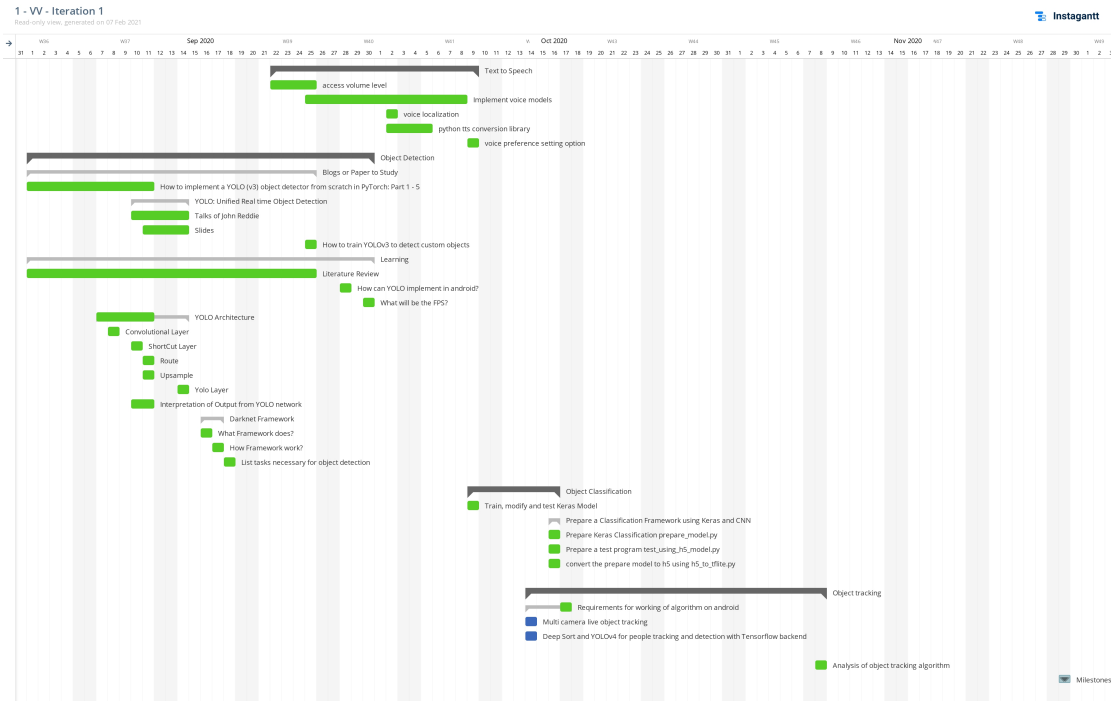


Figure 10.1: Gantt Chart of project for iteration1 made with Instagantt

10.2 Assistance for Running the Project

How to install CUDA 10.1 with cudnn 7.6.5

Do Not Install Nvidia Driver

During CUDA 10 installation, the driver will be installed. So just go ahead and install CUDA

```
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt-get update
sudo apt-get install nvidia-driver-440
```

Reboot here, then run below to verify

```
nvidia-smi
dpkg -l | grep nvidia
```

Install CUDA 10.1

Installation Instruction from Nvidia official website:

```
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804
/x86_64/cuda-ubuntu1804.pin

sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600

wget http://developer.download.nvidia.com/compute/cuda/10.1/Prod/local_
installers/cuda-repo-ubuntu1804-10-1-local-10.1.243-418.87.00_1.0-1_amd64.deb

sudo dpkg -i cuda-repo-ubuntu1804-10-1-local
-10.1.243-418.87.00_1.0-1_amd64.deb

sudo apt-key add /var/cuda-repo-10-1-local-10.1.243-418.87.00/7fa2af80.pub

sudo apt-get update

sudo apt-get -y install cuda
```

Restart Machine

Install cuDNN

Goto page https://developer.nvidia.com/rdp/cudnn-download
Download all three .deb: runtime/developer/code sample

```
sudo dpkg -i libcudnn7_7.6.5.32-1+cuda10.1_amd64.deb
sudo dpkg -i libcudnn7-dev_7.6.5.32-1+cuda10.1_amd64.deb
sudo dpkg -i libcudnn7-doc_7.6.5.32-1+cuda10.1_amd64.deb
```

Now we can verify the cuDNN installation (below is just the official guide, which surprisingly works out of the box):

1. Go to the MNIST example code:

   ```
   cd /usr/src/cudnn_samples_v7/mnistCUDNN/
   ```

2. Compile the MNIST example

   ```
   sudo make clean && sudo make
   ```

3. Run the MNIST example:

   ```
   /mnistCUDNN
   ```

   If your installation is successful, you should see

   ```
   Test passed!
   ```

   at the end of the output.

Deep Sort And YOLO V4

Clone Git Project

```
https://gitlab.com/khwopa1/VV.git
```

Installation Required

To run the project, python3.6 or above required and pip3 with latest version 20.X.X
Upgrade the pip3 version to latest version... (Required)

```
sudo -H pip3 install --upgrade pip
```

To run the required libraries run the install.sh file as:

```
./install.sh
```

Inclusion needed (Optional since file is already converted and stored in model_data)

Download and add yolov4.weights file in model_data folder, then run

```
python3 convert.py
```

This will convert the yolov4 weights file to keras model (.h5 file) The keras model will save in the model_data directory..

Inference

```
python3 main.py
```

Change video name on line 59 in main.py

```
self.filename = '[filename]'
```

Mot-ground-truth

How to use?

```
python3 main.py --data [directory containing images] --dest [filename
to store result]
```

Preprocessing

For this, the image must be stored in a directory and name it like 1.jpg, 2.jpg, ... and do not brake in naming from 1,2,3.....

During Run time

Image will open... Be careful don't click on image,

1. The First image will open with name 1.jpg.
2. There select the rectangle box:
    - First pick the first point.
    - Pick the second point.
    - Then, you will be ask to input track id, there enter the track id.
3. Repeat the 2. step for other object in that image also.
4. Press ESC to get next image.
5. Repeat the step 2 and 3.

MOT-evaluation

How to use?

```
python3 evaluate_tracking.py --seqmap [Path to seqmap file]
--track [Path to Tracking result directory]
--gt [Path to Ground-truth annotation directory]
```

Preprocessing

The sequence map must be update with the required directory name. Generally the directory name will match with the video file name.

Bibliography

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[2] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[3] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[5] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[6] B. Chen, X. Liu, H. Zhao, and J. C. Principe, "Maximum correntropy kalman filter," *Automatica*, vol. 76, pp. 70–77, 2017.

[7] O. Sheremet, "Intersection over union (iou) calculation for evaluating an image segmentation model," Jul 2020. [Online]. Available: https://towardsdatascience.com/intersection-over-union-iou-calculation-for-evaluating-an-image-segmentation-model-8b22e2

[8] S. K, "Non-maximum suppression (nms)," Oct 2019. [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c