Minor Project Report

On

"DOCUMENT CONTROL BY USING OBJECT DETECTION"

Submitted to

Amity University Uttar Pradesh

in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted By

Student Name: Mrigank Singh
Enrollment No.: A7605217004

under the guidance of
Dr. Sheenu Rizvi
Assisstant Professor
Department of Computer Science and Engineering

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY
AMITY UNIVERSITY UTTAR PRADESH
LUCKNOW (U.P.)
December 2020

AMITY UNIVERSITY

————————UTTAR PRADESH————————

DECLERATION BY THE STUDENT

I, Mrigank Singh, student of B.Tech hereby declare that the project titled "**Document Control by using Object Detection"** which is submitted by me to Department of Computer Science and Engineering, **Amity School of Engineering and Technology**, Amity University Uttar Pradesh, Lucknow, in partial fulfillment of requirement for the award of the degree of BACHELOR OF TECHNOLOGY in Computer Science and Engineering has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

The Author attests that permission has been obtained for the use of any copy righted material appearing in the Project report other than brief excerpts requiring only proper acknowledgement in scholarly writing and all such use is acknowledged.

Lucknow

Date

Mrigank Singh
B.Tech (CSE)
A7605217004

AMITY UNIVERSITY

————————UTTAR PRADESH————————

CERTIFICATE

On the basis of declaration submitted by Mrigank Singh student of B.Tech, I hereby certify that the project titled **"Document Control by using Object Detection"** which is submitted to **Amity School of Engineering and Technology**, Amity University Uttar Pradesh, Lucknow, in partial fulfillment of the requirement for the award of the degree of BACHELOR OF TECHNOLOGY in Computer Science and Engineering , is an original contribution with existing knowledge and faithful record of work carried out by her under my guidance and supervision.

To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Lucknow
Date :

Dr. Sheenu Rizvi
Assistant Professor
Amity University

External Examiner

Prof. (Dr.) Deepak Arora
Head – CSE & IT

Wg.Cdr (Dr.) Anil Kumar (Retd)
Asst. Pro. VC & Director ASET
Amity University, Lucknow

AMITY UNIVERSITY

————————UTTAR PRADESH————————

FACULTY GUIDE CERTIFICATE

I hereby certify that

- Mrigank Singh, A7605217004, student of B.Tech.(CS&E) and batch 2017-2021 at Amity School of Engineering and Technology, Amity University, Uttar Pradesh has completed the Project Report on "**Document Control by using Object Detection"** , during semester 7 under my supervision.
- The presented work embodies original research work carried out by the student as per guidance given in University Regulations.
- The research and writing embodied in the thesis are those of the candidate except where due reference is made in the text.
- I am satisfied that the above candidate's prima facie is worth examination both in terms of its content and its technical presentation related to the standards recognised by the university as approved for examination.
- I certify that in accordance with NTCC guidelines, the report does not exceed the prescribed maximum word limit or prior approval has been sought to go beyond the word limit.
- Wherever work from other source has been used, all depth (forward, data arguments and ideas) have been approximately acknowledged and referenced in accordance with the requirements of NTCC Regulations and Guidelines.

Date :

Dr. Sheenu Rizvi
Assistant Professor
Amity University, Lucknow

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task will be incomplete without the mention of the people whose ceaseless co-operation made it possible, whose constant guidance and encouragement crowns all the efforts with success.

I would like to express my most sincere and profound gratitude to **Dr. Deepak Arora, HOD, Dept. of Computer Science and Engineering, ASET, Amity University Lucknow**, for giving me inspiration and requisite facility by giving me a chance to show my capabilities and also for making me feel comfortable in the strictly professional environment of the college premises.

I am also thankful to **Dr. Sheenu Rizvi, Assistant Professor, Dept. of Computer Science and Engineering, ASET, Amity University Lucknow**, for his guidance throughout the work with his help and suggestions. I am also grateful to him for cooperation in providing me with all required resources.

I extend special thanks to my friends and family for their constant support.

This work would not have been completed without the help of the above mentioned people.

Mrigank Singh

Date :

INDEX

ABSTRACT

This project's aim is to help people control their Presentations and Portable Document Format (PDF) files through their hand gestures, instead of using a mouse or any other pointing device. This would mostly aid in accessibility and provide mobility to the presenter. The Project is mainly written in Python 3.8, and makes use of Python libraries like OpenCV and PyautoGUI to receive input from the computer's Webcam and recognize gestures to control the PowerPoint Presentation and Portable Document Format (PDF) files.

INTRODUCTION

There have been numerous disclosures in the field of Computer Science to improve individuals' life and be more helpful, and Artificial Intelligence is the most astounding one. Artificial Intelligence is getting increasingly more advanced as time passes. People are continually astounding themselves with the headway they have made around there; making machines think and work in an insightful way. Not exclusively is the space of Artificial Intelligence boundless, yet in addition there is positively no part of supportability where Artificial Intelligence can't be utilized. It has incredible breadth in giving the help people need and consequently it is being centered around by all technologists and experts all around the world.

One of the numerous utilizations of Artificial Intelligence is Object Detection, which would be the primary worry of my undertaking.

ARTIFICIAL INTELLIGENCE IN OBJECT DETECTION

Right off the bat, we ought to comprehend what Artificial Intelligence is. Artificial Intelligence is the reproduction of human intelligence in machines with the goal that they can think and copy the human mind. This term is related with any machine which can tackle issues or have attributes identified with human personalities. Usually, when a lot of people think about Artificial Intelligence, they consider Robots. This is because of the explanation that films and books everywhere in the world have consistently depicted robots to be human-like machines that are enormously cutting-edge and regularly make destruction on earth. In any case, this is a long way from truth or reality, and there is no ruin that we have to stress over.

Artificial Intelligence depends on the rule that human intelligence is characterized by a machine such that it can without much of a stretch execute errands like people; from the most basic positions to those that are unpredictable, in any event, for people themselves. The objectives of Artificial Intelligence incorporate learning, reasoning, and perception.

Artificial Intelligence is ceaselessly developing so various businesses are profited by it. Machines are wired utilizing a cross-disciplinary methodology situated in arithmetic, software engineering, semantics, brain research, and the sky is the limit from there. This cross-disciplinary methodology is the principle reason with respect to how Artificial Intelligence is being demonstrated useful in practically a wide range of businesses.

Some applications of Artificial Intelligence include-

- Healthcare industry for studying the dosage of drugs and different treatments for patients, as well as for surgical procedures in the operating room.
- Self Driving Vehicles
- Playing Chess and other games
- Artificial intelligence also has applications in the financial industry; it is used to detect and flag suspicious activity in banking and finance, such as unusual debit card usage and large account deposits — thus helping a bank identify fraudulent customers and defaulters.

The applications of artificial intelligence are endless. The technology can be used in almost every industry and help them in unimaginable ways.

CHARACTERIZATION OF ARTIFICIAL INTELLIGENCE

Artificial Intelligence can be characterized as *weak* and *strong* Artificial Intelligence.

1. **Weak Artificial Intelligence -** Weak Artificial intelligence is designed to do one particular job, instead of many complex jobs at once.The examples of weak Artificial intelligence are: Amazon's Alexa, Apple's Siri or a machine playing chess. In all these examples the Artificial Intelligence has only one job, either to win or answer a question.
2. **Strong Artificial Intelligence -**This kind of Artificial Intelligence is designed to behave even more like the human brain, hence their actions are more human-like.These are more complex and complicated systems.These systems are designed to handle situations in which they have to solve highly complex problems without the human's intervention.The examples of strong Artificial Intelligence are self driving cars and hospital operating rooms. These systems need to be developed almost completely accurately, leaving only minor or no shortcomings whatsoever.

Out of the many applications discussed above, one application of Artificial Intelligence is Object detection. We will discuss object detection in more detail as we proceed further in the report.

COMPUTER VISION

Computer Vision can be characterized as a field of Artificial Intelligence that discloses how to remake, intrude, and comprehend a 3-Dimensional scene from its 2-Dimensional pictures, regarding the properties of the structure present in the scene. It manages demonstrating and imitating human vision utilizing computer programming and equipment. A straightforward similarity of Computer Vision can be the natural eye; the way where people see objects and the general climate and the mind responds in a proper way. It resembles giving the machine eyes and encouraging its mind to work as per what it can see.

Computer Vision overlaps significantly with the following fields −

Image Processing − This focuses on image handling or image manipulation .

Pattern Recognition − This explains various techniques to identify and classify patterns.

Photogrammetry − This is concerned with obtaining accurate measurements and dimensions from images.

Now, since we have discussed a lot about Computer Vision, it seems mainly like image processing. However, it is important to understand the difference between the two.

Computer Vision	Image Processing
It is the construction of explicit, meaningful depictions of physical objects from their 2-dimensional image.	It deals with image-to-image transformation.
The output of computer vision is a portrayal or an interpretation of structures in 3-Dimensional scenes.	The input and output of image processing are both 2-Dimensional images.

These are some more applications of Computer Vision:

1. **Robotics Application**
    a. Localization − Determine robot location automatically
    b. Navigation
    c. Obstacles avoidance
    d. Assembly (peg-in-hole, welding, painting)
    e. Manipulation (e.g. PUMA robot manipulator)
    f. Human-Robot Interaction (HRI) − Intelligent robotics to interact with and serve people
2. **Medicine Application**
    a. Classification and detection (e.g. lesion or cells classification and tumor detection)
    b. 2-Dimensional/3-Dimensional segmentation
    c. 3-Dimensional human organ reconstruction (MRI or ultrasound)
    d. Vision-guided robotics surgery
3. **Industrial Automation Example**
    a. Industrial inspection (defect detection)
    b. Assembly
    c. Barcode and package label reading
    d. Object sorting
    e. Document understanding (e.g. OCR)
4. **Security Application**
    a. Biometrics (iris, fingerprint, face recognition)
    b. Surveillance − Detecting certain suspicious activities or behaviors
5. **Transportation Application**
    a. Autonomous vehicle
    b. Safety, e.g., driver vigilance monitoring

OBJECT DETECTION

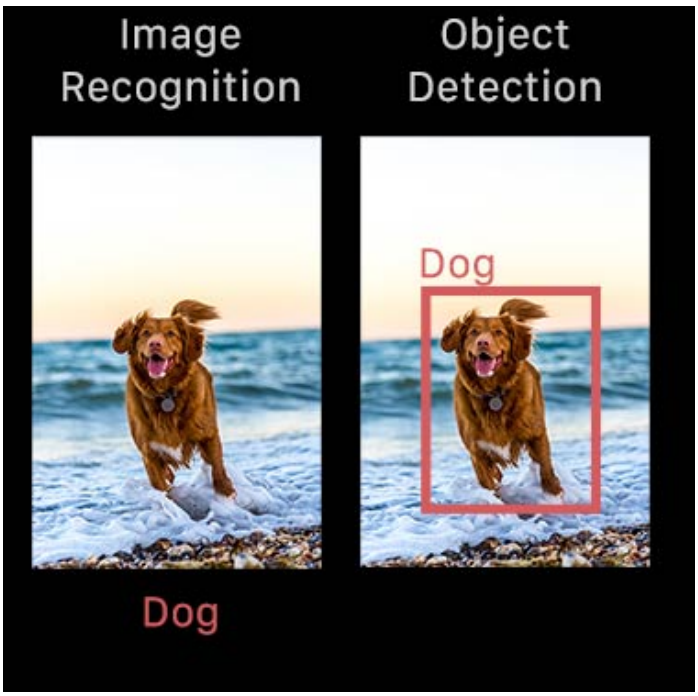
Object detection is a Computer vision procedure that helps in finding and distinguishing objects in a picture or a video. With the assistance of Object Detection, we can include the quantity of articles in a picture, recognize the sort of article appeared in the picture (for instance, a tree or a canine, and so on), or decide the exact area of any article with precise naming.

This can be handily clarified by the accompanying model: Imagine a picture contains two felines and an individual. Object Detection encourages us to recognize and separate the felines and the individual without any problem.

Object Detection is frequently mistaken for image recognition, thus first we have to comprehend the contrast between the two.

Image recognition doles out a name to a picture. An image of a canine gets the mark "canine". An image of two canines actually gets the mark "canine". Picture acknowledgment, as the term recommends, is simply "Recognising" the object in the picture.

Object detection, then again, draws a container around each canine and names the case "canine". The object detection model predicts where each object is, and what name ought to be applied to it. Along these lines, object detection gives more data about a picture and its substance, as opposed to simply acknowledgment.



MODES OF OBJECT DETECTION

Basically, there are two modes of object detection:

1. Machine Learning based object detection
2. Deep Learning based Object Detection.

In more conventional Machine Learning-based methodologies, Computer Vision procedures are utilized to take a gander at different highlights of a picture, for example, the shading histogram or edges to recognize gatherings of pixels that may have a place with an article. These highlights are then taken care of into a relapse model that predicts the area of the item alongside its mark.

Then again, profound learning-based methodologies utilize convolutional neural networks (CNNs) to perform start to finish, unsupervised object detection, in which features don't need to be characterized and extracted independently.

Why is Object detection important?

Object detection can be used in a number of ways, namely:

- Crowd counting
- Self-driving cars
- Video surveillance
- Face detection
- Anomaly detection

Some of the use cases and applications mentioned above are discussed below in detail-

Video surveillance

Since cutting edge object detection procedures can precisely distinguish and follow different cases of a given object in a scene, these strategies normally make themselves helpful to robotizing video observation frameworks.

For example, object detection models are equipped for following numerous objects on the double progressively, as they travel through a given scene or across video outlines. This sort of granular following could give priceless bits of knowledge into security,

laborer execution, and wellbeing, retail pedestrian activity, and so on from retail locations to mechanical production line floors.

Crowd counting

Crowd Counting is another favorable utilization of object detection. In thickly populated territories like city squares, amusement parks, and shopping centers, object detection can support organizations and regions to gauge various types of traffic all the more gainfully — regardless of whether through vehicles, by walking, or something else.

This capacity to restrict and follow individuals as they démarche through different spaces could successfully assist organizations with advancing anything from store hours, to move planning, coordinations pipelines to stock administration, and some more. Essentially, object detection could assist urban areas with arranging and devoting city assets, overseeing and sorting out occasions, and so forth.

Anomaly detection

Anomaly detection is a utilization instance of object detection that is best clarified through explicit models from enterprises.

In horticulture, for instance, an altered object detection model could definitely recognize and find expected occurrences of plant illnesses, permitting ranchers to distinguish perils to their harvest yields that would in some way or another not be perceivable to the exposed natural eye.

Considering medical care, object detection could be utilized to help treat conditions that have explicit and special suggestive sores. One such case of object detection in the medical care industry comes as skincare and the therapy of skin break out — an object detection model could find and recognize cases of skin inflammation in a flash, helping clinical experts in the determination and proper therapy.

What is especially significant and convincing about these potential use cases is the manner by which they use and give information that is commonly simply accessible to

rural specialists or specialists, individually, just as make their positions somewhat simpler.

Self-driving cars

Real-time car detection models are vital to the accomplishment of self-ruling vehicle frameworks. These frameworks should have the option to accurately distinguish, find, and track objects around them to travel through the world securely and productively. In any case, this utilization of object detection should be created with most extreme exactness, since broken ongoing frameworks can likewise be a wellspring of mishaps and harm.

While errands like picture division can be (and regularly are) applied to self-governing vehicles, object detection is a central assignment that sets up current work on making self-driving vehicles a reality.

OpenCV

OpenCV is the acronym for Open Source Computer Vision Library. It is a library of functions which is used mainly for live Computer Vision in real-time. OpenCV was developed by Intel. It is a cross-platform library, which means that it can run on different platforms easily. The library was originally written in C++ but it has bindings in many different languages such as Python, Java, Octave, Matlab, etc. OpenCV provides a lot of functionality to the users when it comes to real-time Computer Vision.

Some of the functionalities provided by OpenCV are listed as follows:

1. Read and Write Images.
2. Detection of faces and their features.
3. Detection of shapes, such as a circle, a rectangle, etc in an image. E.g Detection of a coin in an image.
4. Text recognition in images, for example: Reading Number Plates.
5. Modifying image quality and colors, for instance, Instagram, CamScanner, etc.
6. Developing Augmented reality apps.

And countless more.....

OpenCV makes the use of image processing in a very easy and effective manner. The support of different programming languages makes it very versatile and efficient.

Some of the languages supported are-

1. C++
2. Android SDK
3. Java
4. Python
5. C (Not recommended)

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

Why use OpenCV for computer vision?

OpenCV, or Open Source Computer Vision library, began as an examination venture at Intel. It is at present the biggest computer vision library as far as the sheer number of capacities it holds. It is for these commending insights of OpenCV, that is utilized generally everywhere in the world.

As referenced before, OpenCV contains executions of in excess of 2500 algorithms! It is uninhibitedly accessible for business just as scholastic purposes. Also, the rundown of numerous charms of OpenCV doesn't end here! The library has interfaces for various dialects, including Python, Java, and C++.

The primary OpenCV version, 1.0, was delivered in 2006 and the OpenCV people group has developed a far cry from that point forward.

Presently, let us direct our concentration toward the thought behind this conversation of OpenCV, the plenty of capacities OpenCV offers. We will be taking a gander at OpenCV with the point of view of Object Detection, and find out about a portion of the numerous capacities that make the undertaking of creating and understanding computer vision models simpler and charming too.

PyAutoGUI

PyAutoGUI is a Python package that lets us control our mouse and keyboard so as to automate their tasks with other applications. The package is compatible and runs on Windows, macOS, and Linux. It also runs on Python 2 and 3.

The command to install PyAutoGUI is:

pip install pyautogui.

The different features and automation tasks provided by this package are:-

- Moving the mouse pointer or typing with the keyboard for specified applications.
- Sending keystrokes to applications (such as filling out forms, etc.).
- Taking screenshots
- Locating an application window, resizing it, minimizing it, or closing it.

ABOUT THE PROJECT

The project that I have developed will let you control the Portable Document Format (PDF) files and Powerpoint Presentations, without actually touching your keyboard or mouse. The main advantage of using this is that it would be easier to move around while giving presentations or scrolling the document while reading any book or article on your computer screen. The technologies used in the project are OpenCV, PyAutoGUI, and NumPy.

The very first task that needed to be done was to import all the necessary libraries into our project file.

```
import cv2
import numpy as np
import pyautogui
```

The above code snippet shows the process of importing the necessary libraries for our project. By convention, libraries are imported at the very beginning of the code of any project.

The next step was to start capturing video from our webcam. For doing this, we would require OpenCV. OpenCV has functions ranging from the basics, that is, detecting a camera device, to highly complex processes, that is, object and motion detection.

The function to capture the video from the webcam is :

cap=cv2.VideoCapture(0)

The zero(0) in the parameter tells the system to use the system integrated webcam. If in case we wanted to use an external webcam, then instead of zero we would put ONE(1) inside the parameter of the *VideoCapture* function. This would be helpful if a system does not have a built-in or integrated webcam.

After we have successfully started capturing the video feed from our webcam, we need to distinguish the colors in the current frame, so as to identify the object from the surrounding objects and the background.

To do the above-mentioned task, we will have to specify a lower and upper range of values for a particular color in HSV format. In the case of my project, I have specified it to yellow, since yellow is not easily found in the background. This can however be changed to a different color, according to the natural surroundings of the user.

We can also set the font to suit our choice, for displaying any text on our frame.

```
font = cv2.FONT_HERSHEY_SIMPLEX
yellow_lower = np.array([20, 100, 100])
yellow_upper = np.array([30, 255, 255])
upper_limit=160
lower_limit=320
```

The above code snippet shows how to set the lower and upper range of colors in OpenCV and also how to set the font for display.

The *upper_limit* and the *lower_limit* variables are used further in the code. These values tell the frame from where it needs to create a border for tracking the movement of the object on the screen.

Then, I have created a while loop, that would run endlessly so that our video capture does not stop abruptly even when we do not want it to stop. Inside this while loop, we would be reading our capture from the webcam. Once we start reading the capture from the webcam, it will provide us with two pieces of information. The first is the retrieval and the second one is the frame. We are interested in the frame portion because that is what would be used further in the process.

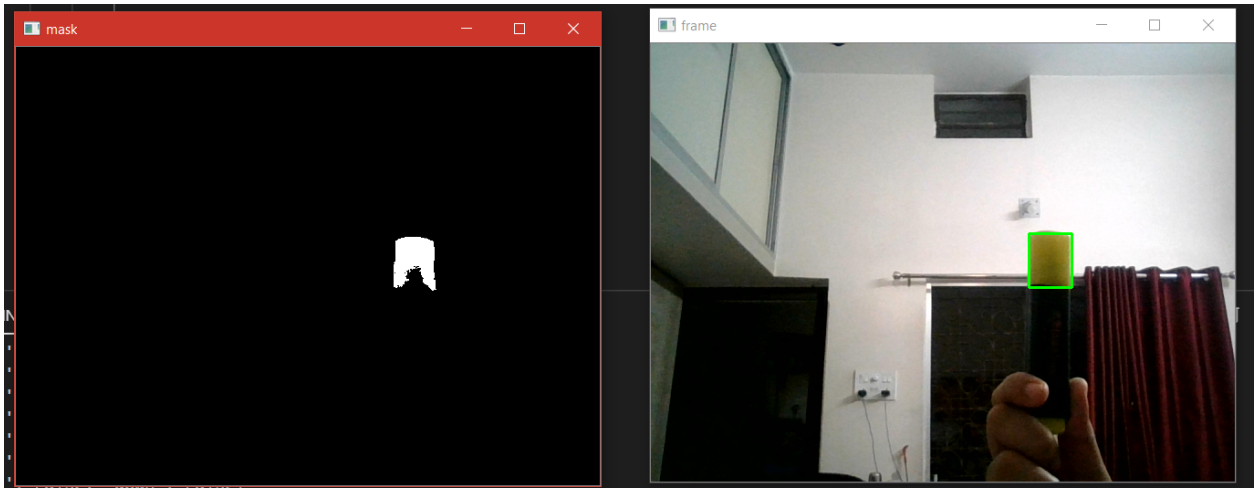
Now since we have started reading the frame, in the next step we would want to convert this particular read frame into HSV color format to get more precise results. After converting the color format, we need to mask it so that the computer only sees the color we want it to see. This will help in making the detection more accurate as the other colors will not be visible and our color would be very easily distinguishable.

```
while True:
    ret,frame=cap.read()
    hsv=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
    mask=cv2.inRange(hsv,yellow_lower,yellow_upper)
```

The above code snippet shows the process of reading the frame, changing the color to HSV format, and masking the frame.

The *hsv* variable would store the output of the *cvtColor* function, which is taking the frame and color conversion specification as input. Thus, the *hsv* variable would have our current frame in the HSV format.

The *mask* variable would store the result of masking the frame in the HSV format, using the *inRange* function which specifies the yellow color's upper and lower bounds.



The above image shows the result of the frame after masking. It can be noticed from the image above that every color except the neon yellow color is masked out or ignored. This helps us to distinguish the object of our choice from the image background to get accurate results.

Now, after identifying the object in our frame or video feed from the webcam, we need to get rid of the noise present in the image. We will be using contours from the OpenCV library to achieve the aforementioned result.

So what are contours? Contours are lines that join continuous points in an image, which have the same color or intensity. Contours help in identifying objects and shapes as well as image recognition.

Hence, to enable contours in my project I will be using the function *cv2.findContours*. This function provides us with two outputs which are 'hierarchy' and 'contours'. The arguments which would be passed into this function would be *mask* since it only works on the binary image, the border definition, in this case, is "*cv2.RETR_EXTERNAL"*. *cv2.RETR_EXTERNAL* would draw the contours as an external border of the object. The last argument to be passed is the contour approximation method. For my project, I will be using *cv2.CHAIN_APPROX_SIMPLE* as I do not want all the points across the object, rathe I only want the two endpoints connecting the contours, thereby giving us a straight line and reducing the memory usage.

After we have drawn the contours on our object, it is time to divide the screen into different regions so that we can distinguish different object movements that the user will make to control the application. For this, I have used the following function of OpenCV:

cv2.line()

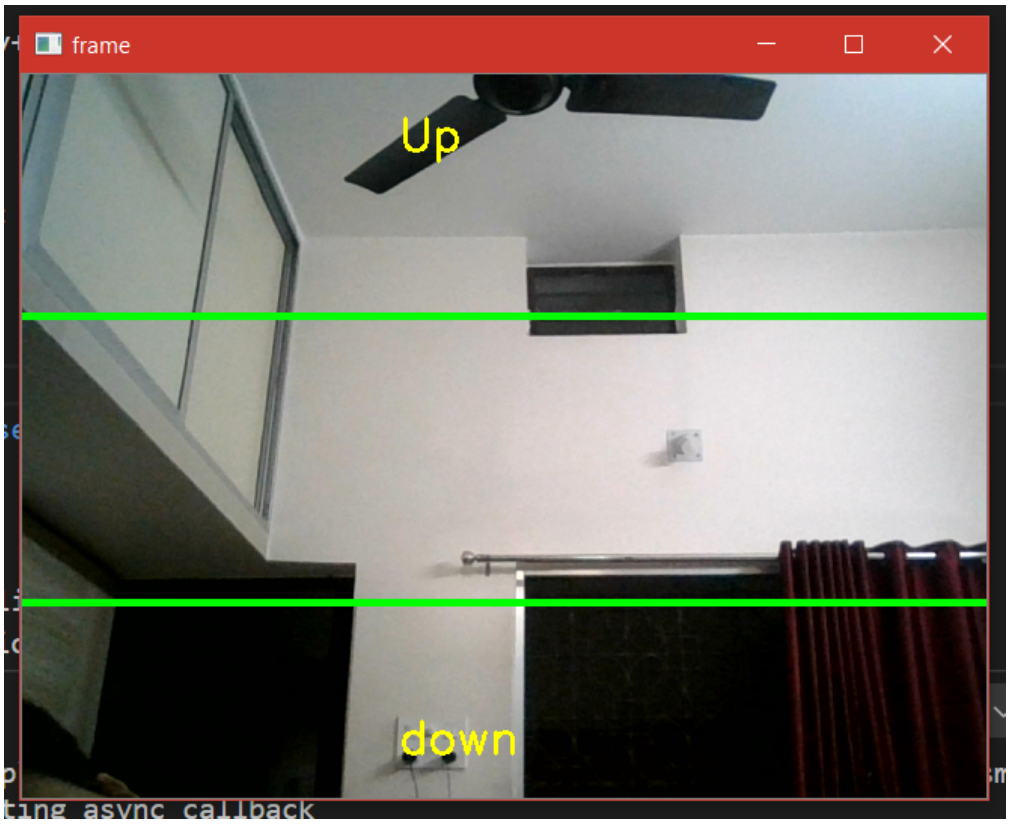
This function will draw a line of the specified dimensions and in a color of our choice, to demarcate the "region of interest" in the frame. The arguments which this function takes as input are: "frame", "x coordinate", "y coordinate", "color in RGB format" and finally the "thickness of the line".

After the line has been drawn on the frame we want to let the user know which region is used for which particular action. This will be done through the OpenCV function *cv2.putText()*.This function takes arguments "frame", "Text_to be_displayed", "region", "font".

```
contours,hierarchy=cv2.findContours(mask,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
frame = cv2.line(frame,(0,160),(640,160),(0,255,0),4)
frame = cv2.line(frame,(0,350),(640,350),(0,255,0),4)
cv2.putText(frame,'Up',(250, 50),font, 1,(0, 255, 255),2, cv2.LINE_4)
cv2.putText(frame,'down',(250, 450),font, 1,(0, 255, 255),2, cv2.LINE_4)
```

The above code snippet shows the process of drawing the contours, drawing lines, and displaying text in the frame, using the aforementioned OpenCV functions.

The output for the above code is shown below:



Now that we have detected our object and drawn contours to identify it, it is time to reduce the noise in the frame so that the computer can focus on one particular object in our webcam video feed.

To do the following task we will use the OpenCV function known as:

cv2.contourArea()

This function calculates the area of the different contours in our video frame. I would be passing all the contours which are being read by the video feed into a “for loop”. For this project, we need an area of more than 300px, this will be enough to do the work.

If the contour area is greater than 300px, only then the object would be identified, otherwise, it would be neglected by the system. After we have detected the object on the screen, it is time to draw a rectangle so that we can track it in our frame. I will draw

a rectangle around the object so that we can clearly distinguish it and track it on our screen.

Coming to the final step of the project, I had to implement the actions part. This will tell the system what action needs to be performed and when it needs to be performed, that is, the conditions in which the specific action needs to be performed. The coordinates of the rectangles drawn on the frame would come in handy for this particular task. We need its y-coordinate to know the region of the object detection in our frame, also known as the "region of interest". To do this, first, we need to define the actions that we want to perform using the object movements in the project.

The different actions which we need in our project up to this stage, is scrolling up and scrolling down. This action could be performed by the arrow up and arrow down keys of the keyboard. Thus, I will be initializing the actions with a dictionary and default value of "false" assigned to both the tasks.

```
actions={'up':False,'down':False}
```

Now, to perform the action, I will be using if-else conditions to distinguish between the actions.

```
if y<upper_limit:
    if actions.get('up')==False:
        pyautogui.press('up')
        actions['up']=True
        print(actions)
elif y>lower_limit:
    if actions.get('down')==False:
        pyautogui.press('down')
        actions['down']=True
        print(actions)
elif y>upper_limit and y<lower_limit:
    if actions.get('up') or actions.get('down'):
        actions['up']=False
        actions['down']=False
        print(actions)
```

The above code snippet shows the different actions being performed for different movements of the object in the frame. The code snippet specifically shows actions for the up key, down key, and no keypress by the keyboard.

If the object is in the upper region, the up arrow key will be pressed, similarly, if the object is in the lower region then the down arrow key will be pressed, and if the object is in the default region, that is, in the middle, then no actions will be performed and the actions would be reset.

Lastly, we want to close the application as per our convenience and we do not want it to run endlessly, so we can press "q' on the keyboard to close the frame window, and all the frames opened by the program.

```
        cv2.imshow('frame',frame)
        if cv2.waitKey(10)==ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
```

The *ord* function returns the ASCII value of the letter "*q*" and the computer checks for the keypress every 10 milliseconds. If the *"q"* key is pressed, it calls the *break* function and the loop is terminated. The function *cap.release()* tells the python interpreter that all the feed which is being read by the webcam needs to be released, and *cv2.destroyAllWindows()* function closes all the frames that were opened by the program.

CONCLUSION/ FUTURE SCOPE

The aim of this project was to demonstrate a very simple implementation of an Object Detection application. This project would be further extended to perform more complex functions and also to control documents such as PowerPoint Presentations and Portable Document Format (PDF) files. This project was just a small example to show the foundation of the more complex operations it is capable of performing. As another example, it can also be used to control actions in a video game and thus be used in motion sensing.

REFERENCES

1. https://www.fritz.ai/object-detection/#:~:text=Object%20detection%20is%20a%20computer,all%20while%20accurately%20labeling%20them.
2. https://www.circuitdigest.com
3. https://www.dx.doi.org
4. ACM Press SIGGRAPH Asia 2014 Courses- Shenzhen, China (2014.12.03-, by Angel, Ed Shreiner- 2014
5. IEEE 2015 ITU Kaleidoscope Trust in the Information Society
6. IEEE 2016 IEEE 21st International Conference on Emerging Technologies
7. IEEE 2018 International Conference on Advances in Computing and Com, by McInnis, Schillaci- 2018

The Report is Generated by DrillBit Plagiarism Detection Software

Submission Information

Author Name	Mrigank
Title	Minor Project
Submission/Paper ID	184141
Submission Date	19-Nov-2020 09:33:55
Total Pages	19
Total Words	4157

Result Information

Similarity	11 %
Unique	89 %
Internet Sources	5 %
Journal/Publication Sources	5 %
Total content under 'Quotes'	1 %

Exclude Information

References/Bibliography	Not Excluded
Quotes	Not Excluded
Sources: Less than 14 Words Similarity	Not Excluded

DrillBit
Anti Plagiarism Software

DrillBit Similarity Report

11	42	B
SIMILARITY %	MATCHED SOURCES	GRADE

A-Satisfactory (0-10%)
B-Upgrade (11-40%)
C-Poor (41-60%)
D-Unacceptable (61-100%)

Sl.No	LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1.	17	www.fritz.ai	1	Internet
2.	32	circuitdigest.com	1	Internet
3.	7	www.dx.doi.org	<1	Publication
4.	4	www.real-estate-key-largo.com	<1	Internet
5.	25	ACM Press SIGGRAPH Asia 2014 Courses- Shenzhen, China (2014.12.03-, by Angel, Ed Shreiner- 2014	<1	Publication
6.	34	Randomized Kinodynamic Planning by LaValle-2001	<1	Publication
7.	13	www.wrightslaw.com	<1	Internet
8.	28	IEEE 2015 24th IEEE International Symposium on Robot and Human Inter by	<1	Publication
9.	6	Neighborhood-based in-library use performance measures for public libr by Christi-2005	<1	Publication
10.	18	arxiv.org	<1	Publication