

Final Report

FLEX LENS

A PROJECT REPORT

Submitted by

Rohan Ghosh (3180700554102)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & BIOMEDICAL ENGINEERING

Guru Jambheshwar University of Science & Technology

December 2020

BONAFIDE CERTIFICATE

Certified that this project report “**FLEX LENS**” is the bonafide work of
“**ROHAN GHOSH (3180700554102)**”, carried out the project work under my/our
supervision.

<<Signature of the Head of the Department>>

SIGNATURE

SUPERVISOR

<<Academic Designation>>

<<Department>>

<<Signature of the Supervisor>>

SIGNATURE

<<Name>>

**HEAD OF THE
DEPARTMENT**

<<Department>>

Submitted for the project viva-voce examination held on 01.12.2020

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

On this great occasion of accomplishment of my project FLEX LENS, I would like to sincerely express my gratitude to my project supervisor and the head of department who have supported through the completion of this project. Whenever I faced any trouble, they were there for me.

TABLE OF CONTENT

Contents

ACKNOWLEDGEMENT	3
ABSTRACT	6
GRAPHICAL ABSTRACT	7
.....	7
CHAPTER I	8
1.1 INTRODUCTION:	8
CHAPTER II LITERATURE SURVEY	11
3.1 CHARACTERISTIC IDENTIFICATION:	13
Objective	13
Single Entity	13
Lifespan.....	13
Require Funds	13
Life Cycle	13
Team Spirit	14
Risk and Uncertainty	14
Directions	14
Uniqueness	15
Flexibility	15
Sub-contracting	15
Cost	15
3.2 CONSTRAINTS IDENTIFICATION:	16
Time	16
Cost	16
Scope	16
Quality.....	16
Benefits	16
Risk	16
3.3 DESIGN FLOW DIAGRAMS:	17
DFD Model 1:	17
DFD Model 2:	18
DFD Model 3:	19
ER- Diagram:	20
Class Diagram:	21
CHAPTER IV	22
4.1 Various Technology used:	22
Software and Libraries Requirement:	22

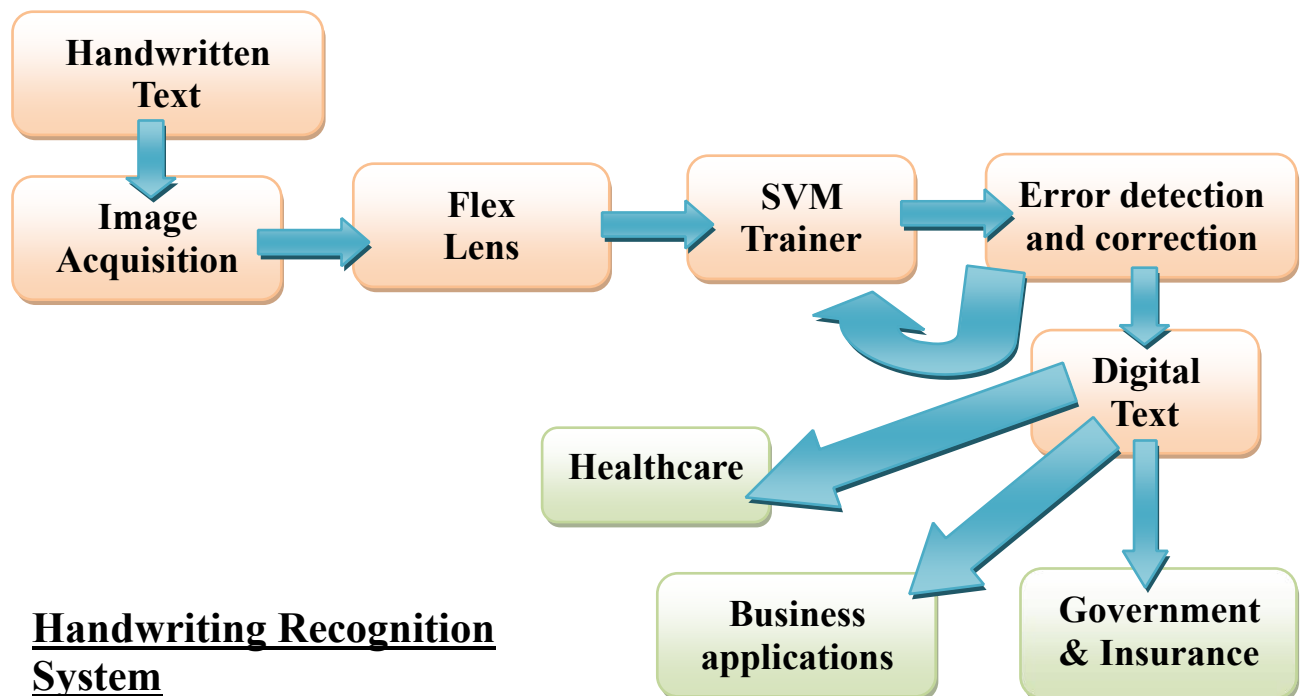
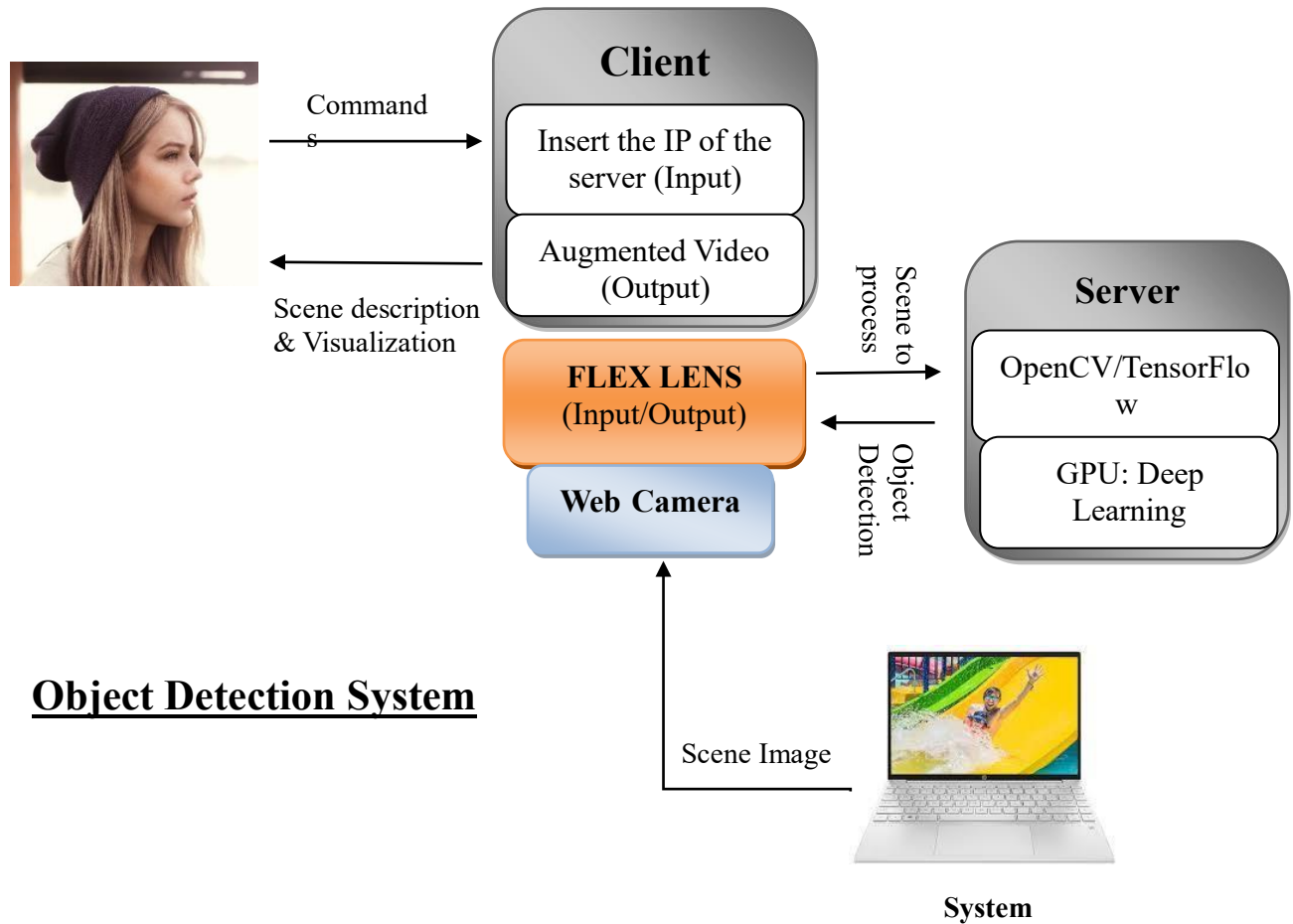
1. Anaconda navigator:	22
2. Jupyter Notebook:	23
3. Tensorflow:	23
4. Matplotlib:.....	23
5. Numpy:	23
CHAPTER V	24
5.1 CONCLUSION:	
.....	24
5.2 FUTURE	SCOPE:
.....	25
5.3 USER	MANUAL:
.....	27
1. Step wise guide for object detection system	27
2. Step wise guide for handwriting recognition system	30

ABSTRACT

Object detection is one of the most basic and central tasks in computer vision. Its task is to find all the interested objects in the image, and determine the category and location of the objects. Object detection is widely used and has strong practical value and research prospects. Applications include face detection, pedestrian detection and vehicle detection. In recent years, with the development of convolutional neural network, significant breakthroughs have been made in object detection. This paper describes in detail the classification of object detection algorithms based on deep learning. The algorithms are mainly divided into onstage object detection algorithm and two-stage object algorithm, and the general data sets and performance indicators of object indicators.

Handwritten character recognition (HCR) is the detection of characters from images, documents and other sources and changes them in machine-readable shape for further processing. The accurate recognition of intricate-shaped compound handwritten characters is still a great challenge. Recent advances in convolutional neural network (CNN) has made great progress in HCR by learning discriminatory characteristics from large amounts of raw data. In this paper, CNN is implemented to recognize the characters from a test dataset. The main focus of this work is to investigate CNN capability to recognize the characters from the image dataset and the accuracy of recognition with training and testing.

GRAPHICAL ABSTRACT



CHAPTER I

1.1 INTRODUCTION:

Computer Vision is the branch of the science of computers and software systems which can recognize as well as understand images and scenes. Computer Vision is consisting of various aspects such as image recognition, object detection, image generation, image super-resolution and many more. Object detection is widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and self-driving cars. In this project, we are using highly accurate object detection-algorithms and methods such as R-CNN, Fast-RCNN, Faster-RCNN, RetinaNet and fast yet highly accurate ones like SSD and YOLO. Using these methods and algorithms, based on deep learning which is also based on machine learning require lots of mathematical and deep learning frameworks understanding by using dependencies such as TensorFlow, OpenCV, imageAi etc., we can detect each and every object in image by the area object in a highlighted rectangular box and identify each and every object and assign its tag to the object. This also includes the accuracy of each method for identifying objects

Handwriting digits and character recognitions have become increasingly important in today's digitized world due to their practical applications in various day to day activities. It can be proven by the fact that in recent years, different recognition systems have been developed or proposed to be used in different fields where high classification efficiency is needed. Systems that are used to recognize Handwriting letters, characters, and digits help people to solve more complex tasks that otherwise would be time-consuming and costly. A good example is the use of automatic processing systems used in banks to process bank cheques. Without automated bank cheque processing systems, the bank would be required to employ many employees who may not be as efficient as the computerized processing system. The handwriting recognition problem can be considered for various alphabets and at various levels of abstraction. The main goal of the work presented in this paper has been the development of an on-line handwriting recognition system which is able to recognize handwritten characters of several different writing styles. This approach has succeeded in having robust pattern recognition features, while maintaining feature's domain space to a small, optimum quantity. Back propagation neural network (BPN) technique has been used as classifier and recognition rate up to 87% has been achieved even for highly distorted hand written characters.

1.2 PROBLEM DEFINITION:

Iterating over the problem of localization plus classification we end up with the need for detecting and classifying multiple objects at the same time. Object detection is the problem of finding and classifying a variable number of objects on an image. The important difference is the “variable” part. In contrast with problems like classification, the output of object detection is variable in length, since the number of objects detected may change from image to image. In this post we’ll go into the details of practical applications, what are the main issues of object detection as a machine learning problem and how the way to tackle it has been shifting in the last years with deep learning.

Handwriting character recognition is one of the research fields in computer vision, artificial intelligence, and pattern recognition. A computer application that performs handwriting recognition can be argued to have the ability to acquire and detecting characters in pictures, paper documents, and other sources and convert them into electronic format or machine-encoded form. The system may obtain Handwriting sources from a piece of paper through optical scanning or intelligent word recognition. Also, the system may be designed to detect the movement of the pen tip on the screen. In other words, handwriting recognition may involve a system detecting movements of a pen tip on the screen to get a clue of the characters being written.

Handwriting recognition can be classified into two:

- **Offline recognition**

Offline handwriting recognition involved the extraction of text or characters from an image to have letter codes that can be used within a computer. It involves obtaining digital data from a static representation of handwriting. A system is provided with a Handwriting document to read and convert the handwriting to a digital format.

- **Online recognition**

Online handwriting recognition, on the other hand, involved automatic detection or conversion of characters as they are written on the specialized screen. In this case, the system sensors movement of pen-tip to detect characters and words. Different methods and techniques are used to ensure that computer systems can read characters from Handwriting images and documents.

Among the existing techniques that are used to model, and train Handwriting character recognition include neural network, Hidden Markov Model (HMM), Machine Learning, and Support Vector

Machine, to mention a few. This paper focuses on artificial intelligence networks, machine learning, Hidden Markov Model, and the Support Vector Machine.

CHAPTER II

LITERATURE SURVEY

Deep convolutional neural networks recently demonstrated very impressive performance on a number of image recognition benchmarks. It has shown good performance on large scale visual recognition challenge. It was a winning model on localization subtask with the process by predicting single bounding box and identifying object category in the image. But the model cannot handle multiple instances of same object in the image. But the model cannot handle multiple instances of same object in the image. But now it can handle the same image having multiple instances and allows cross class generalization at highest level of network. In this paper the computational challenge is addressed. Also, this challenge becomes even harder when an object occurs more than once in the image. How they tackle this by generating a no of bounding boxes. For each box the output is a confidence score i.e., the likelihood of an image existing in that box. Various training exercises are performed for this. The predicted results and the real results are then matched for the learning purpose. They use the Deep Neural Network which produces a fixed number of bounding boxes and then gives the output of each box as a confidence score.

Recurrent neural network (RNN) based language model (RNNLM) is a biologically inspired model for natural language processing. It records the historical information through additional recurrent connections and therefore is very effective in capturing semantics of sentences. At architectural level the parallelism of RNN training scheme is improved and also reduces the computing resource requirement. Experiments at different network sizes demonstrates a great scalability of proposed framework. RNN is a different type of neural network that can operate in time domain. RNN captures the long-range dependencies using the additional recurrent connection. Then it stores them in hidden layer for later use. But the training costs in RNN was really high. So hardware up gradation was necessary to make it feasible. FPGA based accelerators have really caught the attention for tackling this problem. Modern language models are based on statistical analysis. Thus, a proper balance between computation unit and memory bandwidth should be obtained by proper scalability. Next comes the architectural optimization. It has various things to do in it. Like to increase parallelism between output layers and hidden layers. But it can only be done to a certain extent as there are limitations to it. Then there are the hardware implementations. The FPGA hardware design plays a huge role in supporting RNN

Online handwriting recognition online handwriting recognition, where existing challenges are to cope with problems of various writing fashions, variable size for the same character, different stroke orders for the same letter, and efficient data presentation to the classifier. The similarities of distinct character shapes and the ambiguous writing further complicate the dilemma. A solitary solution of all these problems lies in the intelligent and appropriate extraction of features from the character at the time of writing. A typical handwriting recognition system focuses on only a subset of these problems.

CHAPTER III

3.1 CHARACTERISTIC IDENTIFICATION:

Objective

- The main purpose of object detection is to identify and locate one or more effective targets from still image or video data. It comprehensively includes a variety of important techniques, such as image processing, pattern recognition, artificial intelligence and machine learning.
 - It has broad application prospects in such areas such as road traffic accident prevention, warnings of dangerous goods in factories, military restricted area monitoring and advanced human–computer interaction.
 - Since the application scenarios of multi-target detection in the real world are usually complex and variable, balancing the relationship between accuracy and computing costs is a difficult task
- Single Entity**
- Execution of the undertaking with a little group is a quite extreme errand, yet we figured out how to keep the work underway by partitioning the venture in modules and teaming up and guaranteeing the task is finished as a Single Entity

Lifespan

- The project is divided into two parts i.e., “Object detection system” as the 1st part and “Handwriting recognition system” as the 2nd part. Hence, time period of 1 month is required for completion of the process

Require Funds

- The project is a combined product of hardware and programmed interface.
The cost depends on the number of cameras and the miscellaneous circuitry.
Thus, the setup cost of 1 camera set is 10,000/- INR.

Life Cycle

- The undertaking lifecycle began with formulating out the thought for the task.
- After effectively concluding the task, our group began to design out how the beginning stage will be executed.

- The significant terms in regards to the task were characterized and we began with the plan interaction.
- Subsequent to finishing the planning of the undertaking, the task was worked in around multi month time, being tried for any issues and afterward it will be sent as the last advance of the improvement lifecycle.

Team Spirit

- All through the improvement interaction, our group crossed numerous snags to accomplish the ideal outcomes. At times it took us days to determine a single bug which was very baffling as the advancement ended. In any case, keeping our expectations high and keeping us loaded up with enthusiasm and inspiration, we continued pushing ahead beating every one of the obstructions and effectively finished the undertaking

Risk and Uncertainty

- Project advancement represents a ton of hazard and vulnerabilities which should be dealt with during the improvement interaction. Some of them include:
 - Virtual environment crash while setting it up using Jupyter notebook in Anaconda Navigator.
 - Image and video lagging during testing period leading lead to creation of new start from the scratch to detect bugs.
 - Hardware and software incompatibility due to version differences.
 - Financial losses due to purchase of different sets of cameras for software compatibility and efficient functioning tests.
- Keeping a note of this large number of dangers, we worked interminably to adapt to this multitude of vulnerabilities and thought of an answer which limits the above angles. **Directions**
- In the wake of fostering the model of the setup, we took client input about what the machine needs and where it required a few alterations.
- In view of the inputs got, we worked in the headings given by them to construct a machine that welcomes clients.

Uniqueness

- The algorithm used in our machine setup and the component compatibility gives our machine an edge and defines its uniqueness and further workforce for a better, smarter and efficient machine setup will lead us to patent acquisition with ease.
- Our team has trained the AI with numerous test subjects leading to 4-digit testing series.
- With upgradation in the interface and alteration won't lead compromise in its functionality, because we have assigned two members of our team for its testing.

Flexibility

- The task has been planned so that it is generally open to changes as expected by the client. The task is adaptable as far as utilization and changes can be made progressively whenever required **Sub-contracting**

- Our group has complete information about each progression of the undertaking improvement cycle and needed exceptionally less external support. However, we took help from our co-supervisor who directed us to make this venture a triumph. **Cost**

- As referenced about the expense expected for the undertaking of 1 camera setup, regardless of whether the venture goes through certain changes, the cost won't be affected a lot.
- Need of number of camera setup is based on the field of usage.

3.2 CONSTRAINTS IDENTIFICATION:

Time

- During the arranging period of the advancement lifecycle, we zeroed in on how long we would expect for project completion.

- This guaranteed that we worked inside the time constraints. **Cost**

- While sorting out the funds expected to effectively complete the venture, we put forth a most extreme line we can spend for the undertaking and guaranteed that regardless of whether additional assets are required, the cost constraints won't be impacted. **Scope**

- As currently referenced in the venture plan, our undertaking is fairly interesting and many haven't planned anything like this. Therefore, the scope constraints preclude from the conversation as we did a very great job by the decision of the venture

Quality

- Our group has made an honest effort to hold the nature of the venture within proper limits. Rest, the client input will permit us to make alterations as required monitoring the quality constraints

Benefits

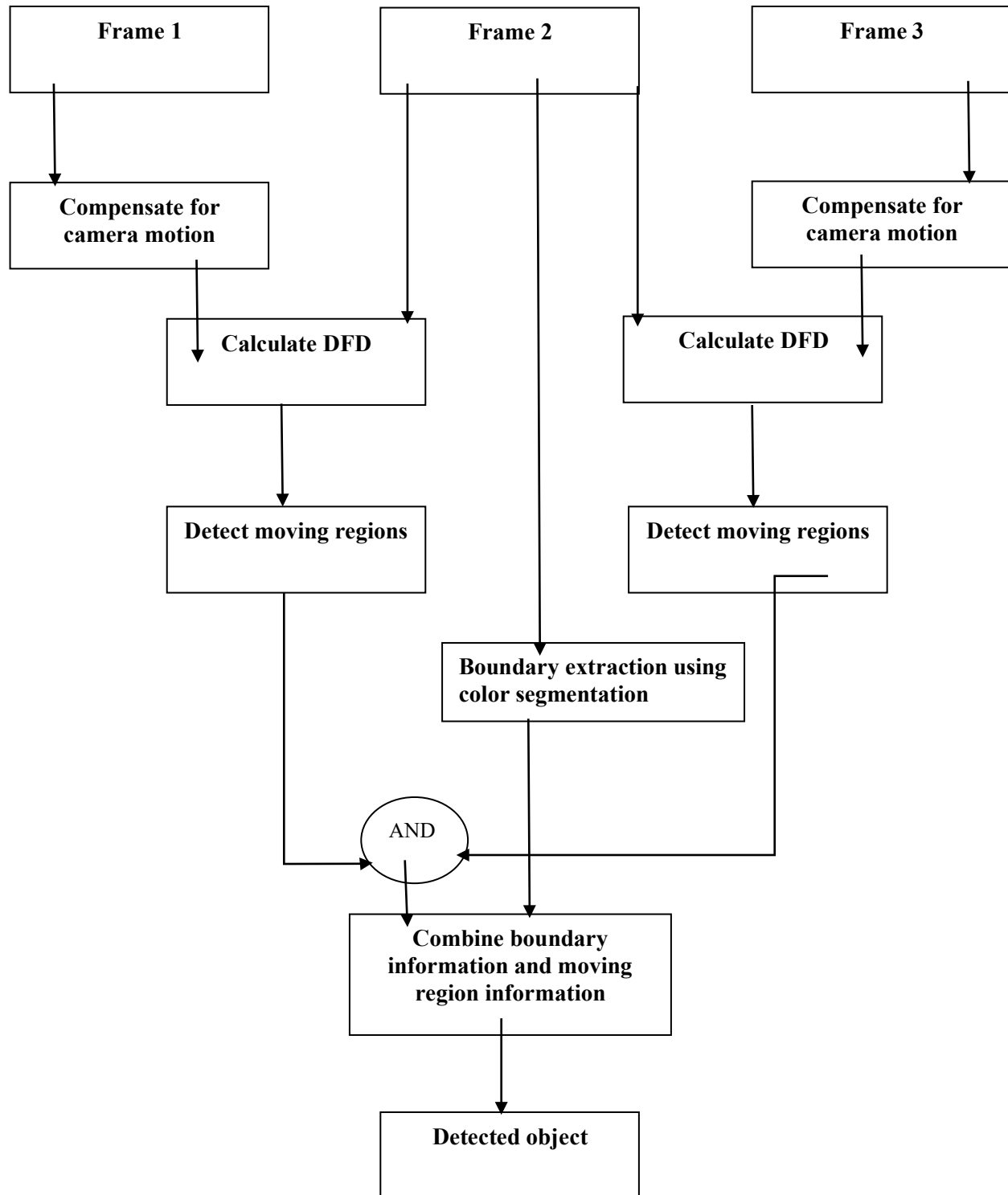
- Being one of a kind in its own particular manner, it has a ton of advantages. The advantages of the application don't appear to stop just like a stage which is connected with endlessly learning is a continuous cycle in any individual's life

Risk

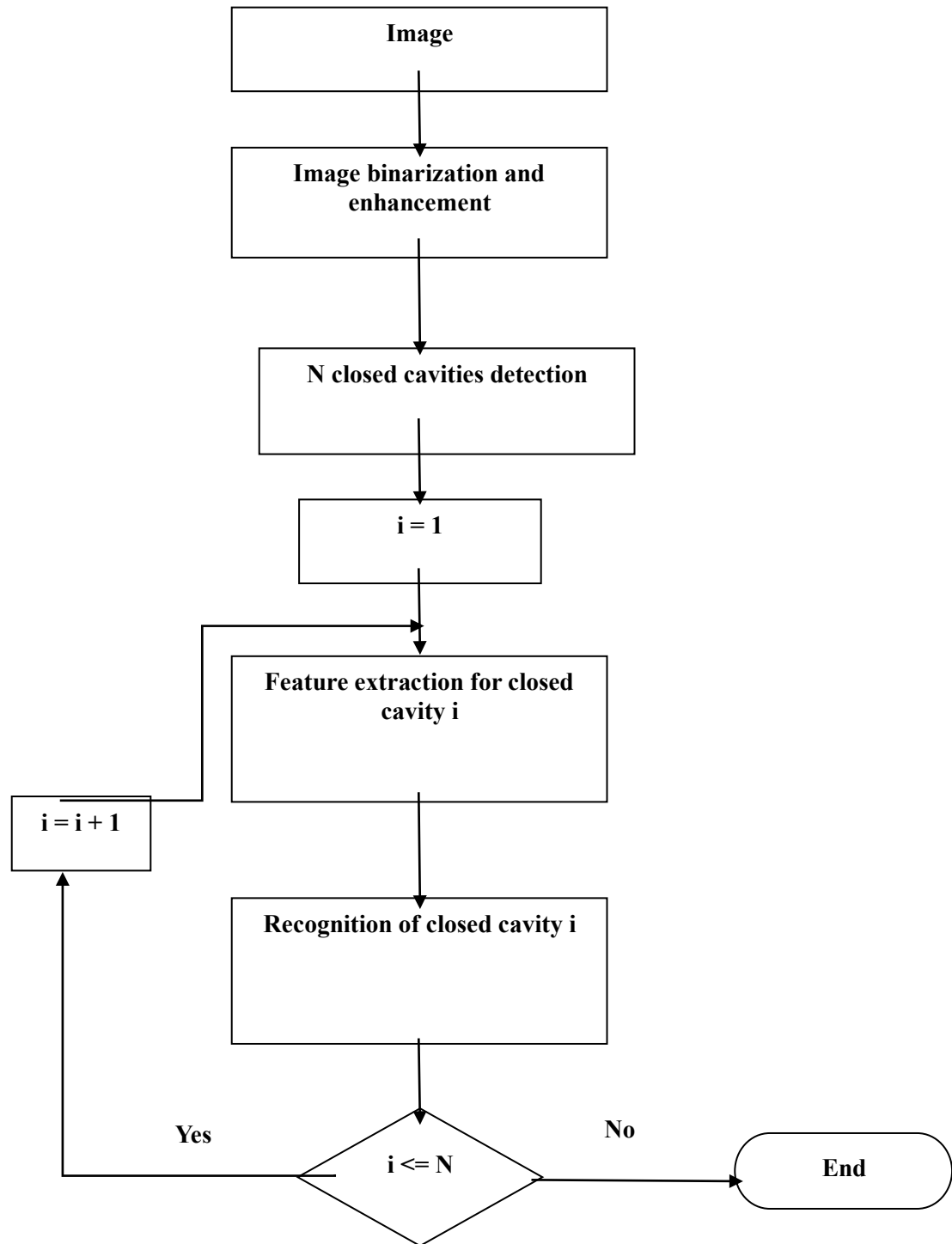
- As currently talked about, the improvement of an undertaking offers a great deal of dangers that might be of some value, however holding them within proper limits is what we truly need to focus on.

3.3 DESIGN FLOW DIAGRAMS:

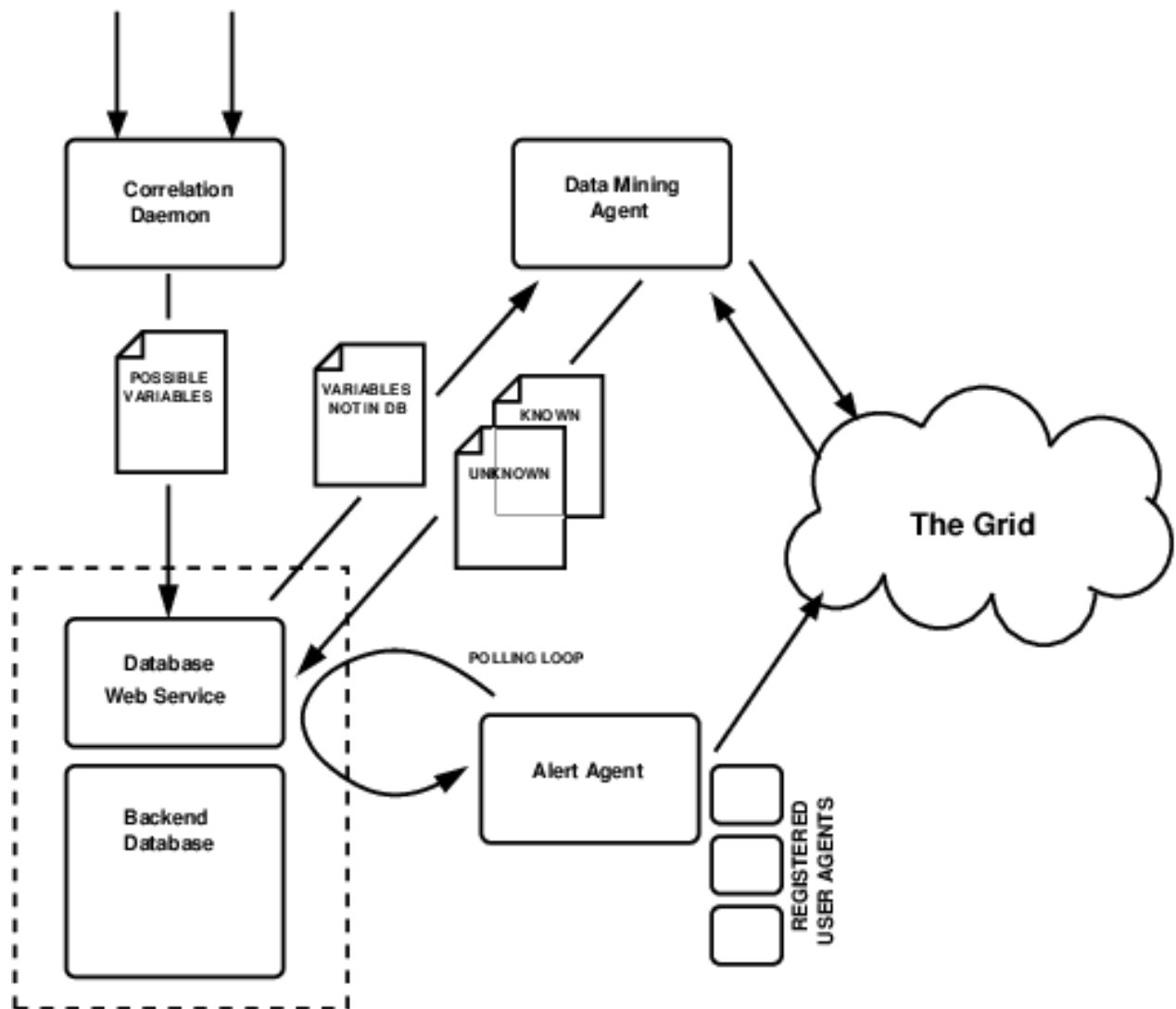
DFD Model 1:



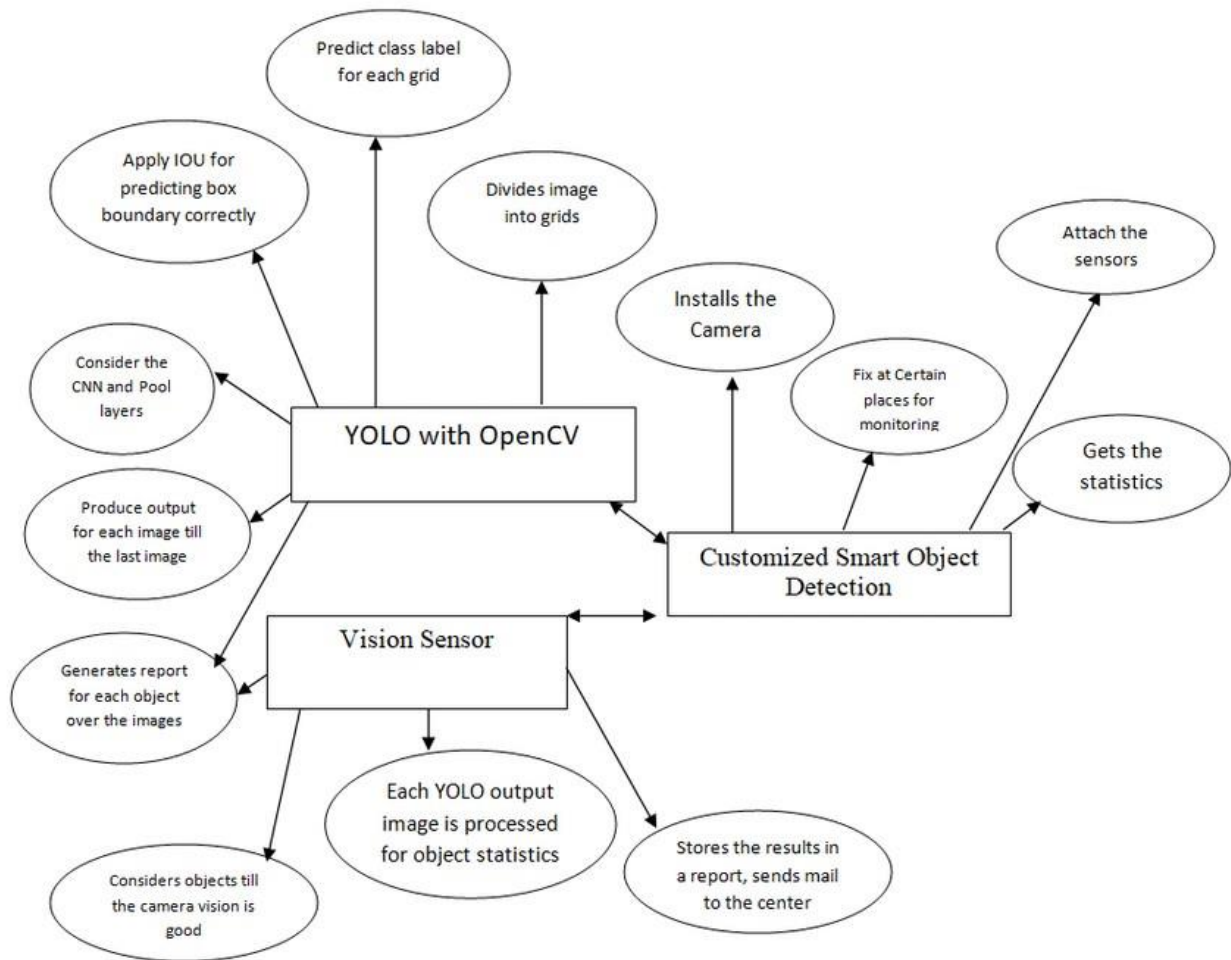
DFD Model 2:



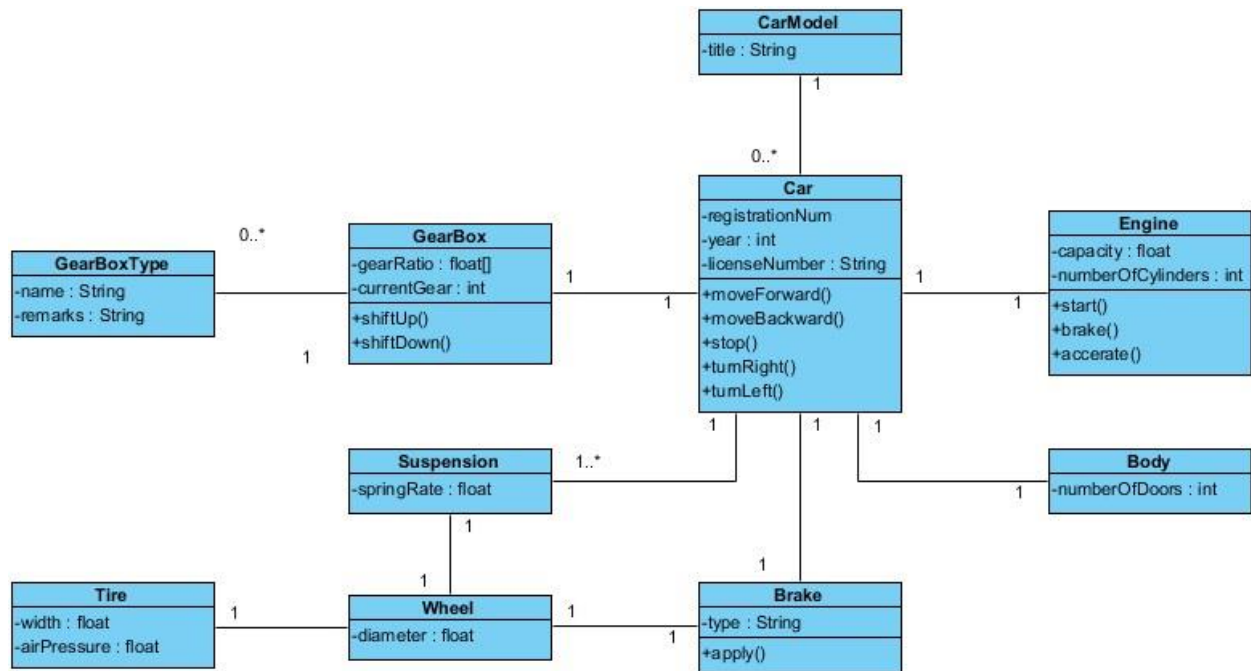
DFD Model 3:



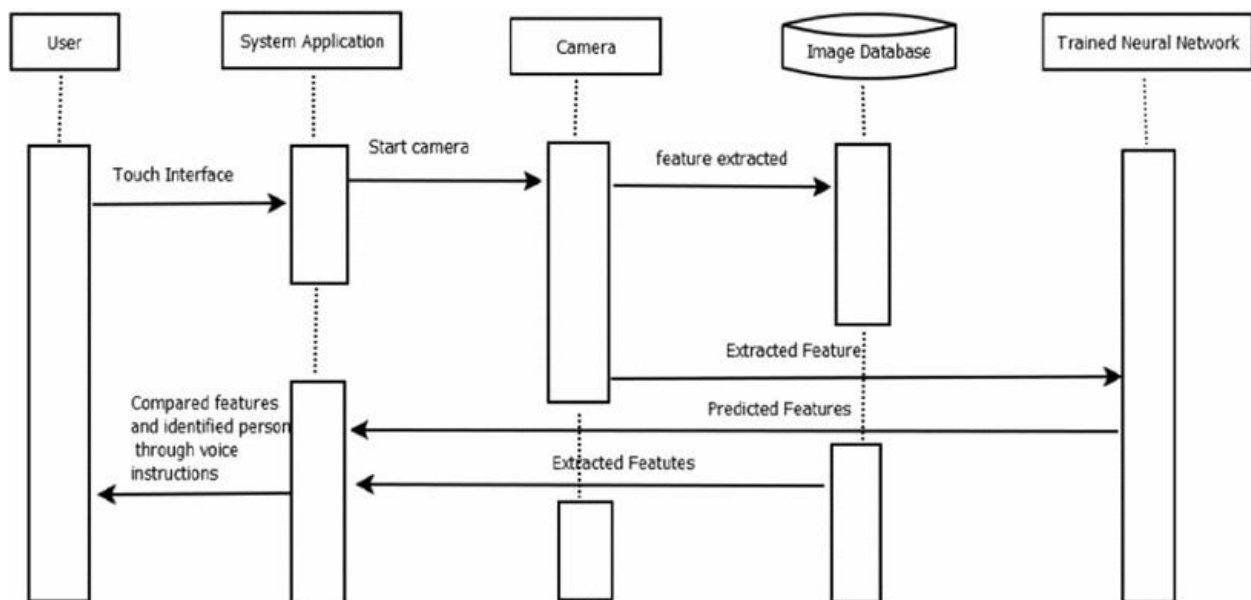
ER- Diagram:



Class Diagram:



Sequence Diagram:



CHAPTER IV

4.1 Various Technology used:

Computer Vision is the branch of the science of computers and software systems which can recognize as well as understand images and scenes. Computer Vision is consisting of various aspects such as image recognition, object detection, image generation, image super-resolution and many more. Object detection is widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and self-driving cars.

Handwriting digits and character recognitions have become increasingly important in today's digitized world due to their practical applications in various day to day activities. It can be proven by the fact that in recent years, different recognition systems have been developed or proposed to be used in different fields where high classification efficiency is needed. Systems that are used to recognize Handwriting letters, characters, and digits help people to solve more complex tasks that otherwise would be time-consuming and costly.

Software and Libraries Requirement:

- Python 3.7.2
- Anaconda navigator
- Jupyter Notebook
- Tensorflow
- Matplotlib
- Numpy

1. Anaconda navigator:

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using commandline commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

2. Jupyter Notebook:

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

3. Tensorflow:

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

4. Matplotlib:

Matplotlib is a Python programming language plotting library and its NumPy numerical math extension. It provides an object-oriented API to use general-purpose GUI toolkits such as Tkinter, wxPython, Qt, or GTK+ to embed plots into applications.

`pip install matplotlib` – command

5. Numpy:

NumPy is library of Python programming language, adding support for large, multi-dimensional array and matrices, along with large collection of highlevel mathematical function to operate over these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several developers. In 2005 Travis Olphant created NumPy by incorporating features of computing Num array into Numeric, with extension modifications. NumPy is open-source software and has many contributors.

`pip install numpy` -command

CHAPTER V

5.1 CONCLUSION:

Deep-learning based object detection has been a search hotspot in recent years. This project starts on generic object detection pipelines which give base architectures for other related tasks. With the assistance of this the 3 other common tasks, namely object detection, face detection and pedestrian detection, are often accomplished. Authors accomplished this by combining 2 things: Object detection with deep learning and OpenCV and Efficient, threaded video streams with OpenCV. The camera sensor noise and lightening condition can change the result because it can create problem in recognizing the objects. generally, this whole process requires GPU's rather than CPU's. But we've done using CPU's and executes in much less time, making it efficient. Object Detection algorithms act as a mixture of both image classification and object localization. It takes the given image as input and produces the output having the bounding boxes adequate to the number of objects present within the image with the category label attached to every bounding box at the highest. It projects the scenario of the bounding box up the shape of position, height and width.

5.2 FUTURE SCOPE:

The proposed feature extraction method helps to select the important feature. To improve the efficiency of the classifier, the number of features should be less in number. Specifically, the contributions towards this research work are as follows,

- An object recognition system is developed, that recognizes the two-dimensional and three-dimensional objects.
- The feature extracted is sufficient for recognizing the object and marking the location of the object. x the proposed classifier is able to recognize the object in less computational cost.
- The proposed global feature extraction requires less time, compared to the traditional feature extraction method.
- The performance of the SVM-kNN is greater and promising when compared with the BPN and SVM.
- The performance of the One-against-One classifier is efficient

As a scope for future enhancement,

- Features either the local or global used for recognition can be increased, to increase the efficiency of the object recognition system.
- Geometric properties of the image can be included in the feature vector for recognition.
- Using unsupervised classifier instead of a supervised classifier for recognition of the object.
- The proposed object recognition system uses grey-scale image and discards the color information.

The color information in the image can be used for recognition of the object. Color based object recognition plays vital role in Robotics

Although the visual tracking algorithm proposed here is robust in many of the conditions, it can be made more robust by eliminating some of the limitations as listed below:

- In the Single Visual tracking, the size of the template remains fixed for tracking. If the size of the object reduces with the time; the background becomes more dominant than the object being tracked. In this case the object may not be tracked.
- Fully occluded object cannot be tracked and considered as a new object in the next frame.
- Foreground object extraction depends on the binary segmentation which is carried out by applying threshold techniques. So, blob extraction and tracking depend on the threshold value.
- Splitting and merging cannot be handled very well in all conditions using the single camera due to the loss of information of a 3D object projection in 2D images
- For Night time visual tracking, night vision mode should be available as an inbuilt feature in the CCTV camera.

To make the system fully automatic and also to overcome the above limitations, in future, Multiview tracking can be implemented using multiple cameras. Multi view tracking has the obvious advantage over single view tracking because of wide coverage range with different viewing angles for the objects to be tracked.

5.3 USER MANUAL:

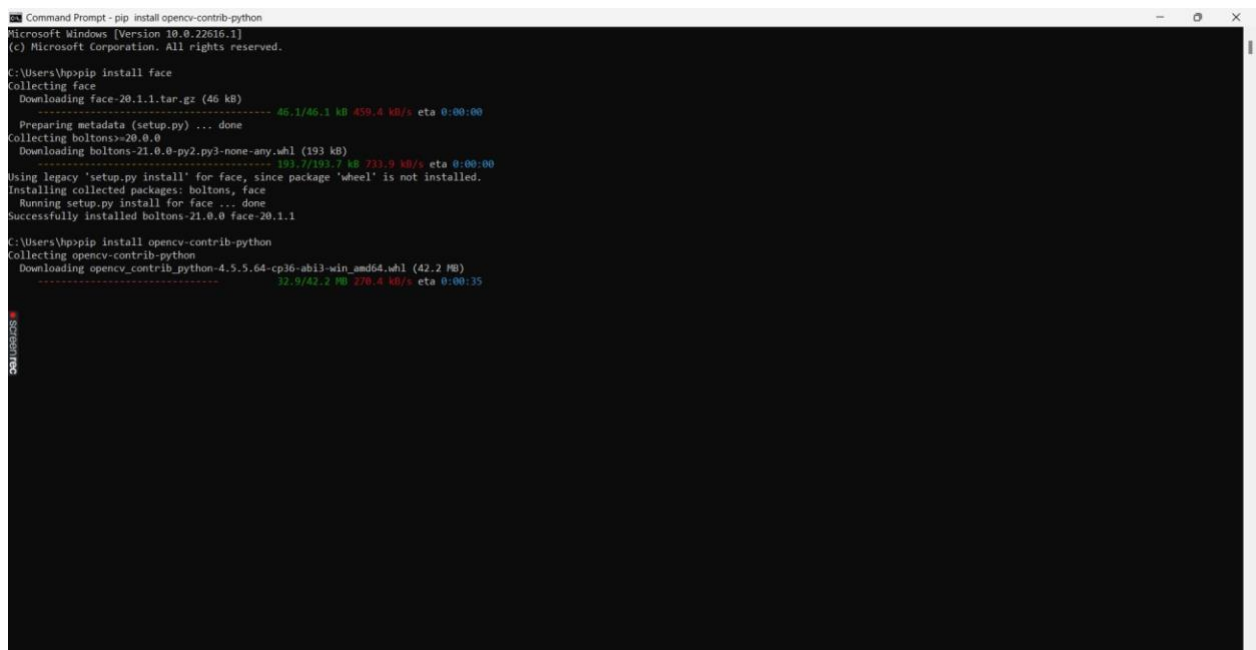
1. Step wise guide for object detection system

Step-1

- Open the command prompt and type the below statements and download the packages.

\$pip install face

\$opencv-contrib-python



```
Command Prompt - pip install opencv-contrib-python
Microsoft Windows [Version 10.0.22616.1]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>pip install face
Collecting face
  Downloading face-20.1.1.tar.gz (46 kB)
    ----- 46.1/46.1 kB 459.4 kB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting boltzons>=20.0.0
  Downloading boltzons-21.0.0-py2.py3-none-any.whl (193 kB)
    ----- 193.7/193.7 kB 733.9 kB/s eta 0:00:00
Using legacy 'setup.py install' for face, since package 'wheel' is not installed.
Installing collected packages: boltzons, face
  Running setup.py install for face ... done
Successfully installed boltzons-21.0.0 face-20.1.1

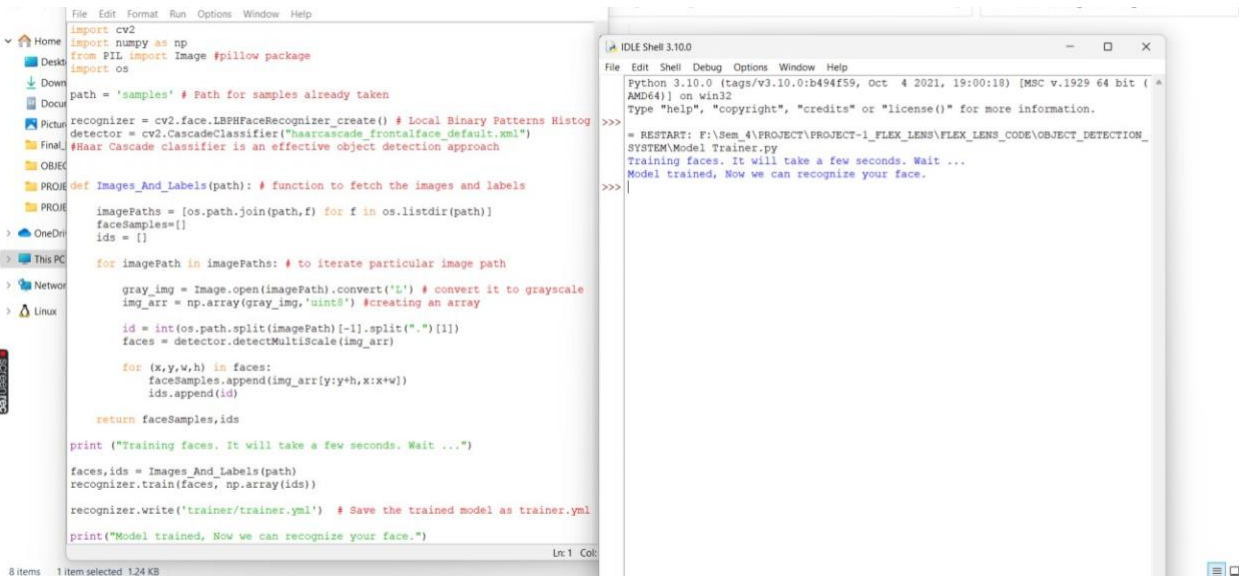
C:\Users\hp>pip install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading opencv_contrib_python-4.5.5.64-cp36-abi3-win_umd64.whl (42.2 MB)
    ----- 32.9/42.2 MB 270.4 kB/s eta 0:00:35
```

Step-2

- Open the file saved as “Sample Generator.py”
- Click on “Run”
- Click on “Run module”
- Pic samples get captured and stored in the folder named as “samples”

Step-3

- Open the file saved as “Model Trainer.py”
- Click on “Run”
- Click on “Run module”



Step-4

- Open the file saved as “Face recognition.py”
- The dialog box with code appears.
- Enable the camera and press run.

```
File Edit Selection View Go Run Terminal Help Face recognition.py - Visual Studio Code
Face recognition.py x
Camera off
Face recognition.py > {} cv2

3 recognizer = cv2.face.LBPHFaceRecognizer_create() # Local Binary Patterns Histograms
4 recognizer.read('trainer/trainer.yml') #load trained model
5 cascadePath = "haarcascade_frontalface_default.xml"
6 faceCascade = cv2.CascadeClassifier(cascadePath) #initializing haar cascade for object detection approach
7
8 font = cv2.FONT_HERSHEY_SIMPLEX #denotes the font type
9
10
11 id = 2 #number of persons you want to Recognize
12
13
14 names = ['', 'LIZA SHARMA'] #names, leave first empty bcz counter starts from 0
15
16
17 cam = cv2.VideoCapture(0, cv2.CAP_DSHOW) #cv2.CAP_DSHOW to remove warning
18 cam.set(3, 640) # set video FrameWidth
19 cam.set(4, 480) # set video FrameHeight
20
21 # Define min window size to be recognized as a face
22 minW = 0.1*cam.get(3)
```

PROBLEMS OUTPUT TERMINAL

[Running] python -u "d:\CV\Face-Recognition-main\Face recognition.py"

Thanks for using this program, have a good day.


Step-5

- Human face detected.

```
File Edit Selection View Go Run Terminal Help Face recognition.py - Visual Studio Code
Face recognition.py x
D:\> CV > Face-Recognition-main > Face recognition.py > {} cv2
1 import cv2
2
3 recognizer = cv2.face.LBPHFaceRecognizer_create() # Local Binary Patterns Histograms
4 recognizer.read('trainer/trainer.yml') #load trained model
5 cascadePath = "haarcascade_frontalface_default.xml"
6 faceCascade = cv2.CascadeClassifier(cascadePath) #initializing haar cascade for object
7
8 font = cv2.FONT_HERSHEY_SIMPLEX #denotes the font type
9
10
11 id = 2 #number of persons you want to Recognize
12
13
14 names = ['', 'LIZA SHARMA'] #names, leave first empty bcz counter starts from 0
15
16
17 cam = cv2.VideoCapture(0, cv2.CAP_DSHOW) #cv2.CAP_DSHOW to remove warning
18 cam.set(3, 640) # set video FrameWidth
19 cam.set(4, 480) # set video FrameHeight
20
21 # Define min window size to be recognized as a face
22 minW = 0.1*cam.get(3)
```

PROBLEMS OUTPUT TERMINAL

[Running] python -u "d:\CV\Face-Recognition-main\Face recognition.py"



2. Step wise guide for handwriting recognition system

Step-1

- Open the command prompt and type the below statements and download the packages.

\$pip install pytesseract

python -v your_script.py

```
Command Prompt
Microsoft Windows [Version 10.0.22616.1]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>pip install pytesseract
Collecting pytesseract
  Downloading pytesseract-0.3.9-py2.py3-none-any.whl (14 kB)
Collecting packaging>=21.3
  Using cached packaging-21.3-py3-none-any.whl (40 kB)
Requirement already satisfied: pillow>=8.0.0 in c:\users\hp\appdata\local\programs\python\python310\lib\site-packages (from pytesseract) (9.1.0)
Collecting pyparsing<=3.0.5,>=2.0.2
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
----- 98.3/98.3 kB 700.8 kB/s eta 0:00:00
Installing collected packages: pyparsing, packaging, pytesseract
Successfully installed packaging-21.3 pyparsing-3.0.9 pytesseract-0.3.9

C:\Users\hp>python -v your_script.py
import _frozen_importlib # frozen
import _imp # builtin
import _thread # <class '_frozen_importlib.BuiltinImporter'>
import _warnings # <class '_frozen_importlib.BuiltinImporter'>
import _weakref # <class '_frozen_importlib.BuiltinImporter'>
import _io # <class '_frozen_importlib.BuiltinImporter'>
import _marshal # <class '_frozen_importlib.BuiltinImporter'>
import _nt # <class '_frozen_importlib.BuiltinImporter'>
import _winreg # <class '_frozen_importlib.BuiltinImporter'>
import _frozen_importlib_external # <class '_frozen_importlib.FrozenImporter'>
# installing zipimport hook
import 'time' # <class '_frozen_importlib.BuiltinImporter'>
import 'zipimport' # <class '_frozen_importlib.FrozenImporter'>
# installed zipimport hook
# C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\_pycache\_init_.cpython-310.pyc matches C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\_init_.py
# code object from 'C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\_pycache\_init_.cpython-310.pyc'
# C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\_pycache\_codecs.cpython-310.pyc matches C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\codecs.py
# code object from 'C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\_pycache\_codecs.cpython-310.pyc'
import _codecs # <class '_frozen_importlib.BuiltinImporter'>
# quart _codecs # <_frozen_importlib_external.SourceFileLoader object at 0x00000298D688E60>
# C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\_pycache\_aliases.cpython-310.pyc matches C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\_aliases.py
# code object from 'C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\_pycache\_aliases.cpython-310.pyc'
import 'encodings.aliases' # <_frozen_importlib_external.SourceFileLoader object at 0x00000298D688FE0>
import 'encodings' # <_frozen_importlib_external.SourceFileLoader object at 0x00000298D688F90>
# C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\_pycache\_utf_8.cpython-310.pyc matches C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\utf_8.py
# code object from 'C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\_pycache\_utf_8.cpython-310.pyc'
import 'encodings.utf_8' # <_frozen_importlib_external.SourceFileLoader object at 0x00000298D688F00>
# C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\_pycache\_cp1252.cpython-310.pyc matches C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\cp1252.py
# code object from 'C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\encodings\_pycache\_cp1252.cpython-310.pyc'
import 'encodings.cp1252' # <_frozen_importlib_external.SourceFileLoader object at 0x00000298D688FD0>
import _signal # <class '_frozen_importlib.BuiltinImporter'>
# C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\_pycache\_io.cpython-310.pyc matches C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\io.py
# code object from 'C:\Users\hp\AppData\Local\Programs\Python\Python310\lib\_pycache\_io.cpython-310.pyc'
```

pip install gTTS

pip install playsound

pip3 install pygame

```
Command Prompt
Microsoft Windows [Version 10.0.22616.1]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>pip install gTTS
Collecting gTTS
  Downloading gTTS-2.2.4-py3-none-any.whl (26 kB)
Collecting click
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
----- 96.6/96.6 kB 785.0 kB/s eta 0:00:00
Collecting requests
  Downloading requests-2.27.1-py3-none-any.whl (63 kB)
----- 63.1/63.1 kB 1.7 MB/s eta 0:00:00
Requirement already satisfied: six in c:\users\hp\appdata\local\programs\python\python310\lib\site-packages (from gTTS) (1.16.0)
Requirement already satisfied: colorama in c:\users\hp\appdata\local\programs\python\python310\lib\site-packages (from click->gTTS) (0.4.4)
Collecting charset-normalizer<=2.0.0
  Downloading charset-normalizer-2.0.12-py3-none-any.whl (39 kB)
Collecting idna<4,>=2.5
  Downloading idna-3.3-py3-none-any.whl (61 kB)
----- 61.2/61.2 kB 3.6 MB/s eta 0:00:00
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
----- 139.0/139.0 kB 2.1 MB/s eta 0:00:00
Collecting certifi-2021.4.12
  Downloading certifi-2021.10.8-py2.py3-none-any.whl (149 kB)
----- 149.2/149.2 kB 1.5 MB/s eta 0:00:00
Installing collected packages: certifi, urllib3, idna, click, charset-normalizer, requests, gTTS
Successfully installed certifi-2021.10.8 charset-normalizer-2.0.12 click-8.1.3 gTTS-2.2.4 idna-3.3 requests-2.27.1 urllib3-1.26.9

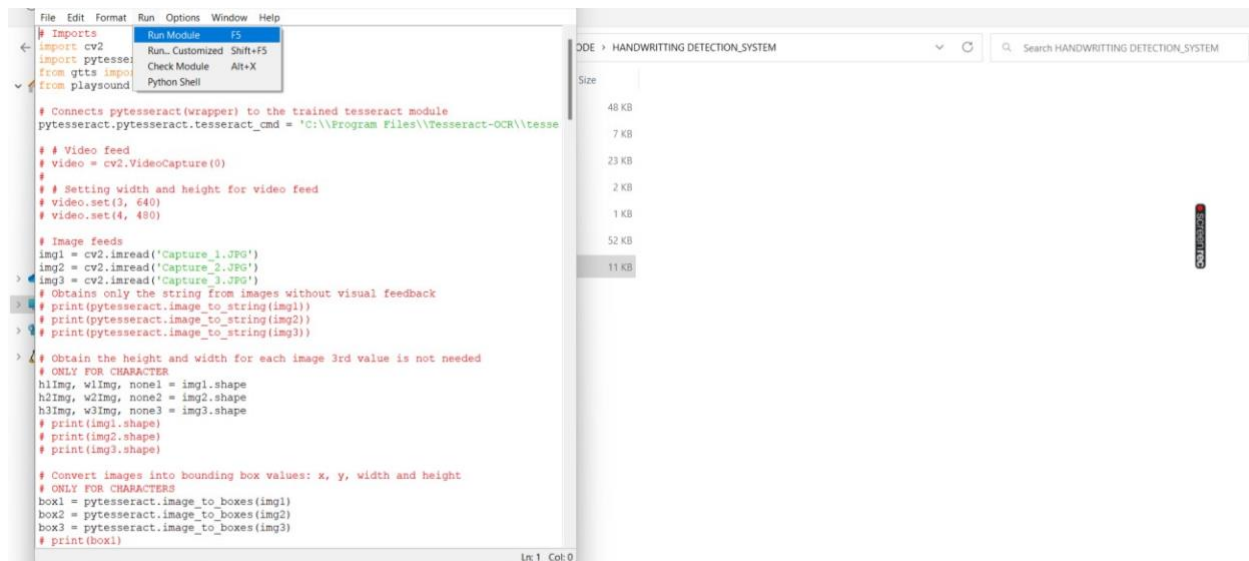
C:\Users\hp>from gtts import gTTS
'from' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\hp>pip install playsound
Collecting playsound
  Downloading playsound-1.3.0.tar.gz (7.7 kB)
  Preparing metadata (setup.py) ... done
Using legacy 'setup.py install' for playsound, since package 'wheel' is not installed.
Installing collected packages: playsound
Running setup.py install for playsound ... done
Successfully installed playsound-1.3.0

C:\Users\hp>pip3 install pygame
Collecting pygame
  Downloading pygame-2.1.2-cp310-cp310-win_amd64.whl (8.4 MB)
----- 8.4/8.4 MB 2.5 MB/s eta 0:00:00
Installing collected packages: pygame
Successfully installed pygame-2.1.2
```

Step-2

- Open the file saved as “Model Trainer.py”
- Click on “Run”
- Click on “Run module”



```
File Edit Format Run Options Window Help
# Imports
import cv2
import pytesseract
from gtts import gTTS
from playsound import playsound

# Connects pytesseract(wrapper) to the trained tesseract module
pytesseract.pytesseract.tesseract_cmd = 'C:\\Program Files\\Tesseract-OCR\\tesseract.exe'

# Video feed
video = cv2.VideoCapture(0)

# Setting width and height for video feed
video.set(3, 640)
video.set(4, 480)

# Image feeds
img1 = cv2.imread('Capture_1.JPG')
img2 = cv2.imread('Capture_2.JPG')
img3 = cv2.imread('Capture_3.JPG')

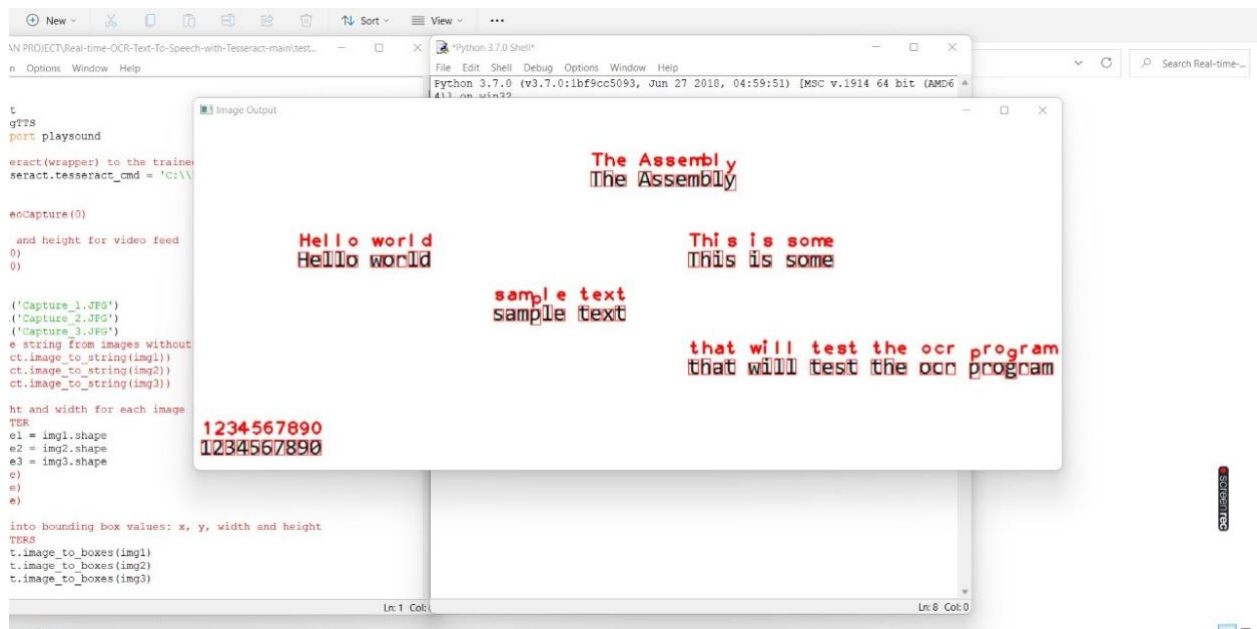
# Obtains only the string from images without visual feedback
print(pytesseract.image_to_string(img1))
print(pytesseract.image_to_string(img2))
print(pytesseract.image_to_string(img3))

# Obtain the height and width for each image 3rd value is not needed
# ONLY FOR CHARACTER
h1img, w1img, none1 = img1.shape
h2img, w2img, none2 = img2.shape
h3img, w3img, none3 = img3.shape
print(img1.shape)
print(img2.shape)
print(img3.shape)

# Convert images into bounding box values: x, y, width and height
# ONLY FOR CHARACTERS
box1 = pytesseract.image_to_boxes(img1)
box2 = pytesseract.image_to_boxes(img2)
box3 = pytesseract.image_to_boxes(img3)
print(box1)
```

STEP-3

- HANDWRITTEN TEXT RECOGNIZED BY SYSTEM AND PROJECT EXECUTED SUCCESSFULLY.



REFERENCES:

- [1] Abdulllah, M., Agal, A., Alharthi, M., & Alrashidi, M. (2018). Retracted: Arabic handwriting recognition using neural network classifier. *Journal of Fundamental and Applied Sciences*, 10(4S), 265270.
- [2] Abe, S. (2010). *Support Vector Machines for Pattern Classification*. Berlin, Germany: Springer Science & Business Media
- [3] Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Basingstoke, England: Springer.
- [4] Balas, V. E., Roy, S. S., Sharma, D., & Samui, P. (2019). *Handbook of Deep Learning Applications*. Basingstoke, England: Springer
- [5] Boukharouba, A., & Bennia, A. (2017). Novel feature extraction technique for the recognition of handwritten digits. *Applied Computing and Informatics*, 13(1), 19-26. doi:10.1016/j.aci.2015.05.001
- [6] Buckland, M. K. (2006). *Emanuel Goldberg and His Knowledge Machine: Information, Invention, and Political Forces*. Santa Barbara, CA: Greenwood Publishing Group.
- [7] Chandio, A. A., Leghari, M., Hakro, D., AWAN, S., & Jalbani, A. H. (2016). A Novel Approach for Online Sindhi Handwritten Word Recognition using Neural Network. *Sindh University Research JournalSURJ (Science Series)*, 48(1)
- [8] Chen, L., Wang, S., Fan, W., Sun, J., & Naoi, S. (2015). Beyond human recognition: A CNN-based framework for handwritten character recognition. 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), 695-699. doi:10.1109/acpr.2015.7486592
- [9] Ding, S., Zhao, H., Zhang, Y., Xu, X., & Nie, R. (2015). Extreme learning machine: algorithm, theory and applications. *Artificial Intelligence Review*, 44(1), 103-115
- [10] Dwivedi, U., Rajput, P., Sharma, M. K., & Noida, G. (2017). Cursive Handwriting Recognition System Using Feature Extraction and Artificial Neural Network. *Int. Res. J. Eng. Technol*, 4(03),