DETECTION OF WILD ELEPHANTS USING IMAGE PROCESSING ON RASPBERRY PI

ANN JESSIE SANDRA P(16Z205)
HARINI S(16Z221)
SWATHI P R(17Z436)
NITHYASRI S(17Z437)

15Z820 PROJECT WORK II

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

Of Anna University

APRIL 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)
COIMBATORE – 641 004

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

DETECTION OF WILD ELEPHANTS USING IMAGE PROCESSING ON RASPBERRY PI

Bona fide record of work done by

ANN JESSIE SANDRA P	16Z205
HARINI S	16Z221
SWATHI P R	17Z436
NITHYASRI S	17Z437

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

of Anna University

August 2020

Lovelyn Rose.S

Dr. Lovelyn Rose S
Faculty guide

g.sudha

Dr. Sudha Sadasivam G
Head of the Department

Certified that the candidate was examined in the viva-voce examination held on 3/9/20

Lovelyn Rose.S 3/9/20
.....
(Internal Examiner)

3/9/20
.....
(External Examiner)

CERTIFICATE

Certified that this report titled **"Detection of wild elephants using Image Processing on Raspberry Pi"**, for the **PROJECT WORK II** (15Z820) is a bonafide work of **Ann Jessie Sandra P(16z205), Harini S(16z221), Swathi P R(17z436), Nithyasri S(17z437)** who have carried out the work under my supervision for the partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion.

Place: Coimbatore

Date:

Dr. Lovelyn Rose S

Designation: Associate Professor

Department of Computer Science and Engineering

PSG College of Technology

Coimbatore - 641004

COUNTERSIGNED

HEAD
Department of Computer Science and Engineering
PSG College of Technology, Coimbatore - 641004

CONTENTS

ACKNOWLEDGEMENT

We would like to extend our sincere thanks to **Dr. K. Prakasan** Principal, PSG College of Technology, for providing us with a wonderful opportunity to develop and complete a project in our field of study.

We profoundly thank, **Dr. G. Sudha Sadasivam**, Professor & Head, Department of Computer Science and Engineering, who has greatly helped in the success of the project, by providing the necessary facilities that were required.

We take immense pleasure in conveying our thanks and deep sense of gratitude to the program coordinator **Dr. G. R. Karpagam**, Professor, Associate HOD, Department of Computer Science and Engineering, for her valuable suggestions and motivation all through the course of the project.

We would like to express our deepest gratitude to our project guide **Dr.Lovelyn Rose S**, Associate Professor, Department of Computer Science and Engineering, for the exemplary guidance, timely advice, valuable directions, constant support and encouragement given in completing our project work.

We express our thanks and gratitude to our tutor **Mr. Prakash J** and assistant tutor **Ms. Mano Chitra M**, Assistant Professor, Department of computer Science and Engineering for help and encouragement in the duration of the course.

We express our sincere thanks to all the panel members and faculty in charge, for helping us in resolving the problems that incurred during the project work.

SYNOPSIS

Image recognition is a developing technology with many practical applications. One such application is recognizing elephants in a forest survey system. Various causes such as increasing population, industrialization and deforestation has put pressure on the wildlife. As safety is an important criterion, prior detection of the animal intrusion is necessary to avoid situations like destruction of agriculture, entry into human living space for food and shelter and also to prevent economic loss. In this project, we propose the process of image processing on Raspberry Pi using machine learning model CNN (Convolutional Neural Networks) for the detection of elephant that access through the forest borders. First the dataset, or in this particular case, the elephant images were obtained from various credential open sources. The four best feature detection methods: SIFT (Scale Invariant Feature Transform), SURF (Speed-Up Robust Feature), HOG (Histogram of oriented gradients) and ORB (Oriented FAST and rotated BRIEF) were applied on the dataset for feature analyses. After that, the Machine Learning model was designed, trained on the dataset and then tested for their accuracies while classifying the images using CNN. Since each feature detection method works differently giving different outcomes, it provided the platform that helps us to compare the results and to identify the reliable one that works best along with the CNN model based on their accuracy factor. As a result of the comparison made among the feature detector methods with CNN the SIFT feature detector works best with CNN with an Accuracy score of 0.85 provided.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In India, elephant population is around 60 percentage thus one-third of the elephants live closely to the human prevailing landscapes. The human population in India is also growing side by side which is also a major cause for the industrial growth. This results in the destruction of the forest areas to make human settlements. Because of this, the elephant populations face shortage of resources such as food and water, this made them to move into the human habitat. Hence, there has been severe Human elephant conflict. The circumstances under which the elephants are moving into the human habitat is becoming a growing research area where many researchers are working hard.

As mentioned, the factors that are contributing to this research is enormous like human population, elephant population, weather, Seasonal habitat structure. Even the fences made by the people or artificially maintained water sources attract the elephants during drought to make an intrusion. At most importance is to be provided to reduce the risk of both human and elephants from damage.

Elephant conservation issues can be divided into two main broad categories namely;

- Direct activities such as hunting, poaching and capturing which affects the elephants
- Indirect developmental activities that are affecting the elephant habitat.

To avoid the human elephant conflict, many methods are being used;

- One such method is the elephant proof trenches. The advance warning has to be given to the people residing near the forest area to avoid loss.
- Electric Fencing is one of the techniques used to repel the elephants, here the detection of the elephants in the field of surveillance is done after that the electric current is connected to the fence. The main problem in this technique is it causes accidents to humans and power consumption.

- Even solar fencing is used nowadays to avoid elephant intrusion. As elephant living landscapes are being diminished, the elephants are moving to the human landscape due to the shortage of the area. This makes the elephant population to get increased and also the food shortage problem arises. In need of food, the elephants are coming out from the forest. The forest officers are trying with many techniques to give prior alert.

But still now they didn't find a suitable solution which works continuously with good outcomes.

1.1.1 IMAGE PROCESSING:

Image Processing is used to either enhance or compress the image information and it transforms one image into another using pixel wise operations. The pixel wise operations will not provide meaningful information thus to overcome this limitation, the computer vision is used which helps in getting the abundant information about the contents of the image using feature detectors. It provides a wide range of use and the reliability obtained is high.

Types:

Analog Image Processing:

- Image processing function conducted on two-dimensional analog signals.

Digital Image Processing:

- Image processing using a digital computer and processing the digital images using various algorithms and it provides a wider range of algorithms that can be applied on the input data and the problem of distortion is avoided during processing.

Advantages:

- Image processing helps in removing noises, formatting images in any form like black and white, negative image and in correcting image density and contrast.

Applications:

- Defense and security.
- Image sharpening and restoration.
- Medical fields.

- Robot vision.
- Pattern recognition.
- Video processing

Limitation:

- The initial cost for setting up is high respective to the systems used and the processor that is being used should be faster.

1.1.2 RASPBERRY PI:

Raspberry Pi is a small single board computer that runs Linux but it provides a set of GPIO (general purpose input/output) pins that helps in physical computing by controlling the electronic components. And only we need an SD card to be inserted into the slot on the board that acts as the hard drive for the Raspberry Pi.

The power is provided to the Raspberry Pi through an USB (Universal Serial Bus) and the video output is displayed on a modern monitor or to any display using HDMI (High-Definition Multimedia Interface) port. The power consumption is extremely low in Raspberry Pi. It provides all the basic capabilities of a normal computer. This gives you all of the basic abilities of a normal computer.

Advantages

- Installation of a full-fledged operating system is possible (Raspbian OS).
- To make the wireless mode of accesses, the advanced Raspberry Pi are used which has Wi-Fi and Bluetooth as built in functions.
- The GPIO helps in better progress than the normal computers as we can connect external components and sensors and make an interaction using python programming language for better efficiency.

Disadvantages

- The board gets damaged if the GPIO (General Purpose Input/output pins) are connected incorrectly.
- The analog to digital conversion on the pins are not possible as it is not built in and for that it is needed to use an extra ADC (Analog-to-Digital

Converter) chip.

1.1.3 FEATURE DETECTION

Feature detection is the first operation to be performed on an image in order to examine each pixel in an image whether it has a feature or not. This is the low-level task to be conducted in an image processing operation. In order to find the features many methods are used.

1.1.3.1 Scale Invariant Feature Transform (SIFT)

For all the objects there are many features and the interesting points on the object are extracted to provide a feature that is the description of the object. When we attempt to locate any object then the description that we already found will be used in that image which contains many other objects.

SIFT image features provide a set of features of an object which are scale invariant and rotation invariant. This makes this method a flexible feature detection method as it removes many complications.

1.1.3.2 Speeded Up Robust Features (SURF)

It is a patented local feature detector and descriptor in computer vision. As the SIFT is comparatively slow there comes the need of a speeded-up version and the SURF was found. It is a speeded-up version of SIFT.

In sift for finding the scale space the Laplacian of gaussian is used but SURF goes further by approximating LoG with Box filter. The advantage of this approximation is that, convolution with box filter can be easily calculated with the help of integral images in parallel for different scales.

SURF is better than SIFT in rotation invariant, blur and warp transform. SIFT is better than SURF in different scale images. SURF is 3 times faster than SIFT because using of integral image and box filter. SIFT and SURF are good in illumination changing images.

1.1.3.3 Histogram of oriented gradients (HOG)

It is a feature descriptor used in computer vision and image processing for the purpose of object detection. It decomposes an image into small squared cells, computes a

histogram of oriented gradients in each cell, normalizes the result using a block-wise pattern, and return a descriptor for each cell.

The global features are extracted using HOG. But it is not as efficient as SIFT because SIFT extracts the local features which gives a bunch of key points. And HOG is neither scale nor rotation invariant.

1.1.3.4 Oriented FAST and rotated BRIEF(ORB)

It is a fast-robust local feature detector that can be used in computer vision tasks like object recognition or 3D reconstruction. ORB is a very fast binary descriptor based on BRIEF, which is rotation invariant and resistant to noise.

The best alternative to SIFT AND SURF is the Oriented FAST and rotated BRIEF(ORB) because it is resistant to both noise and scale invariant. In angle proportional images the ORB and SURF performs well than the SIFT but when it comes to noisy images ORB and SIFT performs well than SURF thus in both the cases the ORB performs well and in equal with the other methods.

Using one of the mentioned techniques, features of the elephant are detected from 500 sample elephant images.

1.1.4 IMAGE CLASSIFICATION:

Image classification refers to the task of extracting information classes from a multiband image where the band refers to the cell values by a single matrix where multiband has multiple spatially coincident matrices of cell values representing the same spatial area.

CNN (CONVOLUTIONAL NEURAL NETWORK):

CNN (Convolutional Neural Network) are used in image recognition where the convolution layers are used for feature extraction. It is designed to map image data (2D multi-dimensional data) to an output variable (1 dimensional data) CNN have proved so effective, that they are the ready to use method for any type of prediction problem involving image data as an input. Then, it performs classification by differentiating elephants and others.

As this project uses image dataset as input, the CNN is used for classification of the images

1.2 OBJECTIVE OF THE PROJECT

The causes for the elephant intrusion are complex and humans are in need to make a solution for this problem. This project that we are working on aims at providing a platform to understand the causes and then making a detection process which helps in safeguarding the people living nearby the forest border areas by using a trained Convolutional neural network model and the project is intended to use raspberry pi for image recognition as it provides an efficient platform.

To meet this requirement, initially a model is built and is trained using suitable image dataset and the trained model is stored so that it can be used later for image classification and in order to make it more efficient the feature detection methods namely Scale Invariant Feature Transform, Speeded up robust features, Histogram of oriented gradients, Oriented FAST and rotated BRIEF are used prior to the classification to detect the features of the images provided.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

- The minimum hardware to run the project is given below

Processor	Core i5 or above
Processor speed	2.4 GHz or above
System Memory	4 GB minimum
Secondary Memory	80 -100 GB
Network Connection	Internet connection with minimum speed 500 kbps
Raspberry Pi	B+
External component	USB, HDMI cable

Table 2.1 Hardware Requirements

2.2 SOFTWARE REQUIREMENTS

- The minimum software to run the project is given below

Operating System	Raspbian
IDE	Thonny
Machine learning framework	TensorFlow -1.14.0
Preprocessing Module	Keras-preprocessing-1.1.0

Table 2.2 Software Requirements

2.3 FUNCTIONAL REQUIREMENTS

2.3.1 Python

- The application is developed in python and run in a python IDE with OpenCV and other libraries.

2.3.2 OpenCV

- It is a library that is necessary to pre-process the input image and convert it into gray scale.
- In our project we use the CV2 version 4.1.0 which helps in finding the meaning information from an image as it works well on computer vision problems.

2.3.3 Keras

- It is an Open source library in python built on top of the TensorFlow which is used in developing deep learning models
- In our project to build the neural network model CNN (Convolutional Neural Network), we use Keras and the version that we used is Keras-2.3.1

2.3.3.1 Keras-preprocessing

- This module is used for data preprocessing and it is present inside the keras library for deep learning and the version that we used in our project is Keras-preprocessing-1.1.0

2.3.4 NumPy

- This library is used to process some mathematical operations as in our project we display the output in the form of Confusion Matrix thus to work with matrices we use this library Numpy-1.16.2

2.3.5 Scikit-learn

- This is a machine learning library and as in our project we work with classifications and numerical operations we use this library Scikit-learn with the version 0.20.2.

2.3.6 TensorFlow

- This is an open source software framework for developing neural

networks. In our project we use the version the Tensorflow-1.14.0.

2.3.7 Matplotlib

- This is used as a plotting library for mathematical operations and the version used is Matplotlib-2.2.3.

2.3.8 Tkinter

- This is used for GUI purpose under Matplotlib in our project to display the detected features from the detection methods (SIFT, SURF, ORB and HOG) we use Tkinter.

2.4 NON-FUNCTIONAL REQUIREMENTS

2.4.1 Reliability

The system must be reliable. The system must be able to correctly classify the elephant image.

2.4.2 Usability

The system must be easy be simple and the interface must be user friendly so that the user can easily learn to use the application.

2.4.3 Performance

The system must be fast and respond within a short period of time.

2.4.4 Efficiency

The system should be efficient and consume minimum power and space.

2.4.5 Availability

The system should detect the elephant correctly as for any kind of proper input received at any time of request, provided the system has requirements satisfied.

CHAPTER 3

LITERATURE SURVEY

Matthias Zeppelzauer [1] in EURASIP Journal on Image and Video Processing has proposed a color model for an automated elephant detection system. The color model is learned from the training images which consists of shape, color and cue. As the data set being investigated consists of videos in different environments, there are limited robust constraints. Hence, this color model uses basic clues like color and texture to detect elephants in the test images. Though shape is a useful clue as elephants have trunks, since elephants maybe photographed in different positions ii is not applicable. Texture maybe helpful to some extend but as the distance increases the visibility of texture decreases with distance. As the colour model uses temporal clues, it is unaffected by background images and lighting. The advantages of using this method are that it can easily detect groups of elephants and elephants performing a variety of activities and is size invariant. It is scalable and elephants are detected irrespective of the distance from the camera. The training data is preprocessed with color segmentation and a color model is generated. The color model is then applied to the test data to detect the points which may contain the elephants. These spatial points are then temporally tracked to get the spatial temporally tracked points. These points are then validated by performing spatio-temporal feature detection to get the final points containing the elephants. It is easily adaptable and can be easily reused for detection of other animals by changing the training dataset.

SJ Sugumar and R Jayaparvathy [2] in The Scientific World Journal have proposed an unsupervised elephant image detection system (EIDS). It is based on cameras mounted on trees that send images to a base station where it is processed by comparing with the dataset images and an SMS is sent to forest officials if an elephant is detected. The EIDS uses Har wavelet for feature detection by using multilevel coefficients. In Har wavelet, the har wavelet decomposition is calculated by calculating the average and difference between the odd and even data sampled iteratively. The images in the dataset are then classified

according to features using k means clustering using Euclidean distance. The test image is compared with dataset and arranged in descending order of similarity using the proposed distance metric. If there are more than five matched images containing elephants, it is classified as containing elephant. A proposed Optimized distance metric may be used instead of Euclidean and Manhattan distances for finding the similarity between images and arranging images in order. Unlike the Euclidean distance which calculates the square root of sum of squares of difference between the centroid and pixel points, the optimized distance is the modulus of the sum of cubes of difference between the centroid and pixel points. It is more reliable, efficient and takes less time.

Angela S Stoeger [3] in BMC Research notes discussed an early elephant warning and monitoring system. This system is based on the infrasonic sound that elephant makes. This method has a significant advantage as the elephant rumbles travel several kilometers and hence elephants can be detected even if they are out of sight. Visual detection is also used which offers high accuracy in nearby cases. The input module includes both acoustic and visual detection. Acoustic detection can be used to determine attributes like call type, age classification and gender classification. Visual detection is used to identify motion patterns and recognize the activity of the elephant. Both acoustic and visual features are used for multimodal detection which estimates the distance, direction and group size. In acoustic detection, image processing on spectrogram of frequency contours is performed to separate the elephant rumbles from other noises in the environment. Then a linear SVM classifier is used to classify the sound as elephant rumbles or not. The visual detection uses a binary SVM model trained from the color data in the image. The detected candidates are then temporally tracked to avoid false positive cases and the final candidates are classified as elephants. Combining both acoustic and visual detection, this method has higher accuracy than existing approaches.

Jerline Sheebha Anni Dhanaraj and Arun Kumar Sangaiah [4] in Journal of Experimental and Theoretical Artificial Intelligence evaluated a Boundary Sense Deep Learning Architecture (BSDL) for elephant feature learning and classification. This method has a background subtraction module where images are preprocessed and hence important visual regions i.e., the regions containing the elephants are highlighted. Elephant detection with Multi view dataset has improved the efficiency of using back-propagation. BDSL

introduces preliminaries which simplifies the classification. This method achieves a higher accuracy than processing a raw image. It also reduces time and achieves 97% accuracy.

Isha Dua, Pushkar Shukla and Ankush Mittal [5] proposed a vision-based human-elephant collision detection system which is primarily based on motion detection. It uses video cameras installed in places with high human elephant interventions. Whenever a motion is detected in the video, the image is captured and the object in the image is classified as elephant or not. The features are extracted using PHOG and SVM classifier was used to classify the detected features. When applied to static images, an accuracy of 85.29% was obtained.

Haering N, Qian R, Sezan M [6] in IEEE Trans. Circuits Syst presented a method for hunt scenes detection by using color and texture features in each pixel. It is a three-level approach which involves analysis, summarisation and inference. At the first level, the video is divided into shots and global motion estimation is performed. The colour and texture features are also detected. At the next level, the detected motion areas are classified as moving- object regions by a neural network. The network uses the colour and texture features extracted at the lower level, and performs a crude classification of image regions into sky, grass, tree, rock and animal. This level also generates shot summaries which describe each individual shot in terms of intermediate level descriptors. At the next level, an event inference model is used to detect if animals are present and if hunting occurs based on the summary and using domain specific knowledge. Also, these features are used by ANN to classify each pixel as belonging to animal class or not. Image segmentation is done before processing to reduce the time and cost of processing each pixel. As it does not involve motion clues, resting animals can also be detected.

CHAPTER 4

ANALYSIS AND DESIGN

4.1 USE CASE DIAGRAM

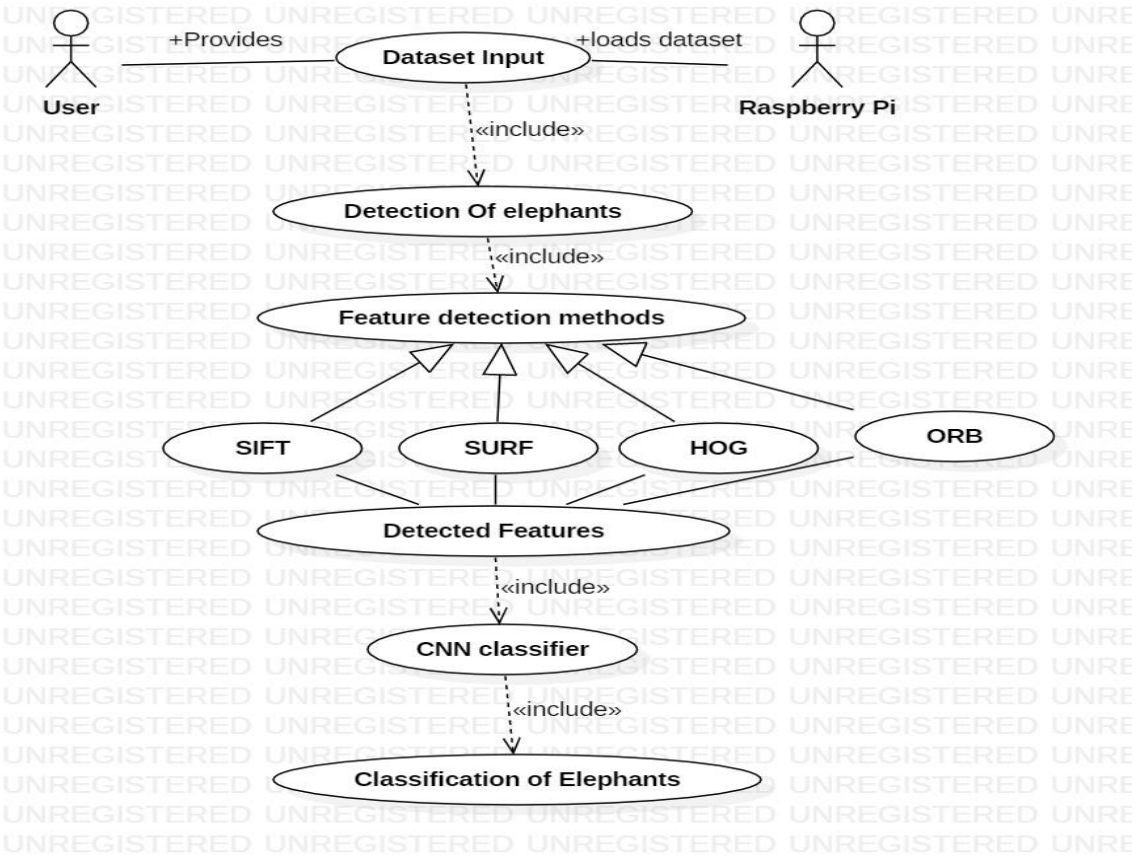


Fig 4.1 Use Case Diagram

FEATURE	CONSTITUENTS
ACTORS	• User • Raspberry Pi
USE CASES	• Dataset Input • Detection of Elephants • Feature Detection Methods • SIFT, SURF, HOG and ORB • CNN classifier • Classification of Elephants

Table 4.1 Use Case Description

4.2 Activity Diagram

4.2.1 Scale Invariant Feature Transform(SIFT)

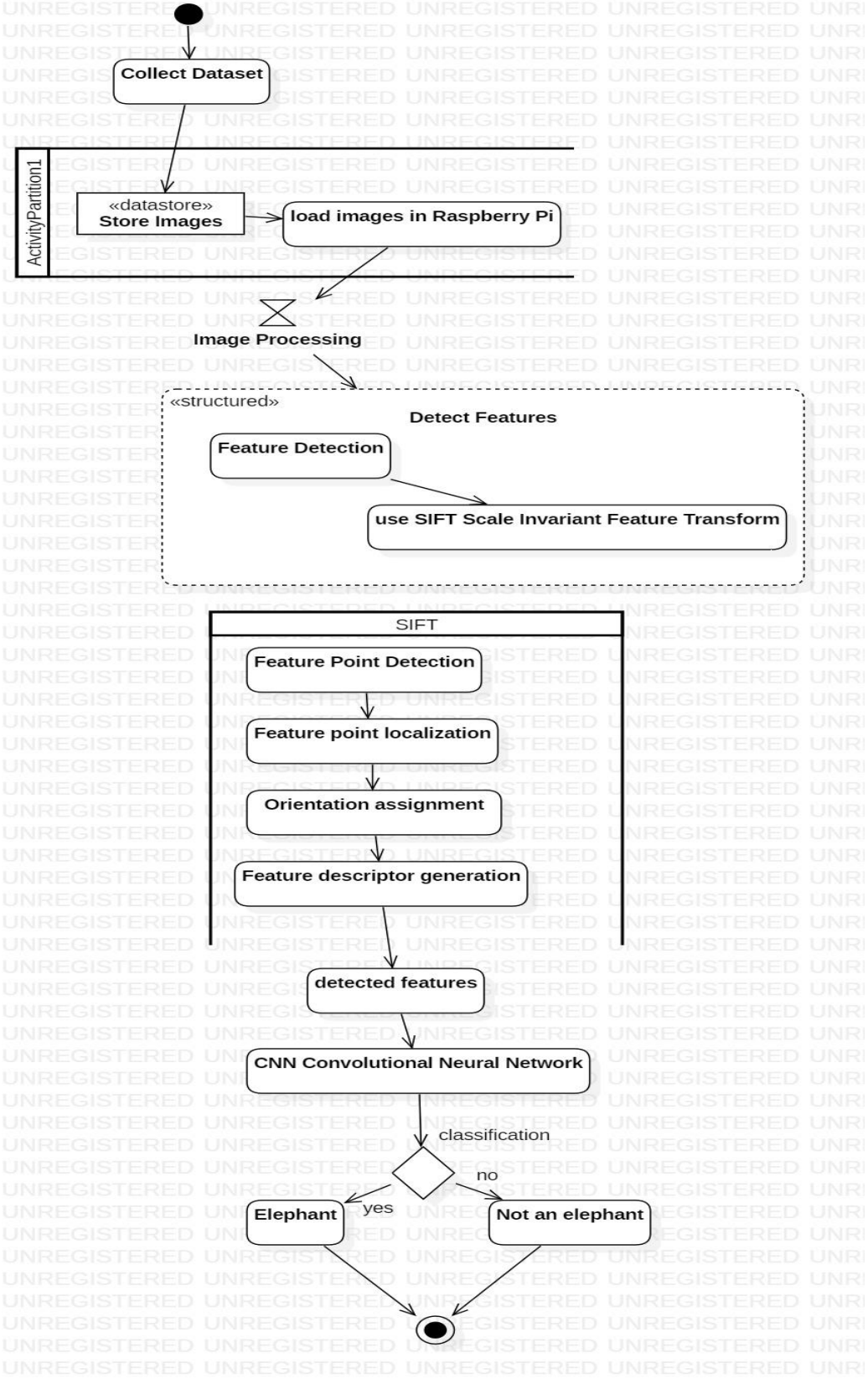


Fig 4.2 Activity Diagram for SIFT

4.2.2 Speeded Up Robust Features (SURF)

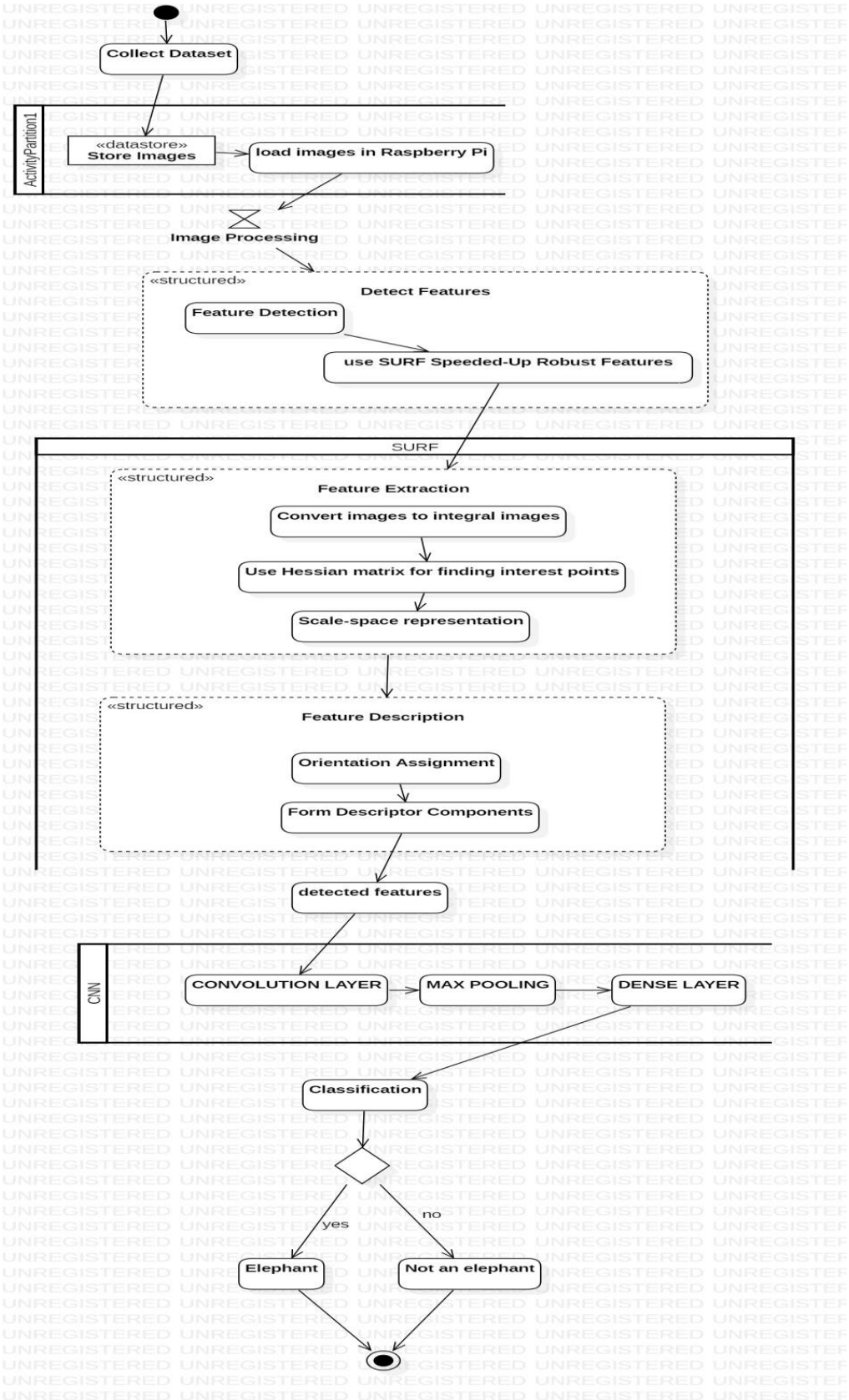


Fig 4.3Activity Diagram for SURF

4.2.3Oriented FAST and rotated BRIEF(ORB)

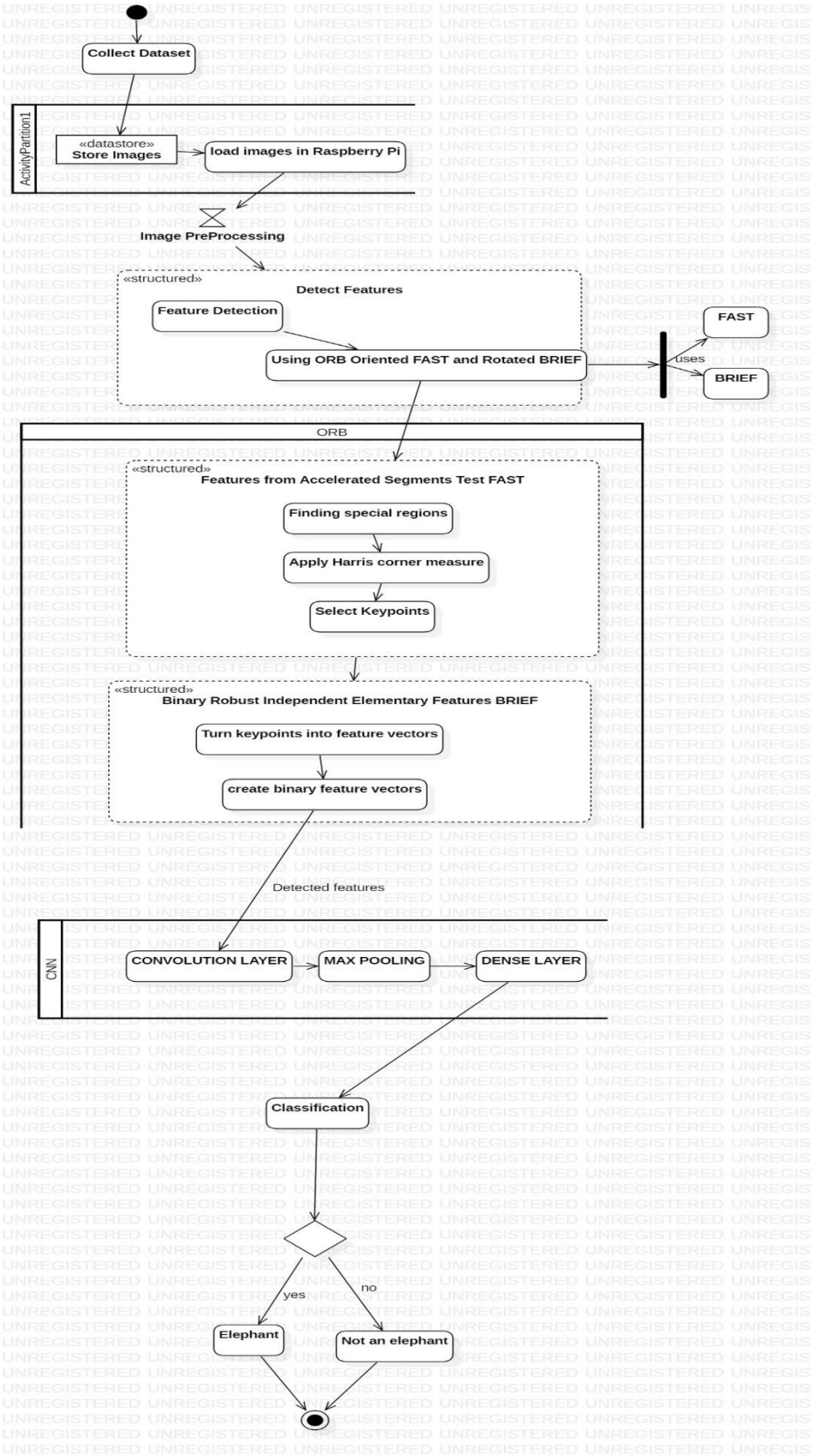


Fig 4.4Activity Diagram for ORB

4.3 FLOW DIAGRAM

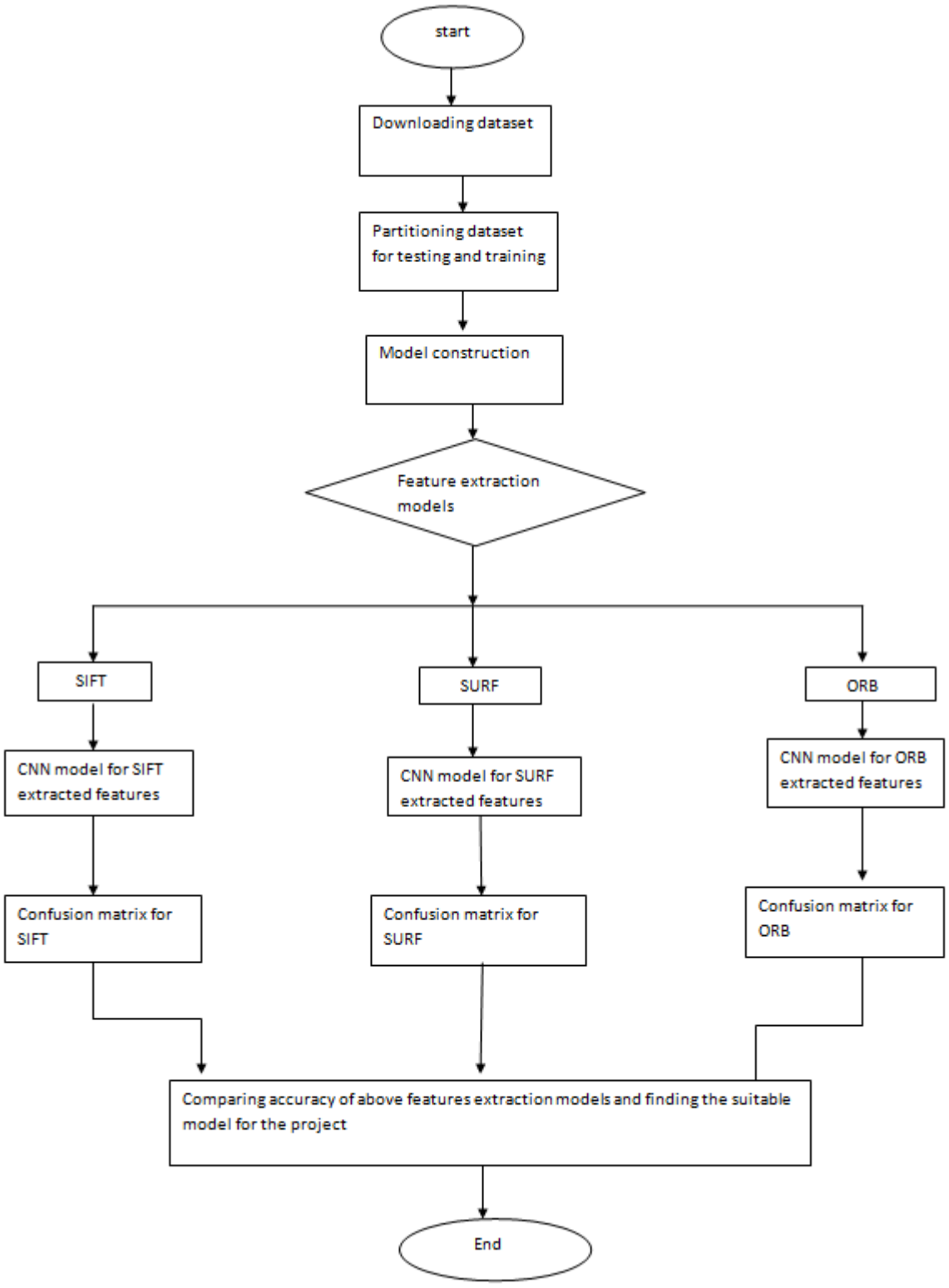


Fig 4.5 Flow Diagram

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 DATASET

5.1.1 Predefined Dataset

The elephant image dataset is predefined which has been used and is collected from Kaggle. Dataset consists of huge number of animal images, in which we are considering only about 500 images. The dataset folder consists of training and testing set of images. The training folder contains both elephants and other animals' images which is trained and tested using the testing setwhich consists of 10 images of elephant and 15 images of another wild animals.

5.2 FEATURE DETECTION:

Feature detection is an essential pre-processing step in many face-related applications. The accuracy rate of these applications depends on the inerrancy of the featuredetection step. In addition, feature detection is an important research problem for its role as a challenging case of a more general problem in otherwords, object detection.

The most straightforward and common example of this problem is the detection of a single face at a known scale and orientation. This is a nontrivial problem, and no method has yet been found that can solve this problem with 100% accuracy. Thefeatures detection technique is classified into four categories.

5.2.1 Scale Invariant Feature Transform:

It is used in applications involving scaling of images, it detects all the pixels of a scaled image. The advantage of this detector that it is used to detect features independent of size or scale of images.

It transforms the image into invariant scale points and also generates many features for even a small object. The feature detection using SIFT is done using following procedures

5.2.1.1 Scale-Space Extrema Detection:

Initially ,a pixel of an image is blurred using Gaussian blurring and it is checked with 8 other neighbor pixels of image as well with 9 pixels of previous as well as next scale of an image .Thus it forms the different octaves and if the taken pixel is an local extrema then it is considered as an potential key point.

5.2.1.2 Key point Localization:

In this step, thekey point which is less than contrast threshold value(default-0.03) and key point which is greater than edgethreshold value(default-10) is removed and only the strong interest points is selected.

In this step the edges of an image are also removed used eigen values and their ratios, since the main aim of SIFT being used here is for feature detection.

5.2.1.3 Orientation Assignment:

For an instance of an image orientation histogram is created for keypoint.it takes patch of image around the key point and calculates the magnitude and orientation of edge around every pixel in that patch of image.

The edge magnitude is calculatedby previous and next pixel along the X coordinates and also the previous and next pixel along the Y coordinate for each pixel in the patch of image. Thus, this step gives the key points with same location but with difference in directions by selecting the highest peak in histogram.

5.2.1.4 Key point Descriptor and Key point Matching:

It creates the descriptor (number of key points obtained from above steps*128bins). the 128 bins are created by creating 16*16 neighborhood around the key point which is divided into 16 subblocks of size 4*4. Thus, each of the 16 subblocks contains 8 bin orientations.

Finally, the key points are matched along with neighbor features between the images of testing and training dataset images, and if the matching value is greater than or equal to 0.8, then the image is to matched correctly.

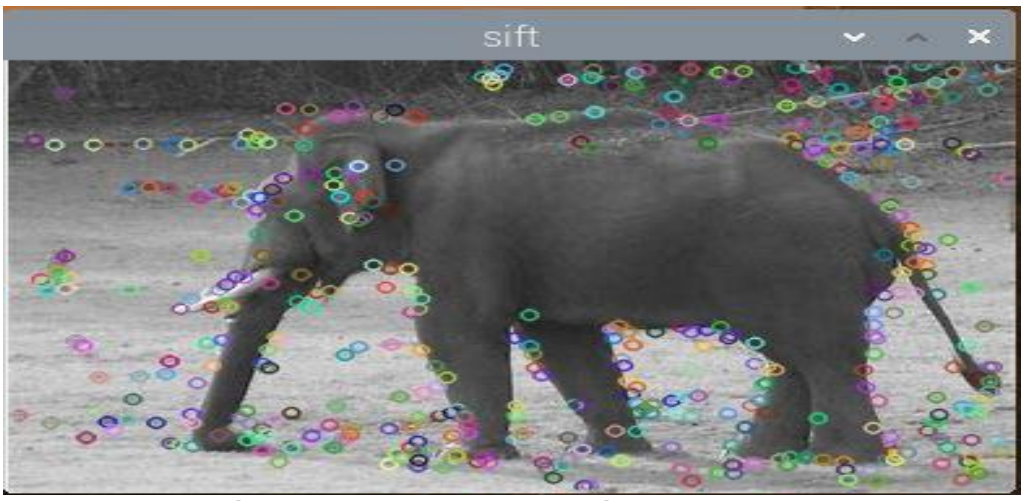


Fig 5.1 SIFT Feature Extraction

5.2.2 Speed-Up Robust Feature:

It is similar to SIFT, but the Key points are identified much faster than SIFT. The image detection steps are similar to SIFT but with different methods. SURF algorithm includes the feature extraction, feature description and feature matching steps.

5.2.2.1 Feature extraction:

This step is used to detect the interesting points as like SIFT but by using the DoG(Difference of Gaussian) instead of **Gaussian** blurring in SIFT. The same steps are done to get the interesting points as like by removing the edge points and low contrast points in SIFT. In SURF the scale invariance of image is obtained by using the various scale of image.

The points are chosen based on maximum changes around the points and its uses box filters for approximation. So, the box filter convolution can be easily achieved and it can be done for images of different scales parallelly

5.2.2.2 Feature Descriptor:

The key point descriptor step in SURF is based on Haar Wavelet response. It is done by dividing the neighboring pixels around the interest points into subregions, responsesare detected and then the responses are finally added to create the descriptor. Thekey point descriptor is obtained by sum of all response.

If the size of the descriptor is low, then the computation and matching speed is reduced. The orientation is calculated based on the need of applications. If the rotation invariance is not required for the image, the need of finding orientation is also not required. Thus, it speeds up the process as compared to SIFT.

5.2.2.3 Feature Matching:

In this step, the features are matched only to identify that the images have same type of contrast. This step is done quickly since the result of above previous steps are with accuracy of minimal information

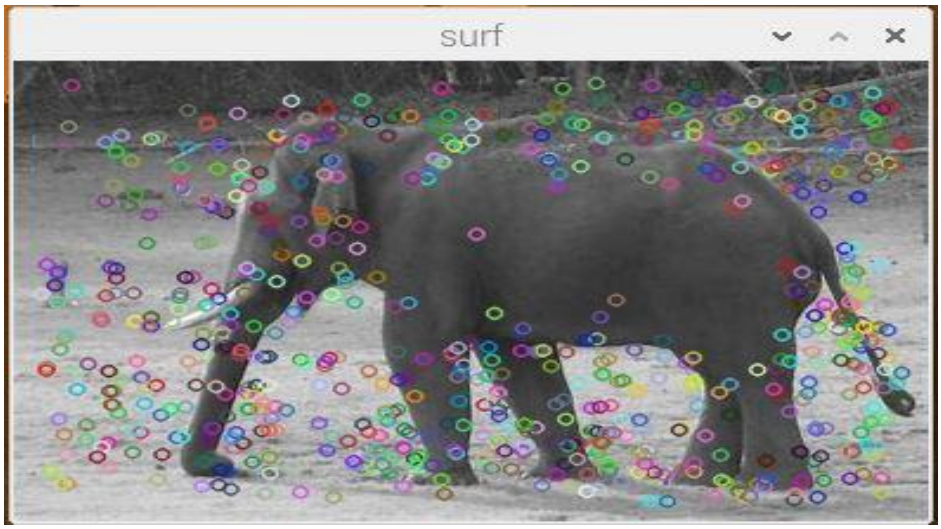


Fig 5.2 SURF Feature Extraction

5.2.3ORB (Oriented FAST and rotated BRIEF):

It is a fast and efficient alternative to SIFT and SURF as it is based on FAST key point detector and BRIEF (Binary Robust Independent Elementary Features) descriptor.

5.2.3.1 Feature extraction:

It uses the FAST key point detector to detect the key points and it applies the Harris corner measure to identify the peak points from the detected key points .In orb ,it creates the multiscale image pyramid of an image .Then the key point is selected by comparing the brightness of taken pixel with 16 other pixels of an images.

If more than 8 pixels are brighter than the taken pixel, then that pixel is selected as key point. This is done by FAST detector. Thekey points are then converted into binary feature vector which is used to detect the objects. ORB doesn't create the descriptor, so for rotation invariance images it calculates the intensity of centroid of the patch of image. Thus, the orientation is calculated from the direction of vector from corner of the patch toward the centroid calculated.

5.2.3.2 Feature Descriptor:

ORB uses BRIEF descriptor for feature descriptors. ORB is not intended for high performance of descriptors. Thus, ORB steers the BRIEF descriptor based on the orientation calculated on previous step. For every key point, it provides the matrix that contains the coordinates.so based on the rotation of an image, the rotation matrix is found by using the

orientation. Thedescriptors are generated as long the rotation matrix is found consistent with rotation of the image.

5.2.3.3 Feature Matching:

In this step the descriptors are used to match the image using the multi-probe LSH (Locality Sensitive Hashing). The Locality Sensitive Hashing is used to solve search of nearest neighborexactly or approximately. The multiprobe LSH is a proposed technique of LSH, that is mainly proposed to reduce the number of hash table required by LSH.

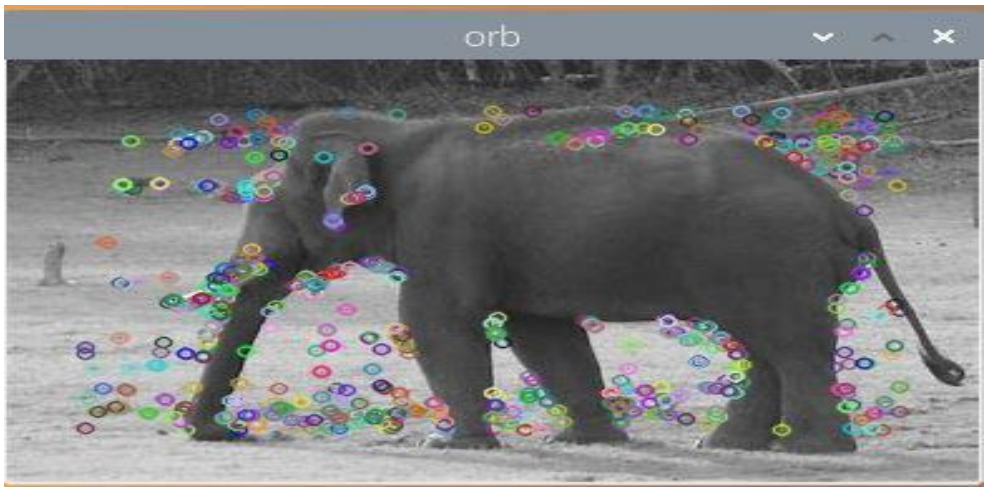


Fig 5.3 ORB Feature Extraction

5.2.4 HOG (Histogram of oriented gradients):

It is similar to SIFT but it represents object as single feature vector other than set of features. It creates only one histograms of oriented gradients for the entire image. It takes each 8*8 pixel of an image and compute the gradient vectors (8*8 as 64 which is finally reduced to 9 by using angular bins) and it is represented as histograms.

The features vector is created by sliding window detector and then creates descriptor for each position in an image. Thus, the final descriptors are combined into a single feature vector. HOG is mostly used with SVM (support vector machine) classifiers to determine the object. Thus, we are not using it with CNN due to its inefficiency reasons with CNN but we are keeping the HOG descriptors as a base for other detectors to work better.

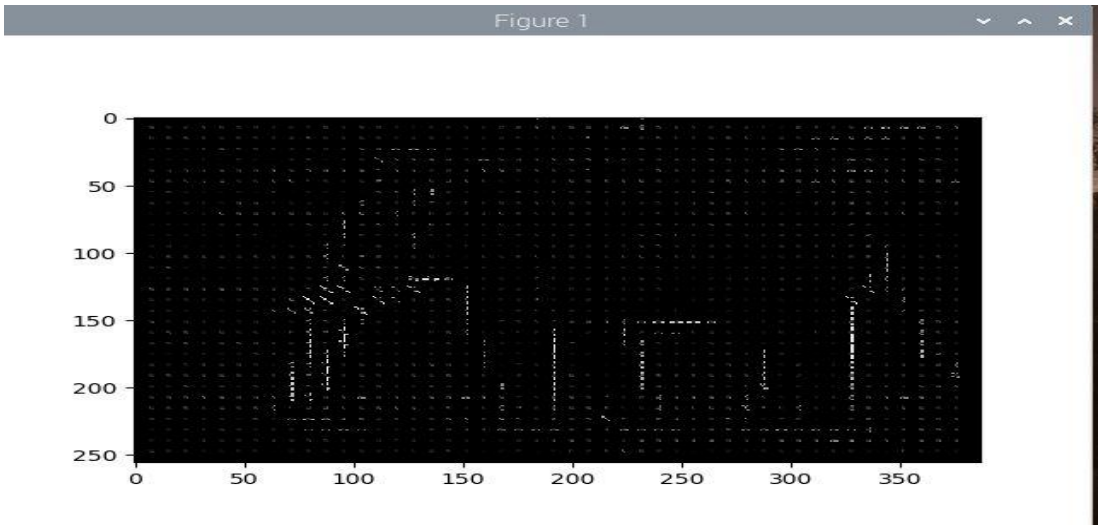


Fig 5.4 HOG Feature Extraction

5.3CLASSIFICATION:

5.3.1 CNN (Convolutional Neural Network):

CNN is a feed-forward artificial neural network in which the connectivity between the neurons are similar to virtual cortex in animal. InCNN each neuronis connected only to small part of others neuronsunlike fully connected network thus it ismainly used in image classification since it reduces the overweight. Here in our project we are using features that are extracted before using features extraction techniques. The CNN has three layers namely

- Convolutional Layers
- Pooling Layers
- Dense Layers

The features which are extracted from the above feature extraction steps are passed through the CNN layers to get the accuracy of matching.

5.3.1.1 CONVOLUTIONAL LAYER

The convolutional layer is the main component of CNN architecture. Convolution is basically a mathematical operation. Here in our project the initial convolution is applied on input image size of 64*64 with 32 layers of filters each of size of 3*3 matrix.

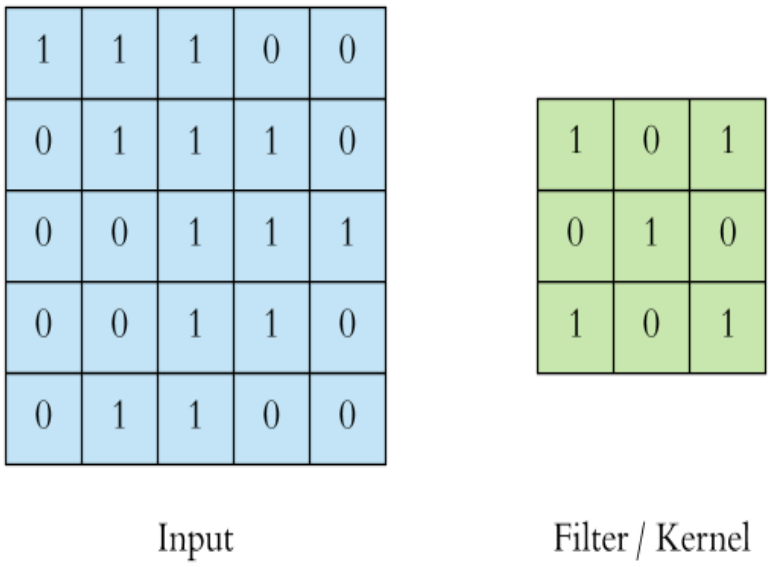


Fig 5.5 Input and Kernel Filter

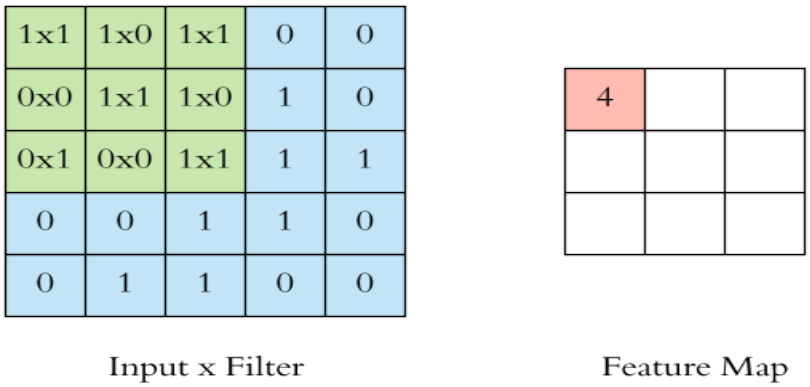


Fig 5.6 Convolution Operation

5.3.1.2 POOLING LAYER

Pooling layer follows the convolution layer. This is mainly done to reduce dimensionality so that training time is reduced, that is to reduce the number of weights to be learnt. Pooling is of two types namely Max pooling and Average pooling. Here in our project we used max pooling of size 2*2, which only takes maximum values so to reduce the weight

5.3.1.3 DENSE LAYER

After the convolution and pooling layers, the dense layer or flatten layer is added in the architecture. This layer is like the fully connected layers in Artificial Neural Networks. The output of the final pooling layer is flattened from 2D vector into a single vector. Here in this project the

final dense layer shows the information of convolution from previous steps. It shows the desired output at the final (detection of elephant or not.)

5.3.2 CNN ARCHITECTURE:

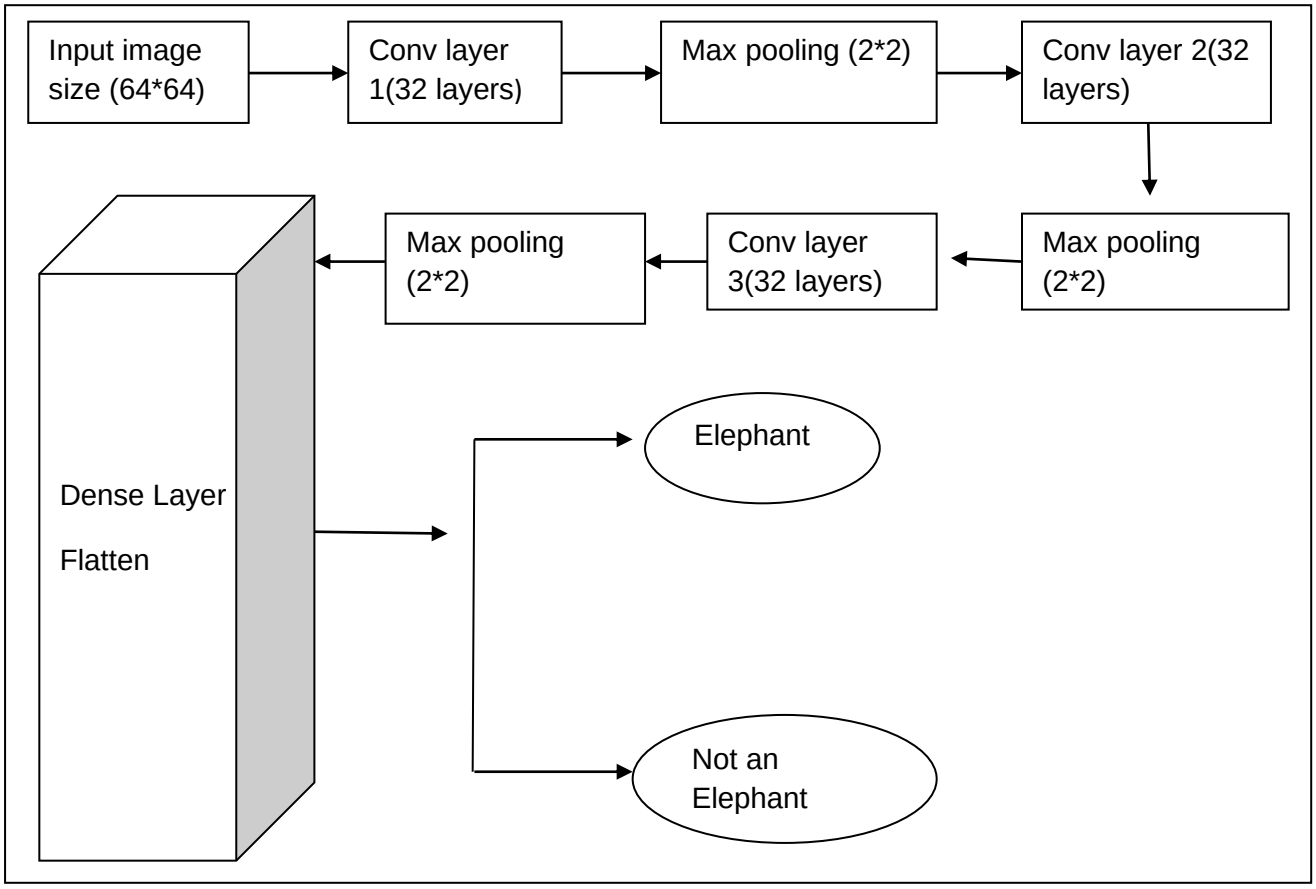


Fig 5.7 CNN Architecture

The Structure of the architecture is as follows:

- First is the input image which has the dimension of 64*64.
- Then there is a convolutional layer with 32 filters each of size 3*3.
- The convolutional layer is followed by a pooling layer of size 2*2.
- The above convolutional and max pooling steps are again continued for further 3 times.
- The final is a dense layer which takes the information from the above steps to flatten the final result to 0ne dimensional.
- The last dense layer output shows the final classification of an image.

5.4 EVALUATION:

In our projectthe evaluation of image matching is to be done using the convolution matrix. it visualizes the performance of CNN classifier on the testing dataset. The final result of our project is shown on confusion matrix of correct as well as incorrect matches. The performance can be measured using the following terms

True Positive:

It is identifying the correct value as correct one. In our project the actual elephant can be identified as elephant.

True Negative:

It is identifying the incorrect value as an incorrect one.

False positive:

It is identifying the incorrect value as a correct value.

False Negative:

It is identifying the incorrect values as an incorrect value

The confusion matrix can be done using anyone of the following performance measures:

Accuracy:

In our project it is the count of how many images are correctly identified as elephants among the given testing set of images of both elephant and other animal images.

Recall:

It is the count of how many elephants are correctly identified as elephants from the given set of elephant images.

Precision:

It is the count of images that are elephants from the given set of images that are actually elephants as well as images that are wrongly considered as an image.

F-measure:

It is the measure of both precision and recall at the same time.

The final result of our project is shown by theconfusion matrix.

CHAPTER 6

TESTING

6.1 TESTING

Testing the system is an essential phase since it makes sure of the reliability and the quality of the application. Testing is required for an effective performance of the system. It's important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development. This chapter deals with the tests that we conducted for the project. It also lists the test cases that we implemented of the project.

Acceptance testing is related to the business requirements testing. Here testing is done to validate that the business requirements are met in the user environment. Compatibility testing and sometimes non-functional testing are also done in this phase.

SNO	MODULE NAME	TEST SCENARIO	TEST CASE	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/ FAIL
6.1.1	Input Module	Loading dataset into the Raspberry Pi for Image Processing	Load the dataset	Dataset loaded by the User	Images are loaded successfully.	Images are loaded successfully.	PASS
		Loading dataset into the Raspberry Pi for Image Processing	Load the dataset	Dataset loaded by the User	Images are loaded successfully.	Images are not loaded successfully due to insufficient memory space in the SD card	FAIL

6.1.2	Powering the Raspberry Pi	Providing the power to the Raspberry Pi by means of USB cable.	Works Successfully with the given Power.	Provide power to the Raspberry Pi by USB	Working Successfully with the connected System	Working Successfully with the connected System	PASS
			Works Successfully with the given Power	Provide power to the Raspberry Pi by USB	Working Successfully with the connected System	Not Working Successfully with the connected System	FAIL
6.1.3	HDMI Connection to the Raspberry Pi	If working with the Desktop Computers for displaying the Raspberry Pi window(Raspbian OS) connect the Raspberry Pi with a HDMI Cable	Displays the screen to the monitor.	Connect HDMI with the Raspberry Pi	HDMI works successfully in order to connect the monitor to the Raspberry Pi for Displaying Purpose	HDMI works successfully in order to connect the monitor to the Raspberry Pi for Displaying Purpose	PASS
6.1.4	Feature Detection	Detecting features of the image is a Pre-Processing step for getting efficient results	Detection of features from the input image	Images from the Dataset	Detects the features successfully for the given image	Detects the features successfully for the given image	PASS
		Detecting features of the image is a Pre-Processing step for getting efficient results	Detection of features from the input image	Images from the Dataset	Detects the features successfully for the given image	Features are not successfully detected for the given image.	FAIL

6.1.5	Feature Detection using different methods	Use SIFT,SURF, ORB methods for detecting the features of the image	Identification of the keypoints and descriptors	Key point identification of the images using the detection methods(SIFT, SURF, ORB)	Extracts the features successfully for the given image	Extracts the features successfully for the given image	PASS
6.1.6	Classificati-on	Train the CNN Model with the positive images (elephant images) and negative images	Classification of the input image as elephant or not	Positive images and negative images as input	Classified the image as elephant or not successfully	Classified the image as elephant or not successfully	PASS
		Train the CNN Model with the positive images(elephant images) and negative images	Classification of the input image as elephant or not	Positive images and negative images as input	Successfully classified the input image as elephant or not.	Not successfully classified the input image as elephant or not.	FAIL

Table 6.1Test cases

6.2 EXPERIMENTAL RESULTS

In our project the accuracy score is calculated based on the confusion matrix along with an overall report depicting the scores of precision, recall, f1-score and support.

```
Confusion Matrix for SIFT :
[[8 2]
 [1 9]]
('Accuracy Score :', 0.85)
Report :
              precision    recall  f1-score   support

           0       0.89      0.80      0.84        10
           1       0.82      0.90      0.86        10

   micro avg       0.85      0.85      0.85        20
   macro avg       0.85      0.85      0.85        20
weighted avg       0.85      0.85      0.85        20
```

6.1 Accuracy report of SIFT

```
Confusion Matrix for SURF :
[[8 2]
 [3 7]]
('Accuracy Score :', 0.75)
Report :
              precision    recall  f1-score   support

           0       0.73      0.80      0.76        10
           1       0.78      0.70      0.74        10

   micro avg       0.75      0.75      0.75        20
   macro avg       0.75      0.75      0.75        20
weighted avg       0.75      0.75      0.75        20
```

6.2 Accuracy report of SURF

```
Confusion Matrix for ORB:
[[8 2]
 [3 7]]
('Accuracy Score :', 0.75)
Report :
              precision    recall  f1-score   support

           0       0.73      0.80      0.76        10
           1       0.78      0.70      0.74        10

   micro avg       0.75      0.75      0.75        20
   macro avg       0.75      0.75      0.75        20
weighted avg       0.75      0.75      0.75        20
```

6.3 Accuracy report of ORB

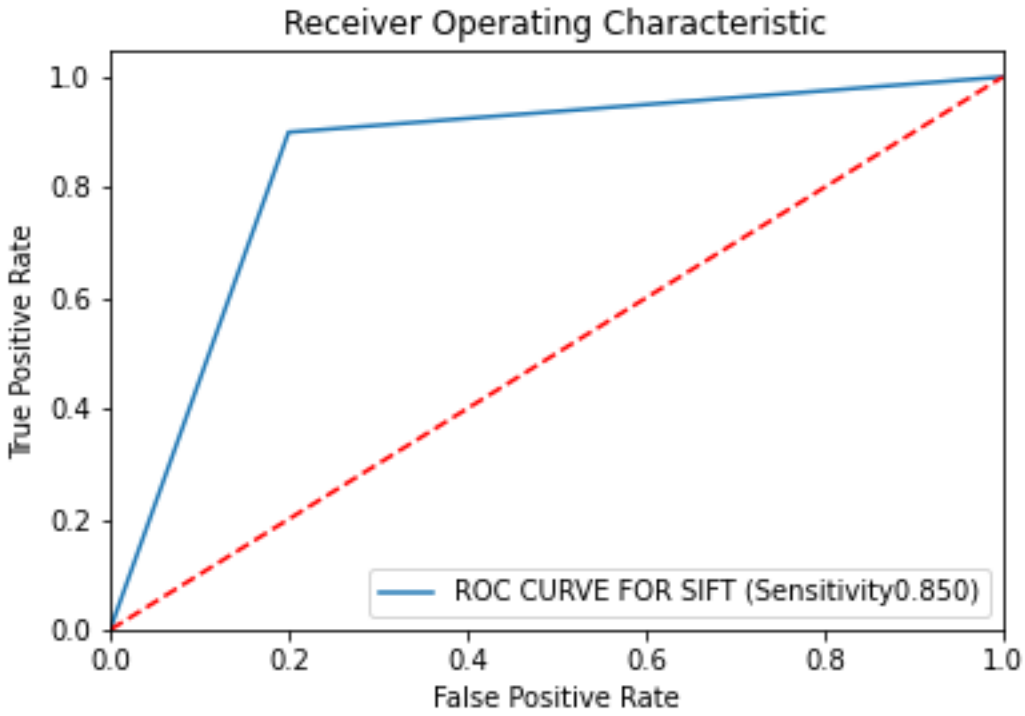
6.3 PERFORMANCE MEASURE USING ROC CURVE

In the classification problems the performance should be measured with varies threshold values the threshold value is totally dependent on the project. To measure the performance of our project we are using the ROC curve.

Receiver Operating Characteristic plot is used to visualise the performance of a binary classifier. It gives the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) at classification thresholds.

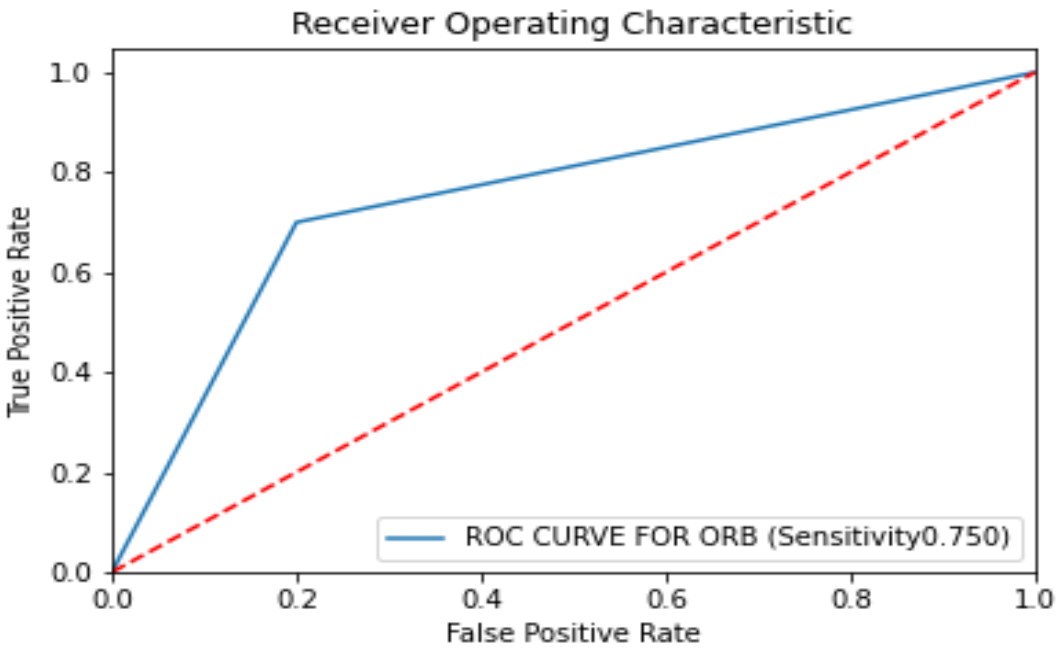
In our project we have set the threshold as 0.90 for SURF, ORB and 0.75 for SIFT depicting the ROC curve.

The sensitivity and specificity rates differ based on the model that is for SIFT-CNN , SURF-CNN and for ORB- CNN.

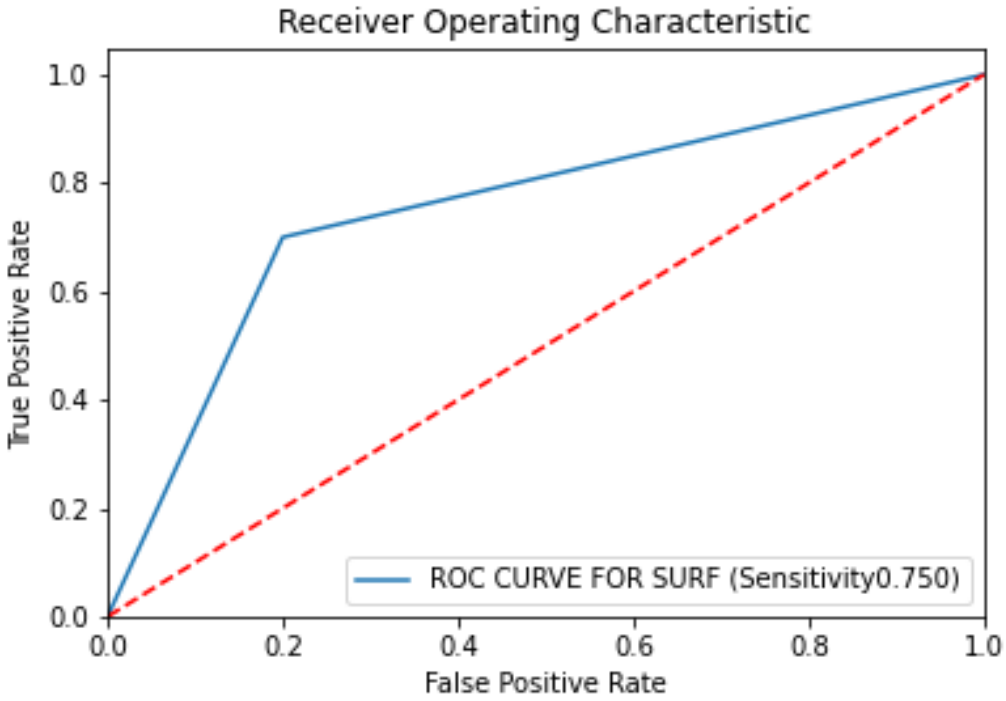


6.4 ROC CURVE FOR SIFT

The sensitivity rate is also called as the true positive rate (TPR) is placed on the y-axis and the false positive rate that is 1-specificity rate is placed on x-axis and for the SIFT detector with the CNN classifier the sensitivity value is 0.85. The sensitivity value for SURF and ORB are 0.75.



6.5 ROC CURVE FOR ORB



6.6 ROC CURVE FOR SURF

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

Thus, the implementation of elephant detection has been done using SIFT, SURF, ORB and HOG for feature extraction and classified using CNN in raspberry pi and an overall accuracy of 0.85 has been achieved. The confusion matrix, precision, recall and f1-score have also been generated for the test images. This implementation can be further improved and made real-time using cameras for real-time input instead of test images and real-time results can be obtained. As raspberry pi is used, the processing can be faster and implemented locally as it requires less power compared to other processors. The training images are taken from KAGGLE and then tested with random images and an accuracy of 0.85 has been achieved. Thus, elephants can be detected in forest borders as an alert system or in forests for research purposes easily without manual labour and with minimum effort and power consumption.

7.2 FUTURE SCOPE

The future implementation for our project includes installing cameras in forest border areas and places with high human elephant interventions. The cameras can be connected to a local base station using RF or other wireless connections. Videos may be continuously recorded whenever there is a motion and sent to the base station. The base station may split the video into images and process the images to detect if an elephant is in the image. If an elephant is found, then the forest officers can be alerted by an SMS or other methods to inform that an elephant is approaching the urban areas so that necessary measures can be taken.

The project can be further improved by including other feature selection and detection methods to improve the accuracy. It can also be used for scientific and research purposes where the behaviour patterns, movements and activities of elephants need to be studied. The cameras may be installed in forests to record videos for several months. As this may contain terabytes of videos, separation of parts containing only elephants can be tedious. This can be solved by using our elephant detection feature. The recorded video may be processed and the parts where elephants are detected can be stored while the other parts may be deleted.

This saves storage space and also time and human labour for the scientists.The detection can also implement other feature extraction and classification methods in the future. For example, ORB can be combined with SVM and detections can be made. The detections from the different methods can be fed to a neural network with the accuracy of the corresponding method as the initial weight for the corresponding edges and trained using training images to detect elephants to further improve the accuracy of detection.

BIBLIOGRAPHY

[1] https://jivp-eurasipjournals.springeropen.com/articles/10.1186/1687-5281-2013-46- - Research paper for Understanding

[2] https://www.hindawi.com/journals/tswj/2014/393958/

[3] https://bmcresnotes.biomedcentral.com/articles/10.1186/s13104-015-1370-y

[4] https://www.tandfonline.com/doi/full/10.1080/0952813X.2018.1552316

[5] https://www.semanticscholar.org/paper/A-vision-based-human-elepahant-collision-detection-Dua-Shukla/1ee65c6cda1c84797e1706de9755b67147d709c5

[6] https://www.researchgate.net/publication/3308186_A_semantic_event-detection_approach_and_its_application_to_detecting_hunts_in_wildlife_vide

[7] https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf -To study about ORB

[8] https://towardsdatascience.com/sift-scale-invariant-feature-transform-c7233dc60f37 - To study about SIFT

[9] https://medium.com/data-breach/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e - To study about SURF

[10] https://www.instructables.com/id/How-to-connect-raspberry-pi-to-laptop-display/ - To make connection with the Raspberry Pi

[11] https://www.youtube.com/watch?v=upY4Fusi4zI – Video for connecting the Raspberry Pi with Laptop

[12] https://towardsdatascience.com/image-classification-python-keras-tutorial-kaggle-challenge-45a6332a58b8 - To learn about CNN

APPENDIX

CODE

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Activation,Dropout,Flatten
from keras.layers import Dense
from keras import backend as k
from keras.preprocessing import image
model1 = Sequential() //initializing model
from keras.preprocessing.image import ImageDataGenerator
img_width,img_height=64,64
if k.image_data_format()=='channels_first': //checks each image size and resized to (64,64)
  input_shape=(3,img_width,img_height)
else:
  input_shape=(img_width,img_height,3)
train_datagen = ImageDataGenerator(rescale = 1./255,
shear_range = 0.2,
zoom_range = 0.2,
horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255) //generator for training images
training_set = train_datagen.flow_from_directory('/home/pi/elephant/positive/training', //path to training images
target_size = (img_width, img_height),
batch_size = 15,
class_mode = 'binary')
```

```
//generator for testing images

test_set = test_datagen.flow_from_directory('/home/pi/elephant/positive/validation', //path to test images

target_size = (64, 64),

batch_size = 15,

class_mode = 'binary')

model1.add(Conv2D(32, (3, 3), input_shape =input_shape, activation = 'relu')) //first convulation layer

model1.add(MaxPooling2D(pool_size = (2, 2))) //operation for spatial data

model1.summary()

model1.add(Conv2D(32, (3, 3), input_shape =input_shape, activation = 'relu')) //second convulation layer

model1.add(MaxPooling2D(pool_size = (2, 2)))

model1.add(Conv2D(32, (3, 3), input_shape =input_shape, activation = 'relu')) //third convulation layer

model1.add(MaxPooling2D(pool_size = (2, 2)))

model1.add(Flatten()) //converts 3D feature to 1D feature vector

model1.add(Dense(units = 64, activation = 'relu'))

model1.add(Dropout(0.5)) //dropout is to prevent overfitting

model1.add(Dense(units = 1, activation = 'sigmoid'))

model1.summary()

model1.compile(optimizer = 'rmsprop', loss = 'binary_crossentropy', metrics = ['accuracy']) //to compile the model constructed

from PIL import ImageFile

ImageFile.LOAD_TRUNCATED_IMAGES=True // to load all images

model1.fit_generator(training_set, //testing the model

steps_per_epoch = 300,

epochs = 15,

validation_data = test_set,

validation_steps =20)
```

```
import numpy as np

from keras.preprocessing import image

count=0

count1=0

count2=0

count3=0

import cv2

import numpy as np

from Tkinter import Tk

from tkFileDialog import askopenfilename

import os,sys,time

from PIL import Image

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score

from sklearn.metrics import classification_report

actual=[0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1] //initialization of elephant and others images for confusion matrix

predicted=[]

Tk().withdraw()

for i in range(1,11):

   filename='/home/pi/elephant/testing/elephant/'+str(i)+'.jpg' //path to elephant testing images

   img1 = cv2.imread(filename, cv2.IMREAD_GRAYSCALE) //read and convert images into grayscale

   sift = cv2.xfeatures2d.SIFT_create() //initializing sift algorithm

    keypoints_sift, descriptors = sift.detectAndCompute(img1, None) //finding keypoints and descriptors for sift

      siftimg = cv2.drawKeypoints(img1, keypoints_sift, None) // draw keypoints on the image

   cv2.imshow("sift", siftimg) // SIFT feature extracted image(form of numpy array)
```

```
    pil_img = Image.fromarray(siftimg) //converting numpy array into .jpg image
    pil_img.save('/home/pi/Desktop/elephant/sift'+str(i)+'.jpg') //saving the .jpg image
    cv2.waitKey(0) //display the image until a key is pressed
    cv2.destroyAllWindows() //close all windows
for i in range(1,11):
    test_image = image.load_img('/home/pi/Desktop/elephant/sift'+str(i)+'.jpg', target_size = (64, 64)) //path to feature extracted images
    test_image = image.img_to_array(test_image) //converting image to array
    test_image = np.expand_dims(test_image, axis = 0)
    result = model1.predict(test_image) //to predict the image
    print(result)
    predicted.append(int(result[0][0])) // storing each image result for confusion matrix
    training_set.class_indices
    if result[0][0] ==0:
        prediction = 'elephant'
        count=count+1
    else:
        prediction = 'others'
        count1=count1+1
    print(prediction)
acc=(count*100)/10
print("TRUE POSITIVE=",count)
print("FALSE POSITIVE=",count1)
print("accuracy=",acc,"%")
for i in range(1,11):
    filename='/home/pi/elephant/testing/others/o'+str(i)+'.jpg' //path to other testing images
    img2 = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    sift = cv2.xfeatures2d.SIFT_create() //initializing sift algorithm
```

```
    keypoints_sift, descriptors = sift.detectAndCompute(img2, None) //finding keypoints and descriptors for images.

        siftoimg = cv2.drawKeypoints(img2, keypoints_sift, None) //Drawing keypoints

    cv2.imshow("sift", siftoimg)

    pil_img = Image.fromarray(siftoimg)

    pil_img.save('/home/pi/Desktop/elephant/sifto'+str(i)+'.jpg')

    cv2.waitKey(0) //display the image until a key is pressed

    cv2.destroyAllWindows() //close all windows

for i in range(1,11):

    test_image = image.load_img('/home/pi/Desktop/elephant/sifto'+str(i)+'.jpg', target_size = (64, 64)) //path to elephant testing images

    test_image = image.img_to_array(test_image) //converting image to array

    test_image = np.expand_dims(test_image, axis = 0)

    result = model1.predict(test_image) //to predict the image

    print(result)

    predicted.append(int(result[0][0])) //storing each image result for confusion matrix

    training_set.class_indices

       if result[0][0] ==0:

        prediction = 'elephant'

        count2=count2+1

    else:

        prediction = 'others'

        count3=count3+1

    print(prediction) //Predicting the image as elephant or not

print("FALSE NEGATIVE=",count2)//Printing the false negative count

print("TRUE NEGATIVE=",count3) //Printing the true negative count

print(actual)

print(predicted)

results = confusion_matrix(actual, predicted)  //calculating confusion matrix
```

```
print ('Confusion Matrix for SIFT :')
print(results)
print ('Accuracy Score :',accuracy_score(actual, predicted) )
print ('Report : ')
print (classification_report(actual, predicted) )
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
fpr,tpr,thresholds=roc_curve(actual,predicted)
print(fpr)
print(tpr)
print(thresholds)
logit_roc_auc1 = roc_auc_score(actual,predicted)
import matplotlib.pyplot as plt
plt.figure()
plt.plot(fpr, tpr, label='ROC CURVE FOR ORB (Sensitivity%0.3f)' % logit_roc_auc1)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.savefig('rocsift')
plt.show()
print(“SURF”)
c1=0
c2=0
c3=0
```

```
c4=0

predicted1=[]

for i in range(1,11):

   filename='/home/pi/elephant/testing/elephant/'+str(i)+'.jpg' //path to elephant testing images

   img3 = cv2.imread(filename, cv2.IMREAD_GRAYSCALE) //read and convert images into grayscale

   surf = cv2.xfeatures2d.SURF_create() //initializing SURF algorithm

   keypoints_surf, descriptors = surf.detectAndCompute(img3, None) //finding keypoints and descriptors for each images

   surfimg= cv2.drawKeypoints(img3, keypoints_surf, None) // drawing keypoints on each image

   cv2.imshow("surf", surfimg) //printing the SURF feature extracted image

   pil_img = Image.fromarray(surfimg) //convering numpy array to .jpg image

   pil_img.save('/home/pi/Desktop/elephant/surf'+str(i)+'.jpg') //saving the .jpg image

   cv2.waitKey(0) //display the image until a key is pressed

   cv2.destroyAllWindows() //close all windows

for i in range(1,11):

   test_image = image.load_img('/home/pi/Desktop/elephant/surf'+str(i)+'.jpg', target_size = (64, 64)) //path to feature extracted images

   test_image = image.img_to_array(test_image) //converting image to array

   test_image = np.expand_dims(test_image, axis = 0)

   result = model1.predict(test_image) //to predict the image

   print(result)

   predicted1.append(int(result[0][0])) //storing each image result for confusion matrix

   training_set.class_indices

      if result[0][0] ==0:

      prediction = 'elephant'

      c1=c1+1

   else:
```

```
        prediction = 'others'
        c2=c2+1
    print(prediction)
acc1=(c1*100)/10
print("elephant=",c1)
print("others=",c2)
print("accuracy=",acc1,"%")
for i in range(1,11):
    filename='/home/pi/elephant/testing/others/o'+str(i)+'.jpg' //path to other testing images
    img4 = cv2.imread(filename, cv2.IMREAD_GRAYSCALE) //grayscale image reading
    surf = cv2.xfeatures2d.SURF_create() //initializing surf algorithm
    keypoints_surf, descriptors = surf.detectAndCompute(img4, None) //finding keypoints and descriptors for images.
    surfoimg = cv2.drawKeypoints(img4, keypoints_surf, None) //Drawing keypoints
    cv2.imshow("surf", surfoimg)
    pil_img = Image.fromarray(surfoimg)
    pil_img.save('/home/pi/Desktop/elephant/surfo'+str(i)+'.jpg')
    cv2.waitKey(0) //display the image until a key is pressed
    cv2.destroyAllWindows() //close all windows
for i in range(1,11):
    test_image = image.load_img('/home/pi/Desktop/elephant/surfo'+str(i)+'.jpg', target_size = (64, 64)) //path to elephant testing images
    test_image = image.img_to_array(test_image) //converting image to array
    test_image = np.expand_dims(test_image, axis = 0)
    result = model1.predict(test_image) //to predict the image
    print(result)
    predicted1.append(int(result[0][0])) //storing each image result for confusion matrix
    training_set.class_indices
```

```
    if result[0][0] ==0:
        prediction = 'elephant'
        c3=c3+1
    else:
        prediction = 'others'
        c4=c4+1
    print(prediction) //Predicting the image as elephant or not
print(actual)
print(predicted1)
results1 = confusion_matrix(actual, predicted1)
print ('Confusion Matrix for SURF :') // calculating confusion matrix for SURF
print(results1)
print ('Accuracy Score:', accuracy_score (actual, predicted1))
print ('Report: ')
print (classification_report (actual, predicted1))
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
fpr,tpr,thresholds=roc_curve(actual,predicted)
print(fpr)
print(tpr)
print(thresholds)
logit_roc_auc1 = roc_auc_score(actual,predicted)
import matplotlib.pyplot as plt
plt.figure()
plt.plot(fpr, tpr, label='ROC CURVE FOR ORB (Sensitivity%0.3f)' % logit_roc_auc1)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.savefig('rocsift')
plt.show()
print("ORB")
ct=0
ct1=0
ct2=0
ct3=0
predicted2=[]
Tk().withdraw()
for i in range(1,11):
    filename='/home/pi/elephant/testing/elephant/'+str(i)+'.jpg'
    img5 = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
    orb = cv2.ORB_create(nfeatures=1500) //initializing ORB algoirthm
    keypoints_orb, descriptors = orb.detectAndCompute(img5, None) //finding keypoints and descriptors for each image
    orbimg = cv2.drawKeypoints(img5, keypoints_orb, None) //drawing keypoints on each images
    cv2.imshow("orb", orbimg)
    pil_img = Image.fromarray(orbimg)
    pil_img.save('/home/pi/Desktop/elephant/orb'+str(i)+'.jpg')
    cv2.waitKey(0) //display the image until a key is pressed
    cv2.destroyAllWindows() //close all windows
for i in range(1,11):
    test_image = image.load_img('/home/pi/Desktop/elephant/orb'+str(i)+'.jpg', target_size = (64, 64))
```

```
    test_image = image.img_to_array(test_image) //converting image to array
    test_image = np.expand_dims(test_image, axis = 0)
    result = model1.predict(test_image) //to predict the image
    print(result)
    predicted2.append(int(result[0][0]))
    training_set.class_indices
     if int(result[0][0]) ==0:
       prediction = 'elephant'
       ct=ct+1
    else:
       prediction = 'others'
       ct1=ct1+1
    print(prediction)
acc2=(count*100)/10
print("elephant=",ct)
print("others=",ct1)
print("accuracy=",acc2,"%")
for i in range(1,11):
    filename='/home/pi/elephant/testing/others/o'+str(i)+'.jpg' //path to elephant testing images
    img6 = cv2.imread(filename, cv2.IMREAD_GRAYSCALE) //read and convert images into grayscale
    orb = cv2.ORB_create(nfeatures=1500)
    keypoints_orb, descriptors = orb.detectAndCompute(img6, None) //finding keypoints and descriptors for orb
    orboimg = cv2.drawKeypoints(img6, keypoints_orb, None) // draw keypoints on the image
    cv2.imshow("orb", orboimg) // ORB feature extracted image(form of numpy array)
    pil_img = Image.fromarray(orboimg) //converting numpy array into .jpg image
    pil_img.save('/home/pi/Desktop/elephant/orbo'+str(i)+'.jpg')
    cv2.waitKey(0) //display the image until a key is pressed
```

```
    cv2.destroyAllWindows() //close all windows
for i in range(1,11):
    test_image = image.load_img('/home/pi/Desktop/elephant/orbo'+str(i)+'.jpg', target_size = (64, 64)) //path to feature extracted images
    test_image = image.img_to_array(test_image) //converting image to array
    test_image = np.expand_dims(test_image, axis = 0)
    result = model1.predict(test_image) //to predict the image
    print(result)
    predicted2.append(int(result[0][0])) //storing each image result for confusion matrix
    training_set.class_indices
       if int(result[0][0]) ==0:
        prediction = 'elephant'
        ct2=ct2+1
    else:
        prediction = 'others'
        ct3=ct3+1
    print(prediction)


print(actual)
print(predicted2)
results2= confusion_matrix(actual, predicted2)
print ('Confusion Matrix for ORB:') //calculating confusion matrix
print(results2)
print ('Accuracy Score :',accuracy_score(actual, predicted2) )
print ('Report : ')
print (classification_report(actual, predicted2) )
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
fpr,tpr,thresholds=roc_curve(actual,predicted)
```

```
print(fpr)
print(tpr)
print(thresholds)
logit_roc_auc1 = roc_auc_score(actual,predicted)
import matplotlib.pyplot as plt
plt.figure()
plt.plot(fpr, tpr, label='ROC CURVE FOR ORB (Sensitivity%0.3f)' % logit_roc_auc1)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.savefig('rocsift')
plt.show()
print(“HOG”)
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt
from Tkinter import Tk
from tkFileDialog import askopenfilename
import os,sys,time
class Hog_descriptor():
    def __init__(self, img, cell_size=16, bin_size=8):
        self.img = img
        self.img = np.sqrt(img / float(np.max(img)))
```

```
        self.img = self.img * 255
        self.cell_size = cell_size
        self.bin_size = bin_size //number of bins of histogram
        self.angle_unit = 360 / self.bin_size
        assert type(self.bin_size) == int, "bin_size should be integer,"
        assert type(self.cell_size) == int, "cell_size should be integer,"
        assert type(self.angle_unit) == int, "bin_size should be divisible by 360"
    def extract(self): //Orientation and magnitude binning
        height, width = self.img.shape
        gradient_magnitude, gradient_angle = self.global_gradient()
        gradient_magnitude = abs(gradient_magnitude)
        cell_gradient_vector = np.zeros((height / self.cell_size, width / self.cell_size,
self.bin_size))
        for i in range(cell_gradient_vector.shape[0]):
            for j in range(cell_gradient_vector.shape[1]):
                cell_magnitude = gradient_magnitude[i * self.cell_size:(i + 1) * self.cell_size,
                                 j * self.cell_size:(j + 1) * self.cell_size]
                cell_angle = gradient_angle[i * self.cell_size:(i + 1) * self.cell_size,
                             j * self.cell_size:(j + 1) * self.cell_size]
                cell_gradient_vector[i][j] = self.cell_gradient(cell_magnitude, cell_angle)
        hog_image = self.render_gradient(np.zeros([height, width]), cell_gradient_vector)
        hog_vector = []
          //Block Normalization.
        for i in range(cell_gradient_vector.shape[0] - 1):
            for j in range(cell_gradient_vector.shape[1] - 1):
                block_vector = []
                block_vector.extend(cell_gradient_vector[i][j])
                block_vector.extend(cell_gradient_vector[i][j + 1])
                block_vector.extend(cell_gradient_vector[i + 1][j])
```

```
            block_vector.extend(cell_gradient_vector[i + 1][j + 1])
            mag = lambda vector: math.sqrt(sum(i ** 2 for i in vector))
            magnitude = mag(block_vector)
            if magnitude != 0:
                normalize = lambda block_vector, magnitude: [element / magnitude for element
in block_vector]
                block_vector = normalize(block_vector, magnitude)
            hog_vector.append(block_vector)
        return hog_vector, hog_image
    def global_gradient(self): //histogram for whole image
        gradient_values_x = cv2.Sobel(self.img, cv2.CV_64F, 1, 0, ksize=5)
        gradient_values_y = cv2.Sobel(self.img, cv2.CV_64F, 0, 1, ksize=5)
        gradient_magnitude = cv2.addWeighted(gradient_values_x, 0.5, gradient_values_y,
0.5, 0)
        gradient_angle = cv2.phase(gradient_values_x, gradient_values_y,
angleInDegrees=True)
        return gradient_magnitude, gradient_angle
    def cell_gradient(self, cell_magnitude, cell_angle): //histogram for regions in image
        orientation_centers = [0] * self.bin_size
        for i in range(cell_magnitude.shape[0]):
            for j in range(cell_magnitude.shape[1]):
                gradient_strength = cell_magnitude[i][j]
                gradient_angle = cell_angle[i][j]
                min_angle, max_angle, mod = self.get_closest_bins(gradient_angle)
                orientation_centers[min_angle] += (gradient_strength * (1 - (mod /
self.angle_unit)))
                orientation_centers[max_angle] += (gradient_strength * (mod / self.angle_unit))
        return orientation_centers


    def get_closest_bins(self, gradient_angle):
```

```
        idx = int(gradient_angle / self.angle_unit)
        mod = gradient_angle % self.angle_unit
        if idx == self.bin_size:
            return idx - 1, (idx) % self.bin_size, mod
        return idx, (idx + 1) % self.bin_size, mod
    def render_gradient(self, image, cell_gradient):
        cell_width = self.cell_size / 2
        max_mag = np.array(cell_gradient).max()
        for x in range(cell_gradient.shape[0]):
            for y in range(cell_gradient.shape[1]):
                cell_grad = cell_gradient[x][y]
                cell_grad /= max_mag
                angle = 0
                angle_gap = self.angle_unit
                for magnitude in cell_grad:
                    angle_radian = math.radians(angle)
                    x1 = int(x * self.cell_size + magnitude * cell_width * math.cos(angle_radian))
                    y1 = int(y * self.cell_size + magnitude * cell_width * math.sin(angle_radian))
                    x2 = int(x * self.cell_size - magnitude * cell_width * math.cos(angle_radian))
                    y2 = int(y * self.cell_size - magnitude * cell_width * math.sin(angle_radian))
                    cv2.line(image, (y1, x1), (y2, x2), int(255 * math.sqrt(magnitude)))
                    angle += angle_gap
        return image
Tk().withdraw() //Withdraw the window
filename = askopenfilename()
img = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
hog = Hog_descriptor(img, cell_size=8, bin_size=8) //With region and bin size as 8 extract the vectors.
vector, image = hog.extract()
```

plt.imshow(image, cmap=plt.cm.gray) **//Plot the extracted vectors on the grayscale image**

plt.show() **//Display the image.**

cv2.destroyAllWindows() **//close all windows**

plt.close() **//Close the displayed image.**