Object Detection and Image Classification using Ornithopter

A report submitted in partial fulfilment of the requirements

for the degree of B.Tech. Mechanical Engineering

with specialization in Design and Manufacturing

by

M.Durga Sai Anil
(Roll No: 118ME0015)

Supervisor: Dr.J.Krishnaiah

Department of Mechanical

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING, KURNOOL

May 2022

Certificate

I, **M.Durga Sai Anil**, with Roll No: **118Me0015** hereby declare that the material presented in the Project Report titled **Object Detection and Image Classification using Ornithopter** represents original work carried out by me in the **Department of Mechanical Engineering** at the **Indian Institute of Information Technology, Design and Manufacturing, Kurnool** during the years **2021–2022**. With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student's Signature

In my capacity as supervisor of the above-mentioned work, I certify that the work presented in this Report is carried out under my supervision, and is worthy of consideration for the requirements of B.Tech. Project work.

Advisor's Name: Advisor's Signature

Abstract

It's no surprise that humanity's initial attempts at flight were in the form of ornithopters, which look like birds and are propelled by humans. Leonardo da Vinci, the famous artist and engineer, is frequently credited with being the first to design a plausible flying machine in 1490: a huge bat-shaped aircraft. The wings of this craft are powered by the pilot's arms and legs. Despite the fact that the aircraft It was a brilliant design that was never built, and we now know it would not have flown. Considering the current state of knowledge, this is a significant accomplishment. At the turn of the century, there was a lot of focus on Both the mode of thrust production, from flapping wings to propellers, has shifted. and the means of generating power, from the human body to internal combustion engines engine. The aerodynamic problem has been considerably exacerbated. Countless engineers and scientists have attempted to solve the flapping-wing flight problem. craftspeople, but only moderate success has been achieved till recently. The Professor James de Laurier's Subsonic Aerodynamics Laboratory at the University of Toronto With successes in remote-controlled flapping-wing flight, the University of Toronto has been a prolific contemporary addition to the body of knowledge about flapping-wing flight. ornithopters, flapping-wing tiny air vehicles, and even a full-scale human-piloted engine-powered ornithopter are all on the drawing board. Professor De Laurier and UTIAS were honoured by the FAI in 1991 with the "Diplôme d'Honneur" for flying the world's first engine-powered remotely-piloted ornithopter. In the years that followed, theoretical and experimental study accelerated, culminating in the successful flight of a full-scale piloted ornithopter on July 8, 2006. For the past 20 years, the University of Toronto has been at the forefront of ornithopter innovation thanks to an unique wing-twisting mechanism and considerable research in aero elastic tailoring.

Acknowledgements

In the accomplishment of completion of my project on Object detection and image classification using an ornithopter, it is my privilege to acknowledge with respect gratitude, the keen valuable and ever-available guidance rendered to us by Dr. J Krishnaiah sir without the wise counsel and able guidance, it would have been impossible to complete the project in this manner. And as well as I would like to convey my gratitude to Prof.DVLN.Somayajulu sir who is the Director of Indian Institute of Information Technology Design and Manufacturing Kurnool. Your valuable guidance and suggestions helped me in various phases of the completion of this project. I will always be thankful to you in this regard

M.Durga Sai Anil

118me0015

Contents

List of Figures

Chapter 1

Introduction

1.1 What is Ornithopter?

An ornithopter is a flapping-winged aircraft that flies. Designers try to mimic the flapping-wing flight of birds, bats, and insects with their creations. Despite their differences in appearance, machinery are frequently manufactured on the same scale as these flying creatures. Manned ornithopters have also been developed, with considerable success. There are two types of machines: those with engines and those propelled by the pilot's muscles.

Micro Aerial Vehicles (MAV) research is relatively new, having only arisen in the last few years. Miniaturization of electric components such as electric motors, as well as advancements in microelectronics, allowed for the construction of miniature planes and helicopters at inexpensive costs. It was also able to begin imitating insect and bird flight as a result of this advancement. For their flapping wing motion, they require a highly miniaturised actuation chain. The purpose of this study is to develop small aerial vehicles that can function independently of ground stations, performing tasks such as surveillance and measurement in situations that are difficult to access or even harmful to people.

Ornithopters' flapping wings and motion through the air are engineered to optimise lift while staying within weight, material strength, and mechanical complexity constraints. The effectiveness of a flexible wing material can be increased while keeping the driving mechanism simple. Aero elastic deformation causes the wing to move in a manner close to its ideal efficiency (in which pitching angles lag plunging displacements by approximately 90 degrees) in wing designs with the spar sufficiently forward of the airfoil

that the aerodynamic centre is aft of the elastic axis of the wing. Flapping wings generate drag and are inefficient when compared to propeller-driven aircraft. Some designs boost efficiency by putting more power into the downstroke than the upstroke.

Engineers and researchers have experimented with wings that require carbon fibre, plywood, fabric, and ribs with a rigid, strong trailing edge in order to achieve the desired flexibility and minimal weight. Because any mass aft of the empennage lowers wing performance, lightweight materials and empty gaps are chosen whenever practical. The material for the wing surface must be chosen carefully in order to decrease drag and preserve the appropriate shape. A smooth aerodynamic surface with a double-surface airfoil is more efficient in producing lift than a single-surface airfoil, according to De Laurier's studies.

This thesis covers my 8months of work on object detection and image classification using an ornithopter project.

Chapter 2

About the Project

2.1 Purpose of the project

Building an ornithopter is already existing but my project main goal is to build an intelligent ornithopter.It means I want to provide an eye vision to the ornithopter for detecting the objects and classify those objects. For the purpose of eye vision, I used transfer learning technique to create a model which helps the ornithopter to detect and classify the objects. This project is mostly used in the defence applications, like hostage situations where people get caught by the terrorists or any other kidnappers we can visualise and classify them i.e.,faces,the number of people their armed equipment etc.., through the ornithopter.since it is an intelligent ornithopter it not only detects the persons and objects it also gives their bio data because it is having the facial detection and recognition with anti-spoofing technique.

For to provide intelligence to my ornithopter totally i build 4 deep-learning models which are

- Object Detection and image classification
- Facial recognition
- Anti-spoofing Facial recognition
- Shape Detection

2.2 Object Detection and image classification

This model is used for to detect the objects and classify those objects. This model was build with help of convolution neural network and trained with thousands of different object images. And the model is built using Computer vision library called OpenCV.

2.2.1 Object Detection VS Image Classification

For to understand the difference between the object detection and image classification first we will see the below image.. From above image we can easily recognize that it as

a dog.Now we will Take a step back and consider how we arrived at this decision. You were shown an image and asked to classify it into the appropriate category (a dog, in this instance). In a nutshell, that is what Image Classification is about. There is only one object here, as you can see: a dog. We can simply anticipate the presence of a dog in a given image using an image classification model. But what if we have a single image with both a cat and a dog? Image Localization enters the picture at this point. It assists us in

determining the location of a single object in a given image. When there are several objects present, we use the Object Detection concept. Using OD, we can forecast the location as well as the class of each object. We need to know what the image is contains before we can

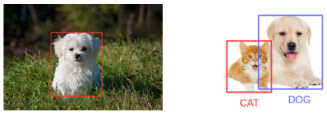


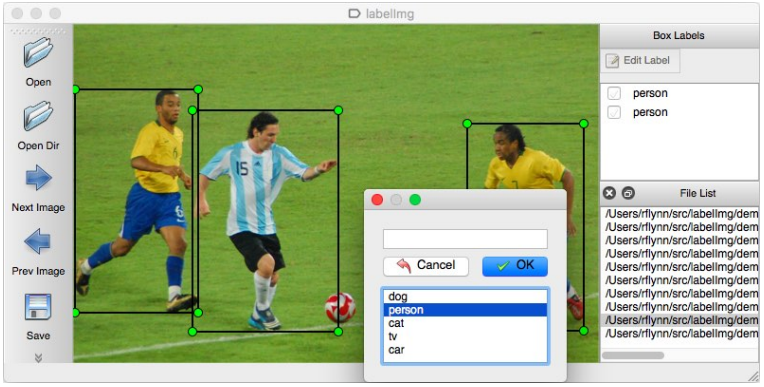
Image Localization Object Detection

detect the objects, and even before we can classify it. Image Segmentation comes in handy

here. The image can be divided or partitioned into multiple segments. It's not a good idea to process the entire image at once because there will be areas of the image that are devoid of information. We can use the key segments for image processing by separating the image into segments. That's how Image Segmentation works in a nutshell. We will only be able to generate a bounding box matching to each class in the image using Object Detection models. However, because the bounding boxes are either rectangular or square in shape, it will reveal nothing about the object's shape. On the other hand, image segmentation models will produce a pixel-by-pixel mask for each object in the image. This method allows us to get a far more detailed understanding of the object(s) in the photograph.

2.2.2 Object Detection & Image Classification using Deep Learning

For to creation of object detection and image classification model, i took dataset of normal pics like car and phone and i tried to create a model for detecting the car and phone so for that i used labelimg software for to train the model to detect the objects like car and phone. like as shown in the image below



similar from above figure i created a rectangle boxes for overall images of both classes like phone and car. so next with those images i tried to train in my local machine but due to high computation requires my laptop was struggled to train so then i research more about machine and deep learning and i got to know the concept called Transfer learning.

2.2.3 Transfer Learning

Transfer learning is a machine learning technique in which a model created for one job is utilised as the basis for a model on a different task. Given the vast compute and time resources required to develop neural network models on these problems, as well as the

huge jumps in skill that they provide on related problems, it is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks. I used a pre-trained model called YOLO to detect the objects.

Object identification is accomplished using a variety of methods, including rapid R-CNN, Retina-Net, and Single-Shot MultiBox Detector (SSD). Despite the fact that these approaches have overcome the constraints of data limitation and modelling in object detection, they are not capable of detecting objects in a single algorithm run. Because of its greater performance over the aforementioned object detection techniques, the YOLO algorithm has gained prominence.

2.2.3.1 What is YOLO?

The acronym YOLO refers for "You Only Look Once," and it's a real-time visual object identification and recognition system. Redmond et al proposed the concept in a paper presented at the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) in 2015. The paper was selected the OpenCV People's Choice Award winner. In contrast to the strategy used by object identification algorithms before YOLO, which repurposed classifiers to perform detection, YOLO proposes the use of an end-to-end neural network that offers predictions of bounding boxes and class probabilities all at once.

By taking a fundamentally novel approach to object recognition, YOLO achieves state-of-the-art results, beating previous real-time object detection algorithms by a significant margin.

YOLO offers the natural advantage of speed, in addition to higher prediction accuracy and a better Intersection over Union in bounding boxes (when compared to real-time object detectors).

YOLO is a lot quicker algorithm than its competitors, reaching speeds of up to 45 frames per second.

YOLO makes all of its predictions using a single fully connected layer, whereas methods like Faster RCNN use the Region Proposal Network to detect prospective regions of interest and then execute recognition on those regions independently.

As a result, methods that use Region Proposal Networks have to execute numerous iterations for the same image, whereas YOLO only has to do one.

Although YOLO appears to be the greatest approach to apply when solving an object detection problem, it has significant drawbacks.

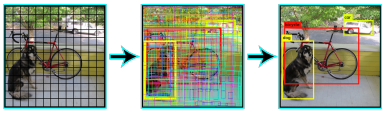
Because each grid can only detect one object, YOLO has difficulty detecting and segregating small things in images that appear in groups. Small things that naturally occur in swarms, such as a swarm of ants, are difficult to detect and locate with YOLO.

When compared to significantly slower object identification methods like Fast RCNN, YOLO is likewise characterised by lower accuracy.

2.2.3.2 How does YOLO Works?

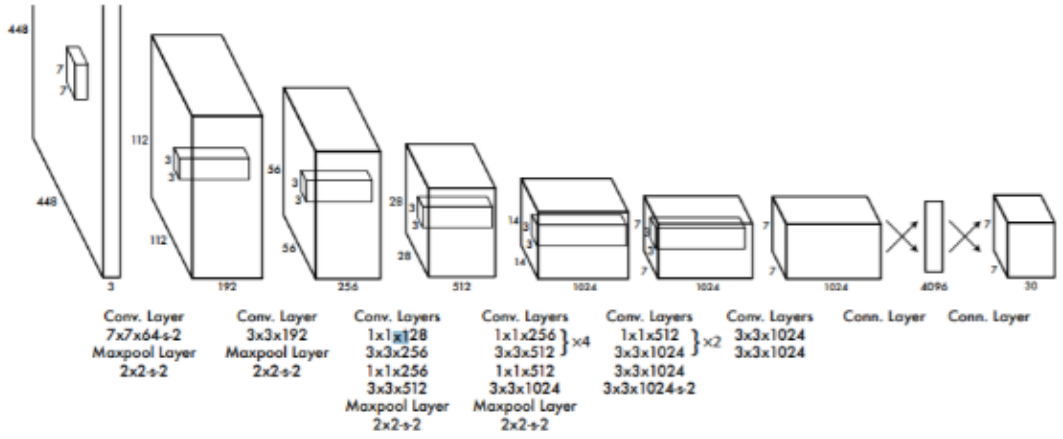
The YOLO method divides a picture into N grids, each with a SxS dimensional sector of equal size. Each of these N grids is responsible for detecting and localising the object contained within each of these N grids. In turn, these grids forecast B bounding box coordinates in relation to cell coordinates, as well as the item label and chance of the object being present in the cell.

Because cells from the image handle both detection and recognition, this strategy decreases computing time dramatically. However, because multiple cells predict the same object with different bounding box predictions, it generates a lot of duplicate predictions. YOLO uses Non Maximal Suppression to solve this problem.



In Non Maximal Suppression, YOLO suppresses all bounding boxes with lower probability values. YOLO does this by looking at the likelihood scores associated with each decision and choosing the one with the highest score. After then, the bounding boxes with the greatest Intersection over Union with the current high probability bounding box are suppressed. This procedure is repeated until all of the bounding boxes have been filled in.

The below picture is architecture of the Yolo algorithm. In this architecture we can see that it has multiple convolution and max-pool layers to extract the exact information in the image with less size so then after convolution of the images the extracted pixel image information will be flatten and that flatten information will be send to neural network with the help neural network architecture the model will be trained.



The people who trained yolo object detection model they created different versions based on their accuracy. so we can use different versions of yolo algorithm with our necessity of accuracy and capacity of our local system computational power.

So now let us see the different versions of Yolo Algorithms.

- YOLOv2

  YOLOv2 was created to fix two of YOLO's primary flaws: tiny object group detection and localization precision. YOLOv2 increases the network's mean Average Precision by introducing batch normalisation. Using Batch Norm, the mAP value can be enhanced by as much as 2Anchor boxes, as presented by YOLOv2, were a significantly more substantial contribution to the YOLO algorithm. YOLO forecasts only one object per grid cell, as we all know. While this simplifies the model, it poses issues when a single cell includes several objects because YOLO can only give a cell a single class. According to empirical research, the number 5 is a good trade-off between model complexity and prediction performance. DarkNet-19, which includes 19 convolutional layers and 5 max-pooling layers, serves as the YOLOv2 architecture's backbone.

- YOLO9000 Using a network design similar to YOLOv2, YOLO9000 was offered as a technique for discovering more classes than COCO as an object detection dataset could have made possible. The object identification dataset on which these models were trained (COCO) has just 80 classes, compared to classification networks like ImageNet, which have 22.000. YOLO9000 integrates the classification and detection duties into a single detection task by using labels from both ImageNet and COCO to enable the identification of many more classes. Because some COCO classes

are supersets of ImageNet classes, YOLO9000 uses a WordNet-inspired hierarchical classification-based technique in which classes and their subclasses are represented in a tree-based way.Despite having a lower mean Average Precision than YOLOv2, YOLO9000 can recognise over 9000 classes, making it a powerful algorithm.

- YOLOv3

  While YOLOv2 is a lightning-fast network, Single Shot Detectors and other solutions with higher accuracies have developed. In terms of accuracy, they exceed YOLOv2 and YOLO9000, although they are significantly slower. YOLOv3 was offered as a way to improve YOLO by including residual networks and skip connections into modern CNNs. While YOLOv2 uses the DarkNet-19 as the model architecture, YOLOv3 uses the DarkNet-53 as the model backbone, which is a 106-layer neural network with residual blocks and upsampling networks. YOLOv3's architectural uniqueness allows it to predict at three different sizes, with feature maps collected at layers 82, 94, and 106 for these predictions. By recognising traits on three separate scales, YOLOv3 compensates for the shortcomings of YOLOv2 and YOLO. This aids it in recognising tiny objects. Because of the design that allows the concatenation of the upsampled layer outputs with the features from preceding layers, the fine-grained features that have been obtained are maintained. This makes it easier to detect smaller items. YOLOv3 only predicts three bounding boxes per cell (vs five in YOLOv2), but at three different scales, for a total of nine anchor boxes.

- YOLOv4+

  Weighted Residual Connections, Cross Mini Batch Normalization, Cross Stage Partial Connections, Self Adversarial Training, and Mish Activation are among the contemporary methods of regularisation and data augmentation that YOLOv4 suggests as methodological breakthroughs. The authors also include a YOLOv4 Tiny version, which provides faster object detection and frame rate at the sacrifice of prediction accuracy.

Among all these versions i used YoloV3 due to its computational power it requires suitable for my local machine. Due to its accuracy of that version model it is performing very accurately. listings

2.2.4 Object Detection code

```
"""
Created on Tue Dec 14 01:07:49 2021
@author: Anil
"""
import cv2
import numpy as np
net = cv2.dnn.readNet('yolov3.weights','yolov3.cfg')
classes = []
with open('coco.names','r') as f:
    classes = f.read().splitlines()
#print(classes)
# laoding the image
img = cv2.imread('image3.jpg')
img = cv2.resize(img,None,fx=0.5,fy=0.5)
height, width, _ = img.shape
print(height,width)
# converting the image to give the input for yolo
blob = cv2.dnn.blobFromImage(img,1/255,(416,416),(0,0,0),
swapRB=True,crop=False)
#for b in blob:
# for n,img_blob in enumerate(b):
# cv2.imshow(str(n), img_blob)
net.setInput(blob)
output_layers_names = net.getUnconnectedOutLayersNames()
layerOutputs = net.forward(output_layers_names)
boxes = []
confidences = []
class_ids = []
for output in layerOutputs:
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:
            center_x = int(detection[0]*width)
            center_y = int(detection[1]*height)
            w = int(detection[2]*width)
            h = int(detection[3]*height)
            x = int(center_x - w/2)
            y = int(center_y - h/2)
            boxes.append([x,y,w,h])
            confidences.append(float(confidence))
            class_ids.append(class_id)
            #print(len(boxes))
indexes = cv2.dnn.NMSBoxes(boxes,confidences,0.5,0.4)
#print(indexes.flatten())
font = cv2.FONT_HERSHEY_PLAIN
colors = np.random.uniform(0,255,size=(len(boxes),3))
for i in indexes.flatten():
    x,y,w,h = boxes[i]
    label = str(classes[class_ids[i]])
```

```
    confidence = str(round(confidences[i],2))
    color = colors[i]
    cv2.rectangle(img,(x,y),(x+w,y+h),color,2)
    cv2.putText(img, label + " " +
confidence,(x+10,y+40),font,2,(255,255,255),2)
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input Image for the above code

FIGURE 2.1: Input Image for the above code

Output will be the detection of objects and classifying the objects in the image:

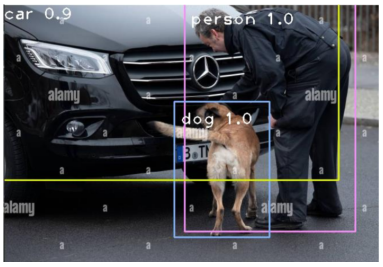


FIGURE 2.2: Output image from the above code

2.3 Facial Recognition

In the view of my ornithopters intelligence system facial recognition plays important role in it because while flying the ornithopter need to detect the human faces and recognize their faces so then in the situations like hostages or kidnapping by terrorists then defence officers need to know the persons inside the hostage building so then the facial recognition system plays important role in this application. so making note of this point i decided to build a facial recognition model.

2.3.1 What is facial recognition?

Facial recognition is a method of recognising or verifying a person's identification by looking at their face. People can be identified in pictures, films, or in real time using facial recognition technology. Biometric security includes facial recognition. Voice recognition, fingerprint recognition, and ocular retina or iris recognition are all examples of biometric software. Although the technology is mostly utilised for security and law enforcement, there is growing interest in other applications.

2.3.2 How does facial recognition works?

Face recognition technology is well-known to many people thanks to FaceID, which is used to unlock iPhones. Typically, facial recognition does not need a large database of images to identify an individual's identification; instead, it merely identifies and recognises one person as the device's only owner, while restricting access to others.

Facial Recognition Process

- **Step 1: Face detection** The camera recognises and tracks the image of a face, whether it is alone or in a crowd. The person in the image could be staring straight ahead or in profile.
- **Step 2: Face analysis** After that, a picture of the face is taken and examined. Because it is easier to match a 2D image with public photos or those in a database, most facial recognition technology uses 2D images rather than 3D images. Your face's geometry is read by the software. The distance between your eyes, the depth of your eye sockets, the distance between your forehead and chin, the curve of your cheekbones, and the contour of your lips, ears, and chin are all important considerations. The goal is to figure out what facial landmarks are important for differentiating your face.
- **Step 3: Converting the image to data** The face capture process transforms analog information (a face) into a set of digital information (data) based on the person's facial features. Your face's analysis is essentially turned into a mathematical formula. The numerical code is called a faceprint. In the same way that thumbprints are unique, each person has their own faceprint.
- **Step 4: Finding a match** After then, your faceprint is compared to a database of other people's faces. The FBI, for example, has access to up to 650 million images

culled from a variety of state databases. Any photo tagged with a person's name on Facebook becomes part of Facebook's database, which can be used for facial recognition as well. A determination is made if your faceprint matches an image in a facial recognition database.

Facial recognition is said to be the most natural of all biometric measurements. This makes intuitive sense, because we know ourselves and others by looking at their faces rather than their thumbprints and irises. Over half of the world's population is thought to be regularly exposed to facial recognition technologies.

Above said process is the General process of facial recognition. Taking the reference of the above process i created a facial recognition model using below dataset.

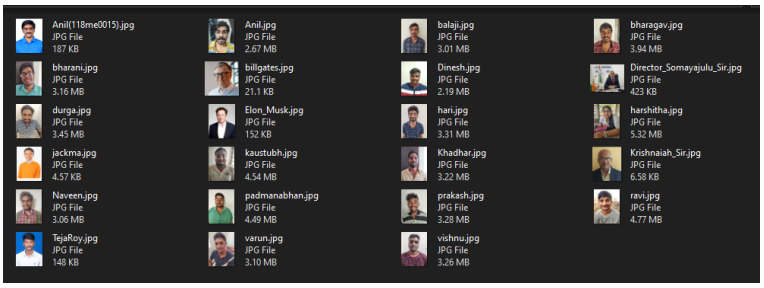


FIGURE 2.3: Face recognition Dataset

from above picture we can see that multiple faces images those are all my mechanical friends those images are taken manually and with that images i trained the model to detect and classify the faces.

my model was created using 68 facial landmarks of our face that was a pretrained model called harcascade_face_detector. It is a pretrained model using deep-learning convolution network. It is used to detect the faces from those faces we can extract the facial encodings of required faces. with those faces we can create live face recognition system.

I saved all encodings of those faces in a list and after that i write a code that which can take the live image of the person and it encodes that persons face and it compare with previous encodings and if the same encoding matches means it will display the matched face name or if not matches mean's it will display as unknown person.

2.3.3 Source Code For Facial Recognition

```
import pickle

import cv2
```

```
import numpy as np
import face_recognition

# code for automatically convert the image to bgr to rgb from particular file
import os

# creating a function for attendence sheet
# recording name and time
from datetime import datetime
def markattendence(name):
    with open('Attendence.csv','r+') as f:
        mydatalist = f.readlines()
        namelist = [] # for to store the list of values in the csv file. for to avoid storing the
        for line in mydatalist:
            entry = line.split(',')
            namelist.append(entry[0])
        if name not in namelist:
            now = datetime.now()
            dtstring = now.strftime('%H:%M:%S')
            f.writelines(f'\n{name},{dtstring}')

# loading the encodelistknown_facesqq
encodelistknown_faces = pickle.load(open("encodelistknown_faces","rb"))
classnames = pickle.load(open("classnames","rb"))
cap = cv2.VideoCapture(0)
cap.set(10,50)

while True:
    success,img = cap.read()
    # we need to reduce the real time image for to increase the speed
    imgs  = cv2.resize(img,(0,0),None,0.25,0.25) # it is resizing the 1/4th of orginal image
    # converting current image bgr to rgb
    imgs = cv2.cvtColor(imgs, cv2.COLOR_BGR2RGB)
    # finding the face locations in current image
    facescurrframe = face_recognition.face_locations(imgs)
    # encoding the faces in current frame
    encodinscurrframe = face_recognition.face_encodings(imgs,facescurrframe)

    for encodeface,faceloc in zip(encodinscurrframe,facescurrframe): # for to take both encoding a
        matches = face_recognition.compare_faces(encodelistknown_faces,encodeface)
        faceDis = face_recognition.face_distance(encodelistknown_faces,encodeface)
        # taking out the lowest distance which tells us the best match for the given image
        matchindex = np.argmin(faceDis)
        # now for match index we need to draw the bounding box and the name to it
        if matches[matchindex]:
            # printing the name of the current image if the faces are matches
            name = classnames[matchindex].upper()
            #print(name)
            # drawing a bounding box to matched face
            y1,x2,y2,x1 = faceloc
            y1, x2, y2, x1 = y1*4,x2*4,y2*4,x1*4 # this is because we reduce the size by 1/4th so
            cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)
```

```
            cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
            cv2.putText(img,name,(x1+6,y2-6),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2)
            markattendence(name)

    cv2.imshow('webcam',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Above code will load the encoded faces and it will take the input image from the webcam and it will encode if it finds any face in that image and it will matches with encode face and if it matches it displays the person name like the images below from figure-2.4

2.4 Antispoofing for Facial Recognition

If facial recognition system is available in the system means it definitely requires antispoofing because if our ornithopter was detecting the faces means it needs to identify that whether that face is real face or fake face.if it was detecting faces simply means it cannot be said as artificial intelligence. it can be said as simple python program so we need to add antispoofing system also generally antispoofing system will be used in security systems like face recognition attendance system, face unlock, etc.

In these applications antispoofing plays very important role because like for example if our phone is unlocking with normal photos means then attackers will unlock our phone with simple picture of ours. so at this situation our mobile needs to identify the whether the face is live face or spoof face.

so similarly there are multiple types of attacks occur by attackers they are

- Presentation attacks

  Presentation attacks means as we discussed earlier phone unlocks to the normal picture. And Presentation attacks are carried done at the sensor level (1), without requiring access to the system's guts. Purely biometric weaknesses are the target of presentation attacks. In these assaults, intruders employ some form of artefact, usually fake, or attempt to imitate the appearance of actual users in order to gain unauthorised access to the biometric system. Because "biometric qualities are not secrets," attackers are aware that a large quantity of biometric data revealing a person's face, eyes, voice, and behaviour is publicly available, and they utilise that knowledge to try to defeat face recognition systems.

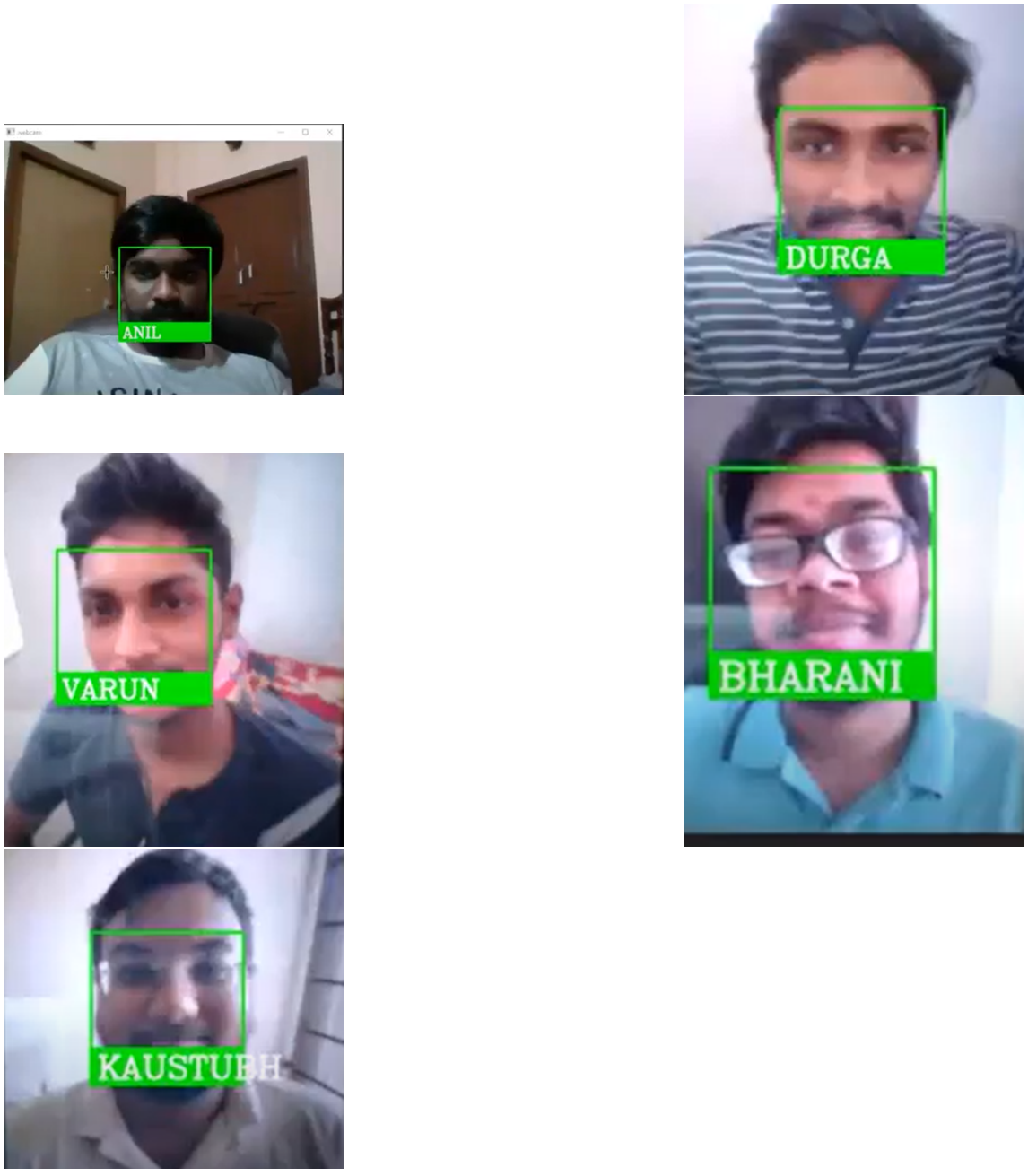


FIGURE 2.4: Faces detected with their names

- Indirect attacks

  Indirect assaults (2–7) can be carried out on the database, communication lines, and so on. This form of attack necessitates the attacker gaining access to the system's interior. Indirect attacks may be stopped using "traditional" cybersecurity tactics rather than biometrics, thus we won't go into them in this post.

Face recognition systems attempt to distinguish between genuine users rather than determining whether the biometric sample presented to the sensor is real or fake. We can do it in four different ways they are

- With the help of sensors
- With the help of dedicated hardware
- Challenge-response method it means we can instruct the user to specific thing like smile or lift your hand etc.
- Using Specific algorithms like
  - Specular feature projections
  - Depth feature fusion
  - Image quality assessment
  - Deep learning

Since we are implementing the facial recognition system in the ornithopter to detect the faces of the people whom we want to know like in defence applications and since it is not using for security purpose we can use simple antispoofing technique so i created a antispoofing model which will classify the faces between live image and fake image.

Process For building an Antispoofing model

- First i collected the images dataset which consists of two classes they are spoof images and real images.
- After i loaded the required libraries to create a CNN model.
- Loaded the dataset size of 30k images consists of two classes.
- Prepossessed the images to maintain all images size and resolution should be equal.

- Split the dataset into train and test with the ratio of 7:3 means 70% of the dataset goes to training the model and 30% of the dataset goes to testing the model
- Convolute our images in the convolution layer, In this layer it will reduce the size of an image and extract the important information from the images
- After the convolution layer the images will be flatten it means the images dimension will be 1dimension which can be used to give the input to neural network.
- Then the neural network trains based on the number of epoch we give.
- After model trained we will test our model with test dataset to know the accuracy, precision, recall, roc_auc_score,Confusion Matrix. These are called evaluation metrics. with the help of evaluation metrics we will decide whether our model is generalized model best fit model or not
- I Checked the the model whether it was classifying correctly or not.
- I Added that model to the facial recognition system

And also i used antispoofing technique by eye blinking sequence. Means the model will predicts the image whether it was fake or not based on eye blink sequence. with this it can easily distinguish the image is phone image are live image and it was the example image quality assessment.

2.4.1 Source Code For Anti-Spoofing

```
import cv2
import numpy as np
import face_recognition
from PIL import Image
from tensorflow.keras.preprocessing.image import img_to_array
import os
from tensorflow.keras.models import model_from_json
# Load Anti-Spoofing Model graph
json_file = open('Anti_spoofing_model/Face_Antispoofing_System/antispoofing_models/antispoofing_mo
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)
# load antispoofing model weights
model.load_weights('Anti_spoofing_model/Face_Antispoofing_System/antispoofing_models/antispoofing_
print("Model loaded from disk")
cap = cv2.VideoCapture(0)
cap.set(10,50)
while True:
```

```
success,img = cap.read()
# we need to reduce the real time image for to increase the speed
imgs  = cv2.resize(img,(0,0),None,0.25,0.25) # it is resizing the 1/4th of orginal image
# converting current image bgr to rgb
imgs = cv2.cvtColor(imgs, cv2.COLOR_BGR2RGB)
# finding the face locations in current image
facescurrframe = face_recognition.face_locations(imgs)
# encoding the faces in current frame
#encodinscurrframe = face_recognition.face_encodings(imgs,facescurrframe)
for faceloc in facescurrframe:
    y1, x2, y2, x1 = faceloc
    y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4  # this is because we reduce the size by 1
    resized_face = cv2.resize(img, (160, 160))
    resized_face = resized_face.astype("float") / 255.0
    resized_face = img_to_array(resized_face)
    resized_face = np.expand_dims(resized_face, axis=0)
    preds = model.predict(resized_face)[0]
    if preds > 0.5:
        label = 'spoof'
        cv2.rectangle(img, (x1, y1), (x2, y2), (0, 0, 255), 2)
        cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 0, 255), cv2.FILLED)
        cv2.putText(img, label, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255)
        #cv2.rectangle(img, (x, y), (x+w,y+h), (0, 0, 255), cv2.FILLED)
        #cv2.putText(img, label, (y1 + 6, x2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2
    else:
        label = 'real'
        cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
        cv2.putText(img, label, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255)
        # cv2.rectangle(img, (x, y), (x+w,y+h), (0, 0, 255), cv2.FILLED)
        #cv2.putText(img, label, (y1 + 6, x2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2
cv2.imshow('webcam',img)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

After adding this code to the Facial Recognition system the output look likes as shown in below figures-2.5,2.6,2.7.

If we will show the image of the person from mobile it won't recognize his face. but when we place real person infront of the camera based on the eye blink ratio the model will recognize it as live face and displays the name of the person.

2.5 Eye Gaze Detection

Actually it was a separate model but i created antispoofing model using CNN - Convolution Neural network to predict whether the person is closed the eyes are opened the eyes. So it

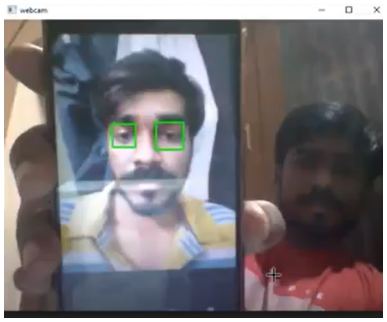


FIGURE 2.5: Model is not displaying the name of the person

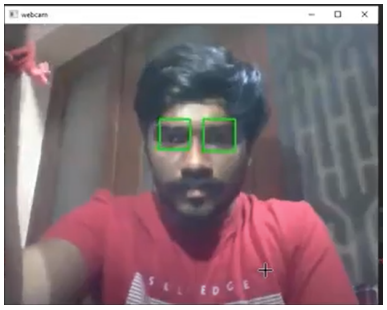


FIGURE 2.6: Model is calculating the eye blink ratio

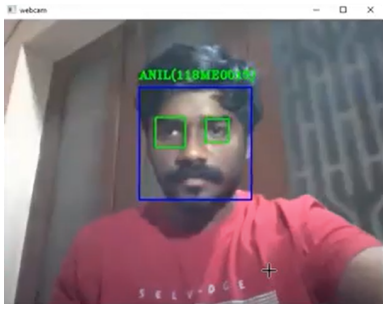


FIGURE 2.7: Model recognized the live face and displays the persons name

requires more computation power to run the code. so taking consideration of ornithopters weight speed of the intelligence i created model antispoofing by eye gaze ratio with the help of simple python programming and 68 facial landmarks which are available in computer vision libraries.

In this model it it will detect the eyes and eye gazes also so the sequence of the eyes should be like gazes should turn left and right and eye should blink in the sequence so for to detect those eye blinks and gaze blinks i wrote a computer vision code. so it will detect the eye and it will calculate the area of the eye so when ever the eye area is reduces or less than the threshold limit value then the model will say that eye was blinked and if the area is

more than that it will not say anything. so by this we can distinguish the eye blinking sequence.

And the model will checks in which side more "0's" are there the it will say that direction. Because our eye gaze is in black and remaining outer part of eye gaze is in black so in computer language 1 represents the White color and 0 represents the Black color so our model will half of our eye and it will see or count the number of 0 and number of 1. so if right side of eye is having more 0's means we can say that our gaze is looking right side similarly if more number of 0's in left side then we can say that our eye gaze is looking left side.

so with the help of these things we can know the whether eye is from real image or fake image.

2.5.1 Source Code For Eye Gaze Detection

```
import numpy as np
import cv2
import dlib
from math import hypot
cap = cv2.VideoCapture(0)
cap.set(10,80)
face_detector = dlib.get_frontal_face_detector()
landmarks_predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
def midpoint(p1,p2):
    return int((p1.x + p2.x)/2),int((p1.y + p2.y)/2)
def get_blink_ratio(eye_points,facial_landmarks):
    left_point = (facial_landmarks.part(eye_points[0]).x, facial_landmarks.part(eye_points[0]).y)
    right_point = (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y)
    # hor_line = cv2.line(frame, left_point, right_point, (0, 255, 0), 2)
    center_top = midpoint(facial_landmarks.part(eye_points[1]), facial_landmarks.part(eye_points[2
    center_bottom = midpoint(facial_landmarks.part(eye_points[5]), facial_landmarks.part(eye_point
    # ver_line = cv2.line(frame, center_top, center_bottom, (0, 255, 0), 2)
    hor_line_length = hypot((left_point[0] - right_point[0]), (left_point[1] - right_point[1]))
    ver_line_length = hypot((center_top[0] - center_bottom[0]), (center_top[1] - center_bottom[1])
    ratio = hor_line_length/ver_line_length
    return ratio
while True:
    success,frame = cap.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_detector(gray)
    for face in faces:
        x,y = face.left(),face.top()
        w,h = face.right(),face.bottom()
        cv2.rectangle(frame,(x,y),(w,h),(0,0,255),3)
        landmarks = landmarks_predictor(gray,face) # predicting 68 landmarks from the current fram
        # detecting blinking
```

```
left_eye_ratio = get_blink_ratio([36,37,38,39,40,41],landmarks)
right_eye_ratio = get_blink_ratio([42, 43, 44, 45, 46, 47], landmarks)
blinking_ratio = (left_eye_ratio+right_eye_ratio)/2
#print(blinking_ratio)
if blinking_ratio > 5:
    cv2.putText(frame,"Blinking",(50,150),cv2.FONT_HERSHEY_COMPLEX,7,(255,0,0))

# detecting gaze of an eye
left_eye_region = np.array([(landmarks.part(36).x,landmarks.part(36).y),
                            (landmarks.part(37).x,landmarks.part(37).y),
                            (landmarks.part(38).x,landmarks.part(38).y),
                            (landmarks.part(39).x,landmarks.part(39).y),
                            (landmarks.part(40).x,landmarks.part(40).y),
                            (landmarks.part(41).x,landmarks.part(41).y)],np.int32)
#cv2.polylines(frame,[left_eye_region],True,(0,0,255),2)

# creating mask for to select the left eye region accurately
height,width,_ = frame.shape
mask = np.zeros((height,width),np.uint8)
cv2.polylines(mask, [left_eye_region], True, 255, 2)
cv2.fillPoly(mask,[left_eye_region],255)
left_eye = cv2.bitwise_and(gray,gray,mask=mask)


min_x = np.min(left_eye_region[:,0])
max_x = np.max(left_eye_region[:, 0])
min_y = np.min(left_eye_region[:,1])
max_y = np.max(left_eye_region[:, 1])
gray_eye = left_eye[min_y:max_y,min_x:max_x]
#gray_eye = cv2.cvtColor(eye, cv2.COLOR_BGR2GRAY)
eye = cv2.resize(gray_eye,None,fx=5,fy=5)
cv2.imshow("Eye",eye) # gray_eye
# threshold for left eye
_,threshold_eye = cv2.threshold(gray_eye,70,255,cv2.THRESH_BINARY)
height,width = threshold_eye.shape
left_side_threshold = threshold_eye[0:height,0:int(width/2)]
left_side_white = cv2.countNonZero(left_side_threshold)

right_side_threshold = threshold_eye[0:height, int(width/2):width]
right_side_white = cv2.countNonZero(right_side_threshold)

gaze_ratio = left_side_white/right_side_white

# cv2.putText(frame,str(left_side_white),(20,100),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,255),3)
# cv2.putText(frame, str(right_side_white), (20, 150), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0,
cv2.putText(frame, str(gaze_ratio), (20, 150), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 3

threshold_eye = cv2.resize(threshold_eye, None, fx=5, fy=5)
cv2.imshow("Threshold_Eye", threshold_eye)
# in this we need to remove the skin part in eye
# so for that we create a mask. it means blank image and we place the
# mask on the frame and we extract the eye then we can find the eye regions with accuratel
```

```
            # cv2.imshow("Left_gray_eye",left_eye)
            # left_side_threshold = cv2.resize(left_side_threshold,None,fx=8,fy=8)
            # right_side_threshold = cv2.resize(right_side_threshold, None, fx=8, fy=8)
            cv2.imshow("Left",left_side_threshold)
            cv2.imshow("Right",right_side_threshold)
        cv2.imshow("Frame",frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
```

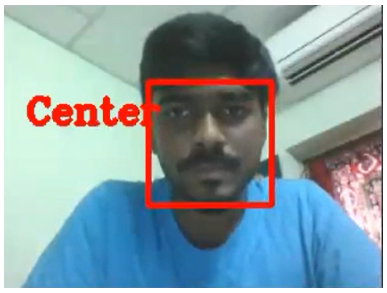


FIGURE 2.8: Model recognizing the gaze is looking center

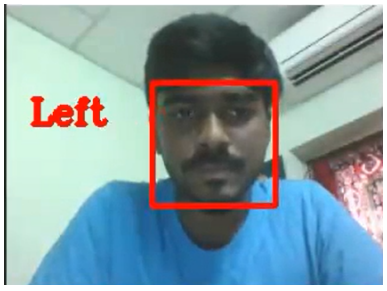


FIGURE 2.9: Model recognizing the gaze is looking Left

FIGURE 2.10: Model recognizing Eyes are blinking

2.6 Shape Detection

This type of detection is also having much necessary in while ornithopter is flying because if ornithopter is flying far away from the ground then the camera cant take the picture of complete image like we our human beings also won't recognize the object or person if the are far away from our eye sight but although we can identify or predict the person name or object name with the help of shape of a person or an object we can recognize them.

so similarly ornithopter also should recognize objects with the help of shapes when the ornithopter is flying far away from the ground.

So i created a model to detect the shape of an object with help of calculating its area of the object. like when an image given as input to the model then model identifies the patterns of shapes and with the help of area it describes its shape of an object.

2.6.1 Source Code For Shape Detection

```
import numpy as np
import cv2
def stackImages(scale,imgArray):
    rows = len(imgArray)
    cols = len(imgArray[0])
    rowsAvailable = isinstance(imgArray[0], list)
    width = imgArray[0][0].shape[1]
    height = imgArray[0][0].shape[0]
    if rowsAvailable:
        for x in range ( 0, rows):
            for y in range(0, cols):
                if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)
                else:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0][0].shape[1], imgArray
                if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor( imgArray[x][y], c
        imageBlank = np.zeros((height, width, 3), np.uint8)
        hor = [imageBlank]*rows
        hor_con = [imageBlank]*rows
        for x in range(0, rows):
            hor[x] = np.hstack(imgArray[x])
        ver = np.vstack(hor)
    else:
        for x in range(0, rows):
            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:
                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)
            else:
                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0])
            if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR_GRAY
```

```
        hor= np.hstack(imgArray)
        ver = hor
    return ver
def getContours(img):
    contours,hierarchy = cv2.findContours(img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
    for cnt in contours:
        area = cv2.contourArea(cnt)
        print(area)
        if area > 500: # Giving threshold used to avoid detecting the noise data
            cv2.drawContours(imgBlank, cnt, -1, (255, 0, 0), 3)
            peri = cv2.arcLength(cnt,True) # calculating the curve length helps to find the approx
            approx = cv2.approxPolyDP(cnt,0.02*peri,True) # finding the corner points
            # above True is because we are assuming all our shapes are closed
            print(len(approx))
            objcor = len(approx)
            # Drawing a bounding boxes around shapes
            x, y, w, h = cv2.boundingRect(approx)
            cv2.rectangle(imgBlank,(x,y),(x+w,y+h),(0,255,0),2)
            # from this bounding boxes we get information like width and height of the shape and c
            # of the shape
            # Categorise thier shapes
            if objcor == 3:
                objectType = "Tri"
            elif objcor == 4:
                # hear for square and rectangle will have 4 corners but square width and height ar
                # so we will divide width and height if the value is nearer to 1 then it is square
                aspRatio = w/float(h)
                if aspRatio > 0.95 and aspRatio < 1.05:
                    objectType = "Square"
                else:
                    objectType = "Rect"
            elif objcor > 4:
                objectType = "Circle"
            cv2.putText(imgBlank, objectType, (x + (w // 2) - 10, y + (h // 2) - 10), cv2.FONT_HER
                        (0, 255, 255), 2)



img = cv2.imread('Learn-OpenCV-in-3-hours/Resources/shapes.png')
# preprocessing the image for to find edges and to find corner points of a shape present in image
imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgBlur = cv2.GaussianBlur(imgGray,(7,7),1)
imgCanny = cv2.Canny(imgBlur,50,50)
imgBlank = np.zeros_like(img)
getContours(imgCanny)
imgBlank1 = np.zeros_like(img)

final = stackImages(0.5,([img,imgGray,imgBlur],
                         [imgCanny,imgBlank,imgBlank1]))
cv2.imshow("Original",final)
cv2.imwrite("Shape_Detection.jpg",final)
```

```
cv2.waitKey(0)
```

The process of detecting the shape of an objects in an image

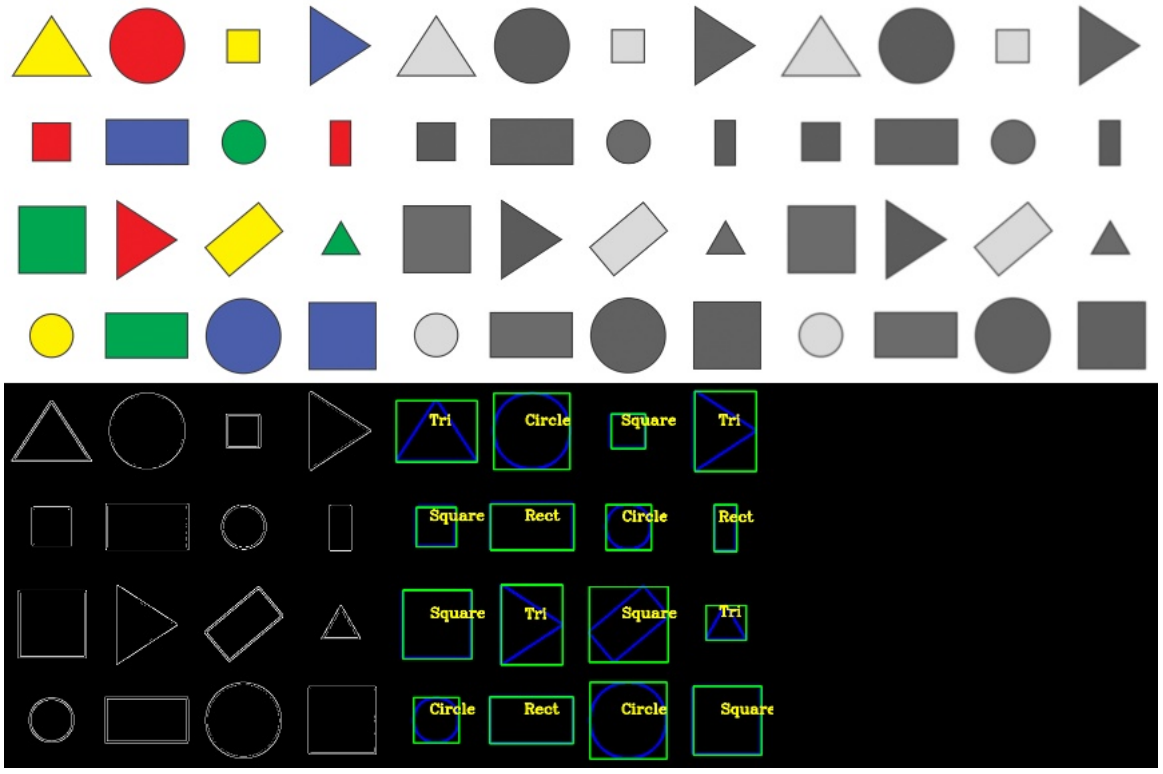


FIGURE 2.11: BGR image is converted to gray scale image and it detected the shape with 95% accuracy means out of 10 images 9 images detected correct.

Chapter 3

Libraries & Tools Used

3.1 Libraries used for to Build ML & DL models

From chapter 2 we can notice that there are almost 5 DL models are there for to provide intelligence to the ornithopter so now lets discuss about libraries that i used to build those models.

- Pandas
- Numpy
- MatplotLib
- OpenCV
- TensorFlow
- Keras

And in Background i used so many libraries but the main important libraries above 6. these 6 libraries are commonly used in above 4 detection models.

- Pandas

  Pandas are used for mainly data analysis. It allows importing the data from various file formats such as comma-separated values Json,SQL,MS excel. Pandas allows various data manipulations operations such as merging,reshaping selecting as well as data cleaning and data wrangling features.

- Numpy

  Numpy has bindings of C++ libraries. It is a python library used for working with arrays. it is also has functions for working with arrays. it is also has functions for working in domain of linear algebra, Fourier transform, Matrices. Numpy was created in 2005 by traits oliphant. it is an open source project and we can use it freely

- MatplotLib

  MatplotLib library is used to visualize the data output with the help of drawing line,bar,tree graphs etc. And with the help of this library i used to visualize the image whether the object is detected or not.

- OpenCV

  The most important library to create these many models is opencv. because it provides so much computer vision library. It really helps a lot to create a model based on real time detection's.

  OpenCV is an excellent tool for image processing and computer vision. It's an open-source library for tasks including face detection, objection tracking, landmark detection, and more. Python, Java, and C++ are among the languages supported.

  Hundreds of useful functions and algorithms are accessible in the library, all of which are free to use. Some of these functions are quite widespread, appearing in nearly every computer vision work. Many of the functions, on the other hand, are still unknown and haven't gotten much attention.

  OpenCV is a vital component of the computer vision community, and it allows us to create thousands of incredible applications. You may have assumed that we utilise some of these applications on a daily basis. When was the last time you scanned a copy of your homework with Cam Scanner? Do you think the second application had anything to do with it? Isn't our cartooned rendition of the image similar to a Snapchat version of the image? The goal of these apps was to urge you to use simple and efficient tools like OpenCV to investigate solutions to real-world situations.

- TensorFlow & Keras

  These libraries are very much used to create a deep learning model they act as back end to Create the Neural network architectures. from keras library it offers the all necessary functions to create neural network architecture like Sequential function to create sequential neural layers.

Chapter 4

Ornithopter Design

4.1 Ornithopter Gear Box Design

For to Build an ornithopter gear box design. i struggled a lot because for designing a gear box is not a easy task. so for to build gear box first i need to select the best gear mechanism design. generally they are so many gear mechanisms can be used for to flap the wings of an ornithopter so the mechanisms are

- Staggered Crank Mechanism

  This mechanism is the most basic gear mechanism to flap the ornithopter wings in this mechanism power transmission will be from shaft it means the shaft is will be in staggered shape and to that shaft there will be two cranks attached and connected to the wings so the power transmission will be from staggered shaft to the cranks and cranks to wings.

- Single Gear Crank

  In this gear mechanism As the mechanism flaps, the centre point where the connector rod and the wing hinges are attached must expand and shrink. Component failure could happen from contracting and expanding at a very high frequency.

- Dual Gear mechanism

  Each wing hinge is controlled separately by two gears in this configuration. Both secondary gears can be driven by the pinion wheel. As a result, the secondary gears would rotate in opposite directions. This design is much easier to install and reduces the misalignment of wing symmetry.

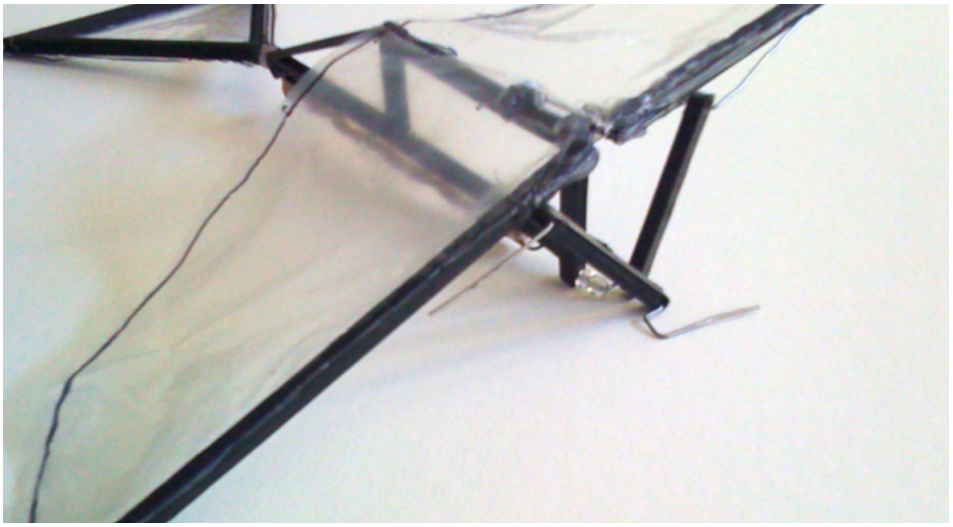


FIGURE 4.1: Staggered Crank Mechanism

FIGURE 4.2: Single gear Crank Mechanism

FIGURE 4.3: Dual gear Mechanism

- Transverse Shaft Mechanism

  The most symmetrical flap is possible with this design. It is, nevertheless, the heaviest and most complex design. Because the flapping wings and turning gears are not on the same plane, the connecting rod must be able to rotate. The connecting rod has a ball bearing, which adds weight to the component alone. This design has the highest number of gears of any other design. The transverse shaft is typically employed for large ornithopters with massive wings that can overcome their weight.

FIGURE 4.4: Transverse Shaft Mechanism

Among all these mechanism i choose Transverse shaft mechanism due to remaining all gears will have the opposition drag force to the mechanism in the air so transverse shaft mechanism will be good for my design of ornithopter so i selected this mechanism so i designed in solid works and i did torque analysis

But due to 3d printed gear i didn't get finished tooth gear so due to that i failed almost 3 design and finally i got a perfect design of gear box which i implemented on my ornithopter.

from figure 4.5 we can see that it is a transverse shaft gear mechanism which was implemented by me. without wings the mechanism was rotating perfectly but when i attached the wings the gear mechanism fails to rotate and the motor starts heated up and the mechanism failed...

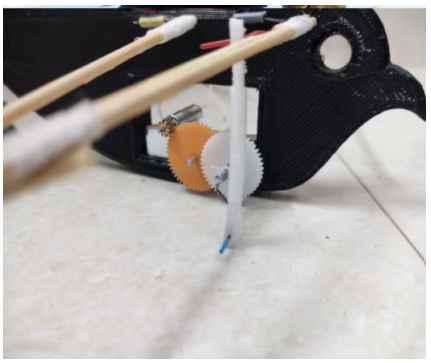


FIGURE 4.5: Transverse Shaft Gear mechanism design_1

so i re designed the gear mechanism with more number of toothed gear with high voltage motor but due to 3d printed gear the problem i arises is gears are not meshing properly. as shown in figure 4.6

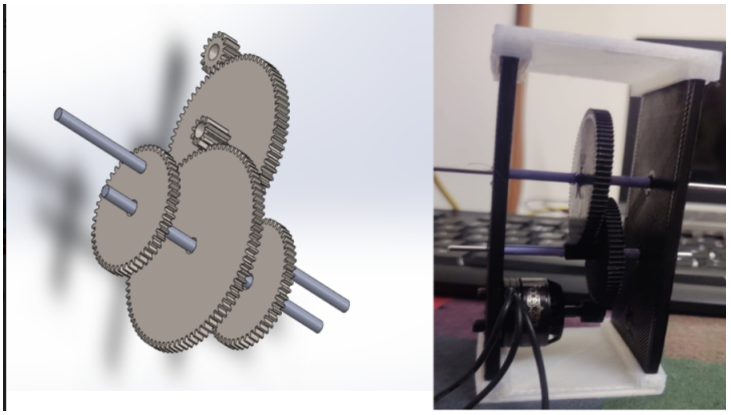


FIGURE 4.6: Transverse Shaft Gear mechanism design_2

Above gear design Specifications:

- pinion gear = 15 teeth
- driven gear = 72 teeth
- compound gear of driven gear = 10 teeth
- output gear = 82 teeth

Gear_Design_3 Specifications:

- Batter Specifications:

  - 11.1v,1500mah – 16.15W, 1.5Ah

- motor speed – 2500rpm – 41RPS
- Input Torque – 0.000555N/m
- Gear ratio – 39.36
- Output torque produces – 0.18648 N/m
- No of flaps can achieve – 13 Flaps Per second

Above gear design 2 is also failed to flap the wings so i redesigned the another gear mechanism by removing the middle 72 teeth gear and contacting the pinion gear and output 82 teeth gear. I designed this design from taking the reference of an Honda activa bike automatic gear transmission. in that bike when we drive slow then the bike produces high torque due to small gear drives the big size gear with help of belt.

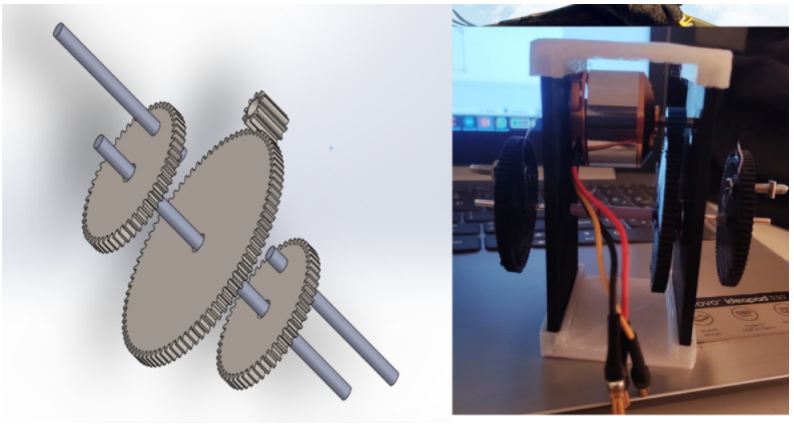


FIGURE 4.7: Transverse Shaft Gear mechanism design_3

from figure-4.7 This design is suited for my ornithopter and with the help of this gear mechanism i build my ornithopter

Gear_Design_3 Specifications:

- pinion gear = 15 teeth
- output gear = 82 teeth

- Batter Specifications:
  - 11.1v,1500mah – 16.15W, 1.5Ah

- motor speed – 2500rpm – 41RPS
- Input Torque – 0.000555N/m
- Gear ratio – 5.47
- Output torque produces – 0.00303 N/m
- No of flaps can achieve – 7 Flaps Per second

4.1.1 Prototype of Ornithopter

4.1.1.1 Ornithopter Body:

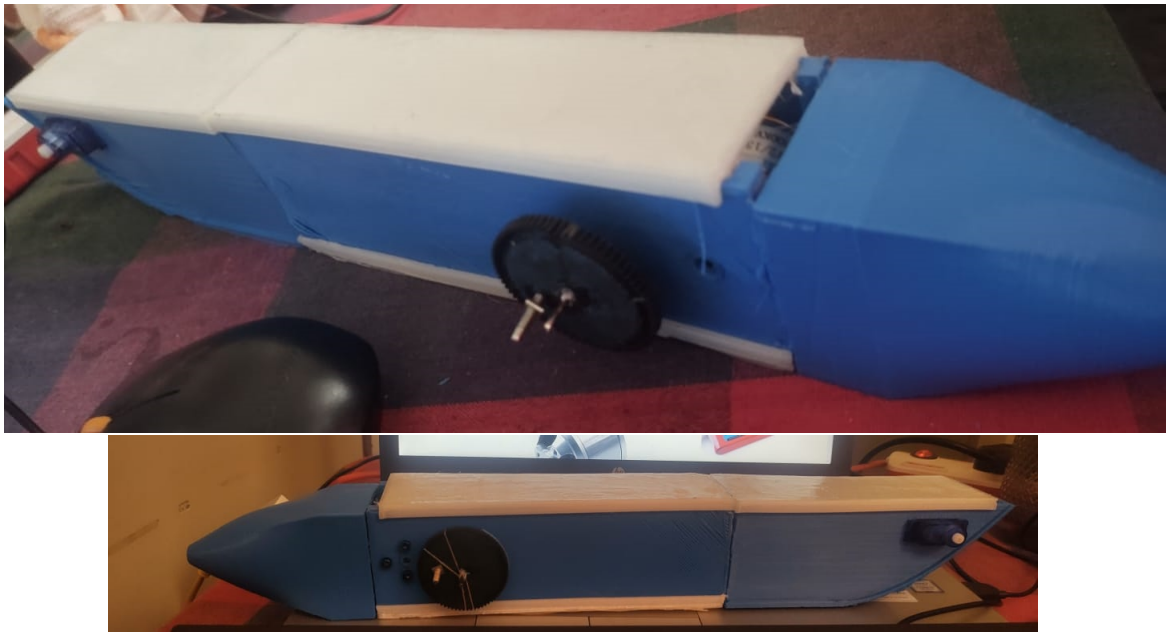


FIGURE 4.8: 3d Printed ornithopter body

From above figure 4.8 we can see that the body of my ornithopter which i designed in solidworks and i 3d printed it. it dimensions are

- Length of the body is 42cm
- height of the body is 5.3cm
- width of the body is 4.6cm
- weight of the body is approx 50grams

4.1.1.2 Wing & Tail Design

In ornithopter design process ornithopter wing design is very important process. so by taking all aspects of my overall intelligent ornithopter it needs to lift around 100grams of weight so the wingspan should be more so i designed my wings with wing span of 40cm. And the wings are necessary because ornithopter gain its lift with help of its flapping of wings so if the wing size is bigger with less number of flaps then the ornithopter can lift its body very easily.

And also tail design also very important design because ornithopter gains lift through wings but it turn its direction with help of ornithopter tail part by taking all aspect into consideration i designed the tail part of my ornithopter.

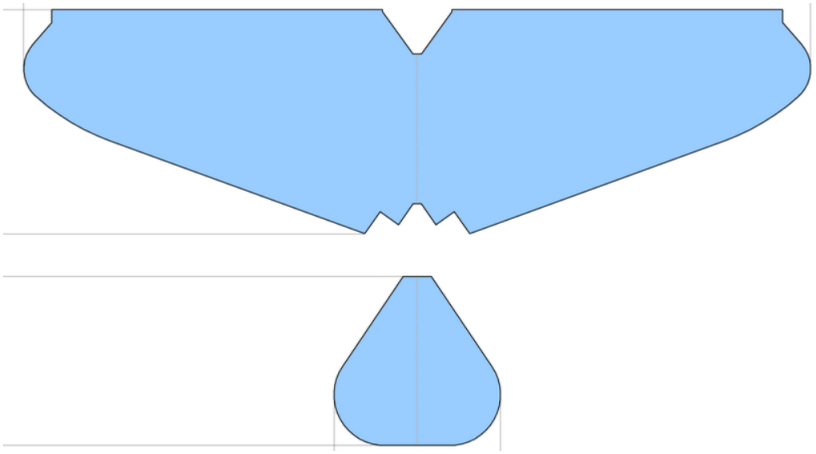


FIGURE 4.9: Wing & Tail Design

Chapter 5

Materials used to Build ornithpoter

5.1 Materials used..

- For to build ornithopter body i used 2 materials which are nylon and PLA material
- 1 bldc motor used for to flap the wings
- 2 servo motors used to turn the ornithopter body
- Polythene cover for to make wings
- cycle tyre spokes for to make wings
- FS-CT6B Transmitter and receiver to send the signal to ornithopter for to turning and flying

FIGURE 5.1: FS-CT6B Transmitter

Bibliography

[1] A. Senkiviskyy, "Anti-spoofing system for facial recognition," Ph.D. dissertation, Ukrainian Catholic university.

[2] Z. J. Jackowski, "Design and construction of an autonomous ornithopter," Ph.D. dissertation, MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

[3] "Ornithopter Instructables i build my ornithopter by taking the reference measurements from this website," https://www.instructables.com/Opensource-Ornithopter-Prototype-Arduino-Powered-a/.

[4] "YOLO i created object detection model by referring yolo website," https://pjreddie.com/darknet/yolo/.