CHAPTER 1

1. INTRODUCTION

1.1 Introduction

Object tracking is a computer vision technique in which a software system can detect, locate, and trace the object from a given image or video. The special attribute about object tracking is that it identifies the class of object (person, table, chair, etc.) and their location-specific coordinates in the given image. The location is pointed out by drawing a bounding box around the object. The bounding box may or may not accurately locate the position of the object. The ability to locate the object inside an image defines the performance of the algorithm used for tracking. Face tracking is one of the examples of object tracking.

These object tracking algorithms might be pre-trained or can be trained from scratch. In most use cases, we use pre-trained weights from pre-trained models and then fine-tune them as per our requirements and different use cases.

Let us first find about Two -shot tracking method. As the name suggests there are two stages involved in this method. One is region proposal and then in the second stage, the classification of those regions and refinement of the location prediction takes place.

Faster-RCNN variants are the popular choice of usage for two-shot models. Here during the region proposal stage, we use a network such as ResNet50 as a feature extractor. We do this by removing the last layers of this network and just use the rest of the layers to extract features from the images. This is usually a better approach as the network is already trained and can extract features from the images. Next, a small fully connected network slides over the feature layer to predict class-agnostic box proposals, with respect to a grid of anchors tiled in space, scale and aspect ratio.

In the second stage, these box proposals are used to crop features from the intermediate feature map which was already computed in the first stage. The proposed boxes are fed to

the remainder of the feature extractor in which the prediction and regression heads are added on top of the network. Finally, in the output, we get the class and class-specific box refinement for each proposal box.

On the contrary side, Single-shot tracking skips the region proposal stage and yields final localization and content prediction at once. It must be noted that two-shot tracking models achieve better performance but single-shot tracking is in the sweet spot of performance and speed/resources which makes it more suitable for tasks like detecting objects in live feed or object tracking where the speed of prediction is of more importance.

1.2 Existing Systems

A. Data-Driven Approaches in Credit Card Fraud Detection: Both supervised and unsupervised methods have been proposed for credit card fraud-detection purposes. Unsupervised methods consist in outlier/ anomaly detection techniques that consider as a fraud any transaction that does not conform with the majority. Remarkably, an unsupervised DDM in an FDS can be directly configured from unlabeled transactions. A well-known method is peer group analysis, which clusters customers according to their profile and identifies frauds as transactions departing from the typical cardholder's behavior. The typical cardholder's behavior has also been modeled by means of self-organizing maps. Supervised methods are by far the most popular in fraud detection, and exploit labeled transactions for training a classifier. Frauds are detected by classifying feature vectors of the authorized transactions or possibly by analyzing the posterior of the classifier.

B. Performance Measure for Fraud Detection: The typical performance measure for fraud-detection problems is the AUC. AUC can be estimated by means of the Mann–Whitney statistic and its value can be interpreted as the probability that a classifier ranks frauds higher than genuine transactions. Another ranking measure frequently used in fraud detection is average precision, which corresponds to the area under the precision– recall curve. While these measures are widely used in

detection problems, cost-based measures have been specifically designed for fraud-detection purposes. Cost-based measures quantify the monetary loss of a fraud by means of a cost matrix that associates a cost with each entry of the confusion matrix.

C. Major Challenges to be Addressed in a Real-World FDS:

1) Class Imbalance: Class distribution is extremely unbalanced in credit card transactions, since frauds are typically less than 1% of the overall transactions in our analysis. Learning under class imbalance has lately received a lot of attention, since traditional learning methods yield classifiers that are poorly performing on the minority class, which is definitively the class of interest in detection problems. Several techniques have been proposed to deal with class imbalance, and for a comprehensive overview, we refer the reader to. The two main approaches for dealing with class imbalance are: a) sampling methods and b) cost-based methods.

2) Concept Drift: There are two main factors introducing changes/evolutions in the stream of credit card transactions, which in the literature are typically referred to as concept drift. At first, genuine transactions evolve because cardholders typically change their spending behaviors over time (e.g., during holidays, they purchase more and differently from the rest of the year). Second, frauds change over time, since new fraudulent activities are perpetrated. In our experiments.

3) Alert–Feedback Interaction and Sample Selection Bias: The majority of classifiers used for credit card fraud detection in the literature are tested in experiments where transaction labels are supposed to be available the very next day since the transaction is authorized. In a real-world FDS, the only recent supervised information is the feedbacks Ft , provided by investigators, while the vast majority of transactions authorized everyday do not receive a label in a short time ($|F_t| << |T_t|$).

The interaction between the FDS (raising alerts) and the investigators (providing true labels) recalls the active learning scenario, where it is possible to select few—very informative—samples and query their labels to an oracle which in the FDS would be the investigators. However, this is not feasible in a real-world FDS, since

investigators have to focus on the most suspicious transactions to detect the largest number of frauds. Requests to check (possibly genuine) transactions for obtaining informative samples would be ignored. Considering the limited number of transactions investigators can check, addressing these questions would necessarily imply that some high-risk transaction is not being controlled, with the consequent loss in detection performance.

1.3 Proposed Systems

Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problem was that of image classification, which is defined as predicting the class of the image. A slightly complicated problem is that of image localization, where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object). The more complicated problem (this project), of object detection involves both classification and localization.

In this case, the input to the system will be a image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of object in each box.

CHAPTER 2

2. REQUIREMENT ANALYSIS

2.1 Requirement Analysis

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

2.2 Requirement Specification

2.2.1 Functional requirement

In software engineering, a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

Functional software requirements help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform.

2.2.2 Non-Functional requirement

A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?

A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

2.3 Computational resource requirements

2.3.1 Hardware requirements :-

- ✓ Processor : Pentium IV or higher
- ✓ RAM : 512 MB
- ✓ Hard Disk : 40 GB
- ✓ Mouse : Optical Mouse.
- ✓ Monitor : 15’ Colour Monitor.

2.3.2 Software requirements :-

- ✓ Operating System : Windows 7 or Above
- ✓ Compiler / Interpreter : Python 3.7
- ✓ Libraries : numpy & opencv-python,

CHAPTER 3

2. DESIGN

3.1 Design

This chapter provides the design phase of the Application. To design the project, we use the UML diagrams. The Unified Modelling Language (UML) is a general- purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

3.2 Architecture

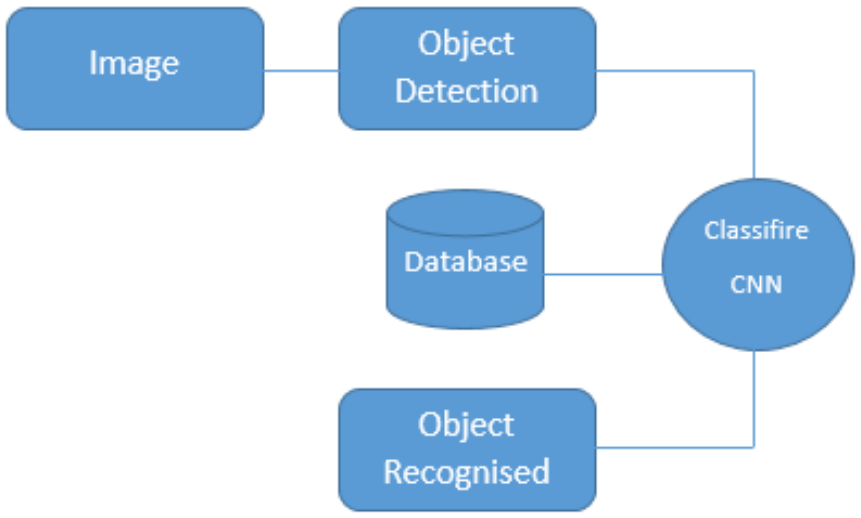

Figure No. 3.1 Architecture

The Above Figure no 3.1 show the architecture of object tracking system

3.3 Use case Diagram

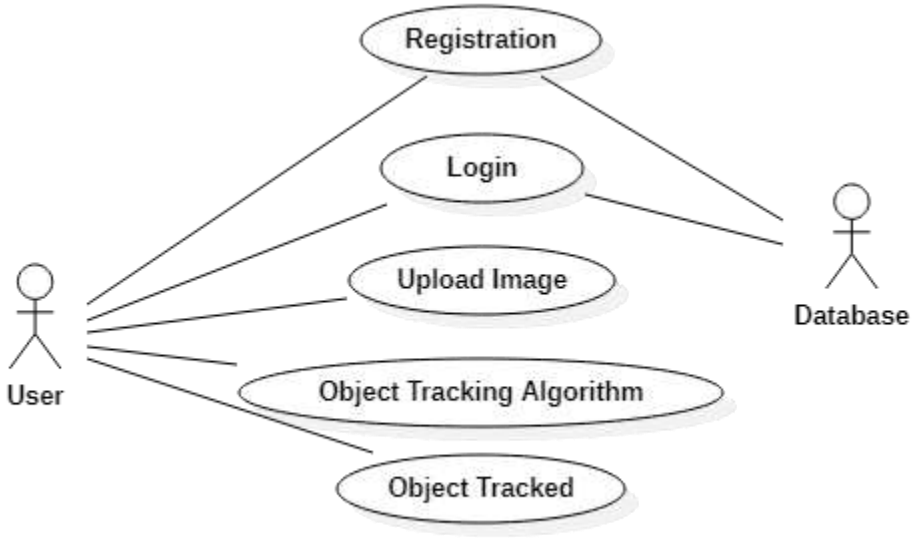

Figure 3.2 Use case Diagram

The above diagram represents the User actor in the project.

- User

3.4 Class Diagram

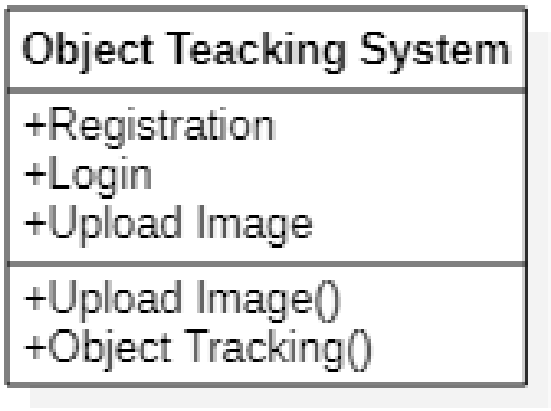

Figure 3.3 class diagram

The above mentioned class diagram represents the system workflow model

3.5 Sequence Diagram

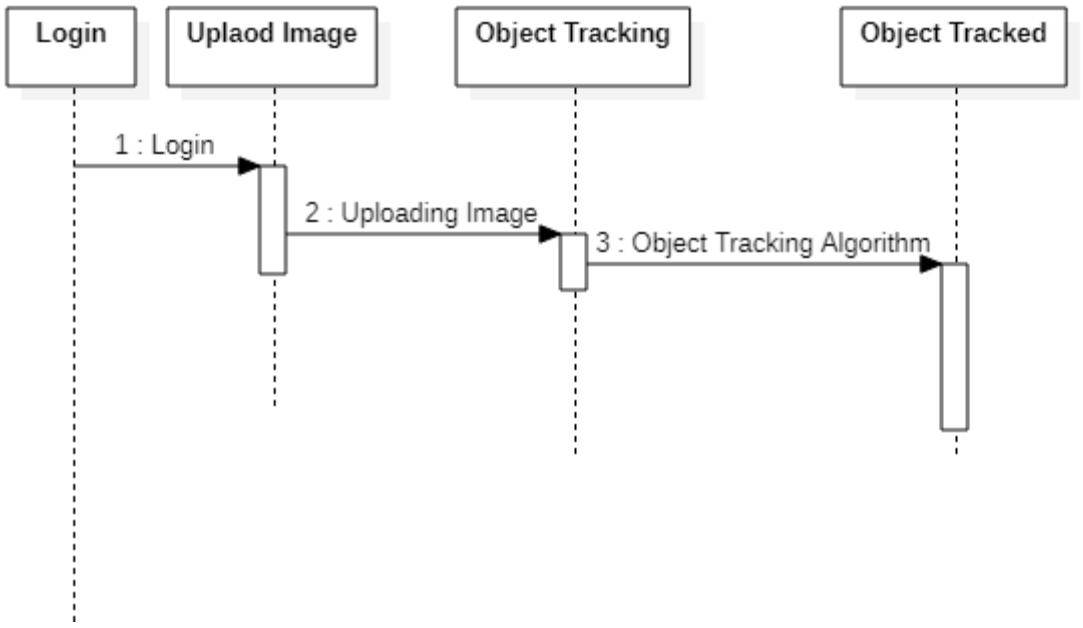

Figure 3.4 Sequence diagram

The above diagram represents the sequence of flow of actions in the system.

3.6 State Chart Diagram

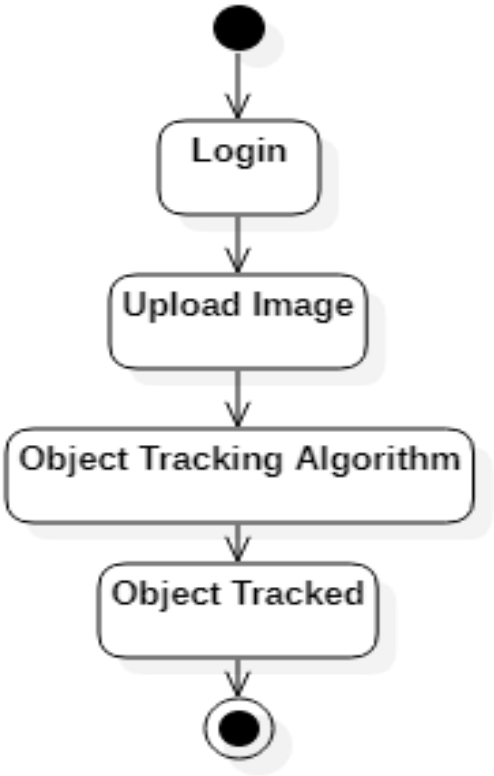

Figure 3.5 State Chart diagram

The above diagram represents the State Chart of flow of action.

CHAPTER 4

4. MODULES

4.1 MODULES

This chapter mainly provides the overview on modules of the application, objectives of the project and a detailed project overview.

4.2 Bounding Box

The bounding box is a rectangle drawn on the image which tightly fits the object in the image. A bounding box exists for every instance of every object in the image. For the box, 4 numbers (center x, center y, width, height) are predicted. This can be trained using a distance measure between predicted and ground truth bounding box.

4.3 Classification

The bounding box is predicted using regression and the class within the bounding box is predicted using classification.

4.4 Two stage method

In this case, the proposals are extracted using some other computer vision technique and then resized to fixed input for the classification network, which acts as a feature extractor. Then an SVM is trained to classify between object and background (one SVM for each class). Also a bounding box regress or is trained that outputs some correction (offsets) for proposal boxes.

4.5 Unified method

The difference here is that instead of producing proposals, pre-define a set of boxes to look for objects. Using convolutional feature maps from later layers of the network, run another network over these feature maps to predict class scores and bounding box offsets.

CHAPTER 5

5. IMPLEMENTATION

5.1 Sample Code

```
#import modules

from tkinter import *

import os

import json

import io

from PIL import Image ,ImageTk

import matplotlib

from matplotlib import pyplot as plt

import matplotlib.patches as patches

import requests

from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from tkinter import filedialog

from tkinter import *
```

```
# Designing window for registration


def register():
    global register_screen
    register_screen = Toplevel(main_screen)
    register_screen.title("Register")
    register_screen.geometry("300x250")

    global username
    global password
    global username_entry
    global password_entry
    username = StringVar()
    password = StringVar()

    Label(register_screen, text="Please enter details below", bg="blue").pack()
    Label(register_screen, text="").pack()
    username_lable = Label(register_screen, text="Username * ")
    username_lable.pack()
    username_entry = Entry(register_screen, textvariable=username)
    username_entry.pack()
    password_lable = Label(register_screen, text="Password * ")
    password_lable.pack()
```

```
    password_entry = Entry(register_screen, textvariable=password, show='*')
    password_entry.pack()
    Label(register_screen, text="").pack()
    Button(register_screen, text="Register", width=10, height=1, bg="blue", command =
register_user).pack()


# Designing window for login


def login():
    global login_screen
    login_screen = Toplevel(main_screen)
    login_screen.title("Login")
    login_screen.geometry("300x250")
    Label(login_screen, text="Please enter details below to login").pack()
    Label(login_screen, text="").pack()

    global username_verify
    global password_verify

    username_verify = StringVar()
    password_verify = StringVar()

    global username_login_entry
```

```
    global password_login_entry


    Label(login_screen, text="Username * ").pack()
    username_login_entry = Entry(login_screen, textvariable=username_verify)
    username_login_entry.pack()
    Label(login_screen, text="").pack()
    Label(login_screen, text="Password * ").pack()
    password_login_entry = Entry(login_screen, textvariable=password_verify, show= '*')
    password_login_entry.pack()
    Label(login_screen, text="").pack()
    Button(login_screen, text="Login", width=10, height=1, command =
login_verify).pack()

# Implementing event on register button

def register_user():


    username_info = username.get()
    password_info = password.get()


    file = open(username_info, "w")
    file.write(username_info + "\n")
    file.write(password_info)
```

```
    file.close()
    username_entry.delete(0, END)
    password_entry.delete(0, END)
    Label(register_screen, text="Registration Success", fg="green", font=("calibri",
11)).pack()


# Implementing event on login button
def login_verify():
    username1 = username_verify.get()
    password1 = password_verify.get()
    username_login_entry.delete(0, END)
    password_login_entry.delete(0, END)
    list_of_files = os.listdir()
    if username1 in list_of_files:
        file1 = open(username1, "r")
        verify = file1.read().splitlines()
        if password1 in verify:
            login_sucess()
        else:
            password_not_recognised()
    else:
        user_not_found()
```

```
# Designing popup for login success
def login_sucess():
    global login_success_screen


    main = tkinter.Tk()
    main.title("Multy Object Detection")
    main.geometry("1300x1200")


    global filename
    global refrence_pixels
    global refrence_pixels
    global sample_pixels
    def uploadTrafficImage():
        global filename
        filename = filedialog.askopenfilename(initialdir="images")
        pathlabel.config(text=filename)
    def model(img, port=None):
        if port:
            url = 'http://localhost:' + str(local_port) + '/model/predict'
        else:
            url = 'http://max-object-detector.codait-prod-
41208c73af8fca213512856c7a09db52-0000.us-
east.containers.appdomain.cloud/model/predict'
```

```
        files = {'image': ('image.jpg', open(img, 'rb'), 'images/jpeg')}
        r = requests.post(url, files=files).json()
        return r
    def exit():
        main.destroy()
    def mmm():
        import matplotlib.image as mpimg
        img = filename
        print("Image Path:", img)
        from PIL import Image
        image = Image.open(img)
        model_response = model(img)
        print(json.dumps(model_response, indent=2))
        image_width, image_height = image.size
        fig, ax = plt.subplots()
        fig.set_dpi(600)
        plt.imshow(image)
        color1 = '#008000'
        color = '#800000'
        # For each object, draw the bounding box and predicted class together with the
probability
        for prediction in model_response['predictions']:
            bbox = prediction['detection_box']
```

```
        # Unpack the coordinate values

        y1, x1, y2, x2 = bbox

        # Map the normalized coordinates to pixel values: scale by image height for 'y'
and image width for 'x'

        y1 *= image_height

        y2 *= image_height

        x1 *= image_width

        x2 *= image_width

        # Format the class probability for display

        probability = '{0:.4f}'.format(prediction['probability'])

        # Format the class label for display

        label = '{}'.format(prediction['label'])

        label = label.capitalize()

        rectangle = patches.Rectangle((x1, y1), x2 - x1, y2 - y1, linewidth=1,
edgecolor=color, facecolor='none')

        ax.add_patch(rectangle)

        # Plot the bounding boxes and class labels with confidence scores

        plt.text(x1, y1 - 5, label, fontsize=2, color=color1, fontweight='bold',
horizontalalignment='left')

        plt.text(x2, y1 - 5, probability, fontsize=2, color=color1, fontweight='bold',
horizontalalignment='right')

    plt.axis('off')

    plt.show()
```

```
font = ('times', 16, 'bold')
title = Label(main, text=' Multy Object Detection', anchor=W, justify=CENTER)
title.config(bg='yellow4', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0, y=5)
font1 = ('times', 14, 'bold')
upload = Button(main, text="Upload  Image", command=uploadTrafficImage)
upload.place(x=50, y=100)
upload.config(font=font1)
process = Button(main, text="Image Detection", command=mmm)
process.place(x=50, y=200)
process.config(font=font1)
pathlabel = Label(main)
pathlabel.config(bg='yellow4', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=50, y=150)
exitButton = Button(main, text="Exit", command=exit)
exitButton.place(x=50, y=350)
exitButton.config(font=font1)
main.config(bg='magenta3')
main.mainloop()
```

```
# Designing popup for login invalid password


def password_not_recognised():
    global password_not_recog_screen
    password_not_recog_screen = Toplevel(login_screen)
    password_not_recog_screen.title("Success")
    password_not_recog_screen.geometry("150x100")
    Label(password_not_recog_screen, text="Invalid Password ").pack()
    Button(password_not_recog_screen, text="OK",
command=delete_password_not_recognised).pack()


# Designing popup for user not found


def user_not_found():
    global user_not_found_screen
    user_not_found_screen = Toplevel(login_screen)
    user_not_found_screen.title("Success")
    user_not_found_screen.geometry("150x100")
    Label(user_not_found_screen, text="User Not Found").pack()
    Button(user_not_found_screen, text="OK",
command=delete_user_not_found_screen).pack()


# Deleting popups
```

```
def delete_login_success():
    login_success_screen.destroy()
def delete_password_not_recognised():
    password_not_recog_screen.destroy()
def delete_user_not_found_screen():
    user_not_found_screen.destroy()


# Designing Main(first) window


def main_account_screen():
    global main_screen
    main_screen = Tk()
    main_screen.geometry("300x250")
    main_screen.title("Account Login")
    Label(text="Select Your Choice", bg="blue", width="300", height="2",
font=("Calibri", 13)).pack()
    Label(text="").pack()
    Button(text="Login", height="2", width="30", command = login).pack()
    Label(text="").pack()
    Button(text="Register", height="2", width="30", command=register).pack()
    main_screen.mainloop(()
main_account_screen()
```

CHAPTER 6

6. SCREENSHOTS

6.1 Screenshots.

6.1.1 Account Login Page

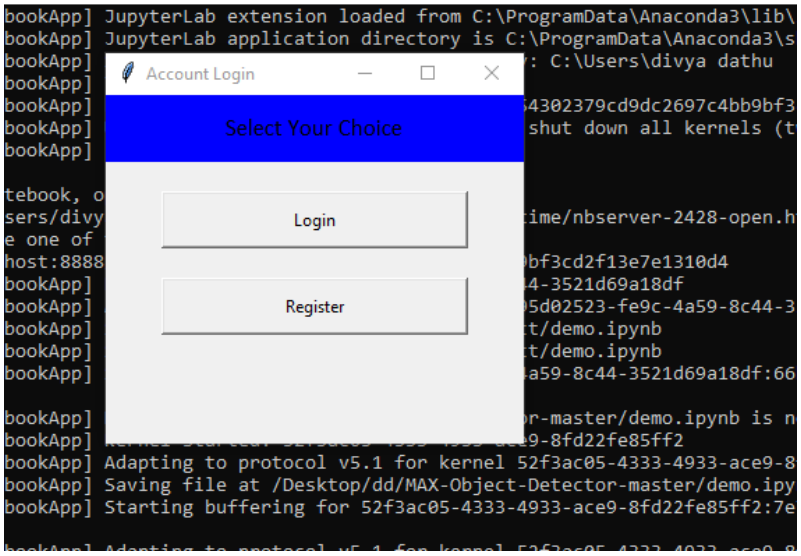

Figure 6.1 Account Login Page

The Screenshot show the Account login or registration of Object Tracking System

6.1.2 Registration Page

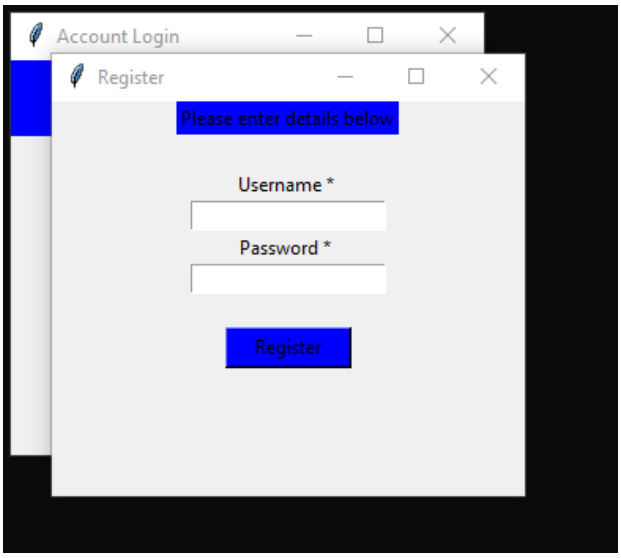

Figure No 6.2 Registration Page

The Screenshot show the Registration Page of Object Tracking System

6.1.3 Login Page

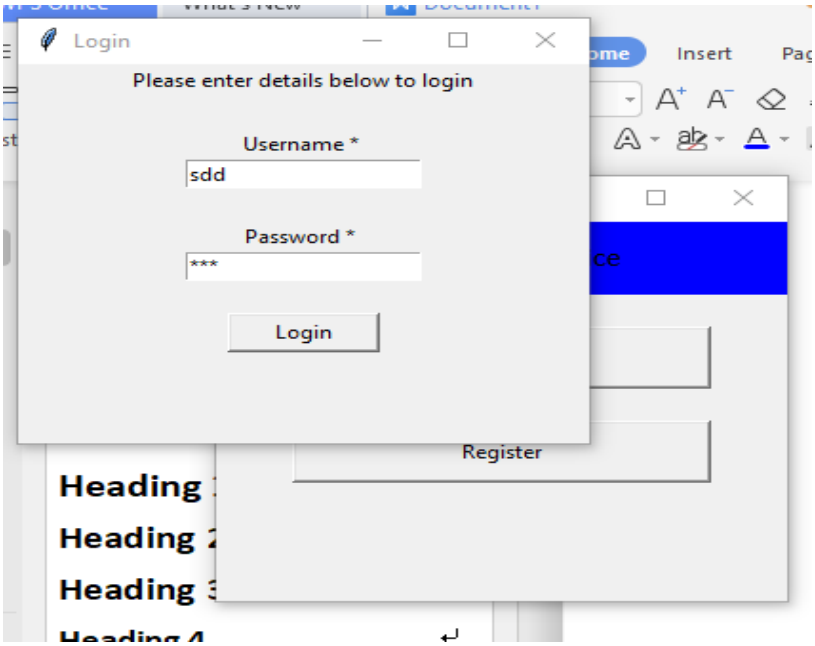

Figure No 6.3 Login Page

The Screenshot show the login of Object Tracking System

6.1.4 Multy Object Detection Page

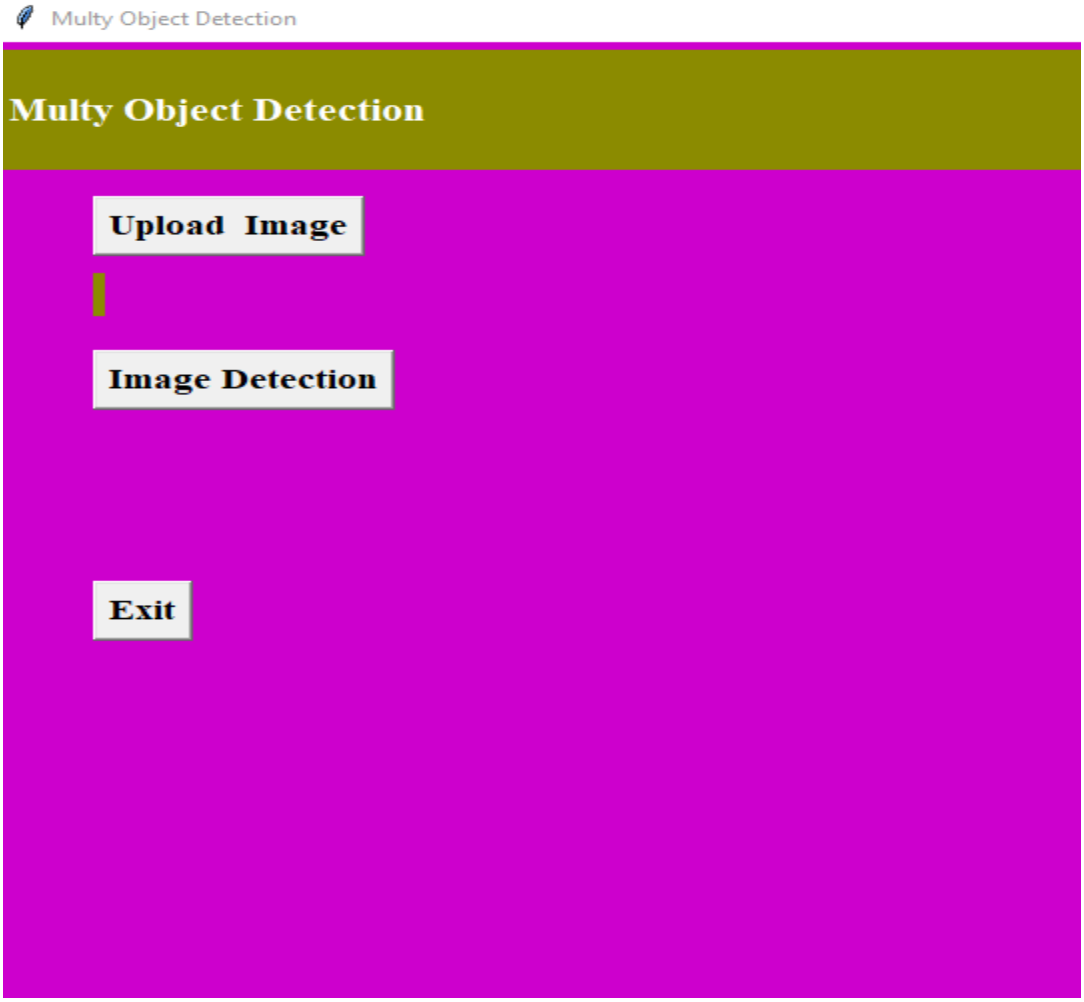

Figure No 6.4 Multy Object Detection Page

The Screenshot show the Multy Object Detection Page of Object Tracking System

6.1.5 Uploaded Image

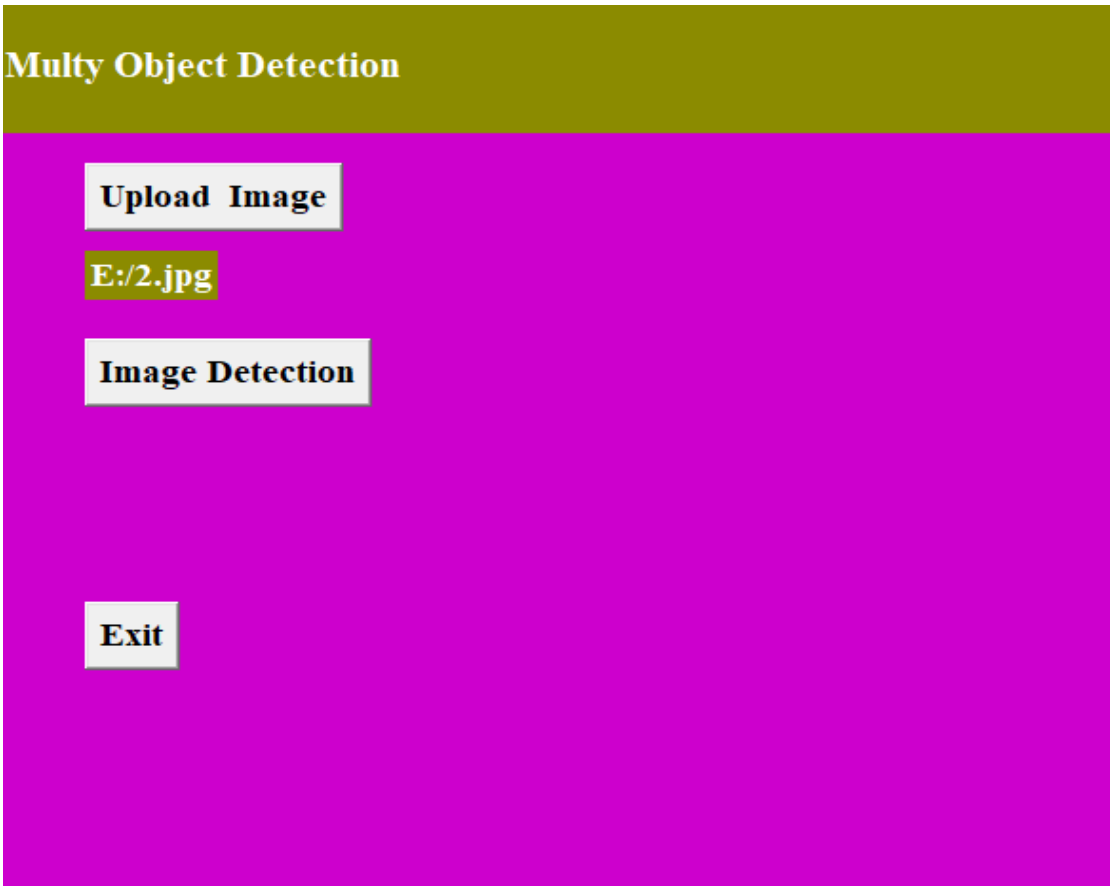

Figure No 6.5 Uploaded Image

The Screenshot show the Uploaded Image in Multy Object Detection Page of Object Tracking System

6.1.6 Tracking Image

Figure No 6.6 Tracking Image

The Screenshot show the Tacking Image in Multy Object Detection Page of Object Tracking System

CHAPTER 7

7. TESTING

7.1 Overview of Testing

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

7.2 Types of Testing

7.2.1 Unit testing

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

7.2.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more

concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.3 Unit Testing :-

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

Test strategy and approach

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

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.
- Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.4 Integration Testing

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

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

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

7.5 Acceptance Testing

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

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

CHAPTER 8

8. CONCLUSION AND FUTURE WORK

8.1 Conclusion

An accurate and efficient object detection system has been developed which achieves comparable metrics with the existing state-of-the-art system. This project uses recent techniques in the field of computer vision and deep learning. Custom dataset was created using label Image and the evaluation was consistent. This can be used in real-time applications which require object detection for pre-processing in their pipeline.

8.2 Scope for Future work

An important scope would be to train the system on a video sequence for usage in tracking applications. Addition of a temporally consistent network would enable smooth detection and more optimal than per-frame detection.

REFERENCES

[1] P. Salesmbier, L. Torres, F. Meyer and C. Gu, "Regionbased Video Coding Using Mathematical Morphology," Proc. of the IEEE, vol. 83, no. 6, pp. 843-857, 1995.

[2] Harris C. & Stennett C. "Rapid - A Video Rate Object Tracker", Proc. British Machine Vision Conference, BMVC-90, Oxford, pp.73-77, 1990.

[3] M. Isard and A. Blake, "Contour Tracking by Stochastic Propagation of Conditional Density," In Proc. European Conf. Computer Vision, pp. 343-356, 1996.

[4] B. Rao, "Data Association Methods for Tracking Systems," In A. Black and A. Yuille, editors, Active Vision, pp. 91-105, MIT, 1992.

[5] Wang, H., et al., “Adaptive object tracking based on an effective appearance filter.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 9, pp. 1661- 1667, 2007.

[6] Tang, P., et al. Stochastic approach based salient moving object detection using kernel density estimation, Wuhan, China: SPIE, 2007.

[7] Arulampalam M.S., Maskell S., Gordon N., Clapp T., “A Tutorial on Particle Filters for Online Nonlinear/NonGaussian Bayesian Tracking”, IEEE Transactions on Signal Processing, vol. 50, no. 2, pp. 174-188, 2002.