1. INTRODUCTION

1.1 CONTEXT

The target of this project is to create a program able to process video information from a webcam (i.e.the program must identify a moving object and track its position). Videos can be treated as a stack of pictures called frames. different frames(pictures) will be compared to the first frame which should be static(No movements initially). We compare two images by comparing the intensity value of each pixel (a pixel's intensity is its **brightness**). In python it can be done easily

1.2 SPECIFICATIONS

This project requires the installation of Python3 and the Pandas and OpenCV libraries.

For learning how to begin this assignment I read articles and watched videos on how to work in python and which predefined functions are useful and how they work.

1.3 Objectives

The main objective for the code to work is to create a way to discern an object from its background.

From what I have learned, I came up with the following small objectives the program has to fulfil when the code is ran:

- 4 new windows will be created
- The first window created is the Gray Frame : In Gray frame the image is a bit blur and in grayscale. (In grey pictures there is only one intensity value whereas in RGB(Red, Green and Blue) images there are three intensity values. So it would be easy to calculate the intensity difference in grayscale.)
- Difference Frame **:** Difference frame shows the difference of intensities of the first frame to the current frame.
- Threshold Frame : If the intensity difference for a particular pixel is more than 30(in my case) then that pixel will be white and if the difference is less than 30 that pixel will be black
- Colour Frame : In this frame, you can see the colour images in the colour frame along with green contour around the moving objects.

2. BIBLIOGRAPHIC STUDY

Installing Dependencies

We have already talked about libraries of Python, which are useful for us to work with our system's webcam. Here, in this part, we will discuss only those Python libraries or modules whose functions are required to design a Python program for using our system's webcam as a motion detector. In this project, we will use the functions of the following libraries to design the required program:

1. OpenCV
2. Pandas

Both OpenCV and Pandas are purely Python-based libraries, and these libraries are also open-source like Python. We should make sure that both of these libraries are present in our system before we proceed with this tutorial. We should also ensure that the latest version of Python or Python having version 3 is installed in our system.

- **Installing OpenCV Library:**

The OpenCV is a Python library or module which comes with multiple functions that we can use to work with pictures and videos. This OpenCV module was designed to work with the pictures and videos in Python by using functions of this library in Python programs. Therefore, the functions of this library become very important for us to write the required program of this tutorial. The OpenCV is not an in-built Python module, which means this module doesn't get installed along with the installation of Python. Therefore, we should first make sure that this module is already installed in our system or not, and if this module is not present in our system, we have first to install this module to work on this project. We can install the OpenCV module in our system through various methods but installing this module through the pip installer method is the simplest and easiest installation method for this module. Therefore, we will use the pip installer method to install this module, and to install this module using pip; we have to use the command prompt shell of our system. First, we have to open the command prompt shell of our system, and after that, **we have to write the following pip installation command to install the OpenCV module through pip installer:**

```
pip install OpenCV
```

The OpenCV module will install in our system in a while as it requires many dependencies which get installed along with this module. After successfully installing this module in our system, we can move to the installation process of the Pandas library.

- **Installation of Pandas Module:**

Pandas is an open-source module of Python designed to work on scientific and mathematical-related tasks. Pandas is one of the most popular and strongest libraries of Python, having lots of in-built functions that can be used for many research & developments related works. The pandas' library can work on our system to perform multiple tasks related to system software, computing operations, development projects, and many others. Therefore, we use the functions of this module along with the OpenCV module's functions to create the required program in this tutorial. The Pandas library functions will help in computing & analyzing the frame capturing through the system's webcam, thus, making it work as a motion detector. We will learn more about this while understanding the working of the required program in the latter part of this tutorial. Therefore, using Pandas library and its functions in this tutorial is very important and required for the proper functioning of the project. The Pandas library is also not an in-built module of Python, which means this module doesn't get installed along with the installation of Python. Therefore, we should first make sure that this module is already installed in our system or not, and if this module is not present in our system, we have first to install this module to work on this project. We can install the Pandas module in our system through various methods but installing this module through the pip installer method is the simplest and easiest installation method for this module too. Therefore, we will use the pip installer method to install this module, and to install this module using pip; we have to use the command prompt shell of our system. First, **we have to open the command prompt shell of our system, and after that, we have to write the following pip installation command to install the Pandas module through pip installer:**

```
pip install pandas
```

After writing this command, we have to press the enter key, and then the Installation process for this module will start. The Pandas module will install in our system in a while as it requires many dependencies which get installed along with this module. After successfully installing this module in our system, we can move to the next sections of this tutorial.

!!! As both of these libraries are already present in our system, it will show 'requirement is already satisfied' as shown in the installation window. If the module is not present already, it will install the module in our system, and after installing the module successfully, show the message 'Module installed successfully' on the command prompt shell's installation window.

Other Requirements:

Apart from the pandas & OpenCV library, we need to use many in-built modules of Python in the required program so that we can use the webcam of our system as a motion detector sensor. These in-built modules provide support to the motion that will be detected in the

frames through the webcam. We will learn more about this while learning about the program's work.

Following are the in-built modules of Python whose functions we will use in the required program of this tutorial:

- Python Datetime Module: This in-built module deals with the functions related to datetime tasks and, more specifically, related to date-related tasks.

- Python Time Module: This is another in-built module of Python that deals with the functions required to perform time-related tasks and time functions.

3. DESIGN

Understanding Main Logic

When we talk about a video, it means that we are talking about frames or a stack of pictures, and together these moving frames create a video. This program is basically designed by us to capture different frames (stack of pictures) and compare them with the initial static frame of the video where no movement is detected. That's how we can detect the difference in positioning and movements of objects between the initial static frame where no movement is detected and random capturing of frames through using different filters in the webcam of our system. This method is not only helpful for us to detect the change in the position of objects present in the initial frames but also helps us to track the movement through the proctoring process.. The two images or frames we compare after capturing them from the webcam of our system are actually compared based on the pixel intensity of the images, and this pixel intensity difference helps us find out the difference between the two images. For this, we will design a Python program, as we have discussed already in this tutorial, in which we will use functions of pandas, cv2, and many in-built modules so that we can compare the pixel intensity of images captured through the webcam of our system. After comparing the pixel intensity and finding out the intensity differences in images captured, we can detect the movement of objects in the frame. This is the main logic of the program we will write where we use the functions so that we can capture the differences and compare the pixel intensity differences using the program itself.

The Program

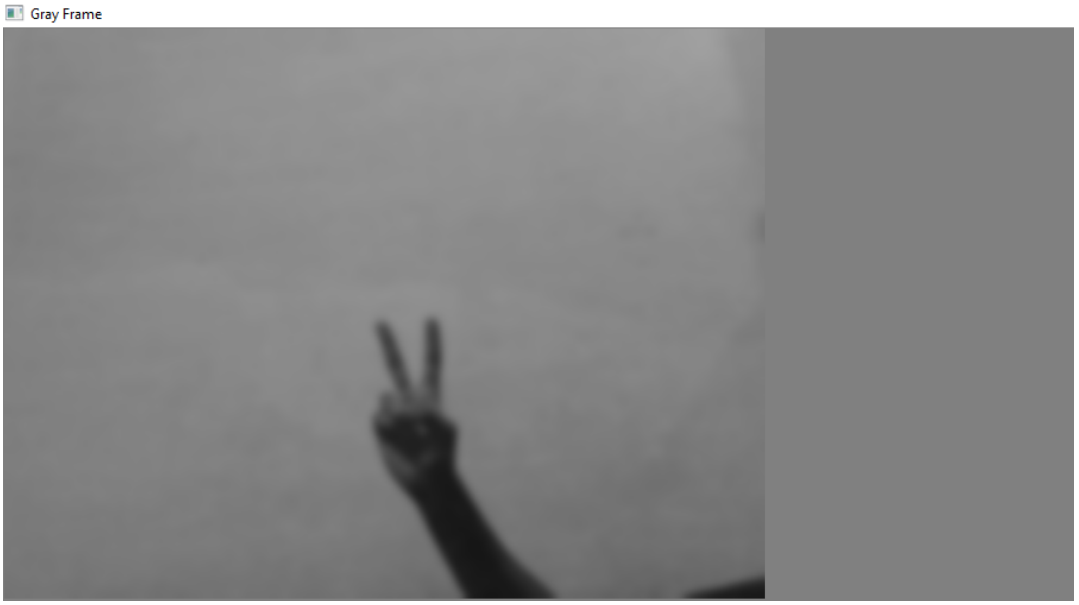
Output:

When we run the program, four different types of video frames will be opened on the output screen, which will capture the movements of an object present in the initial frame and added to the video frame later on. And apart from the frames, a CSV file will all the motion timing

details will be saved in the same directory where the code is present, but we will discuss this only after understanding all the four output video frames. These frames have their own specialty, and we will describe all of them by explaining the functioning of each of these frames. Each of these frames will be used to capture the movements and work like a motion detector using our system's webcam. These 4 video frame windows will capture different types of results, and each of them has different output, and therefore, we will study each of them separately as the output of the program. **Following is the description of the different color frames which will be displayed to us in the output window when we execute the program:**

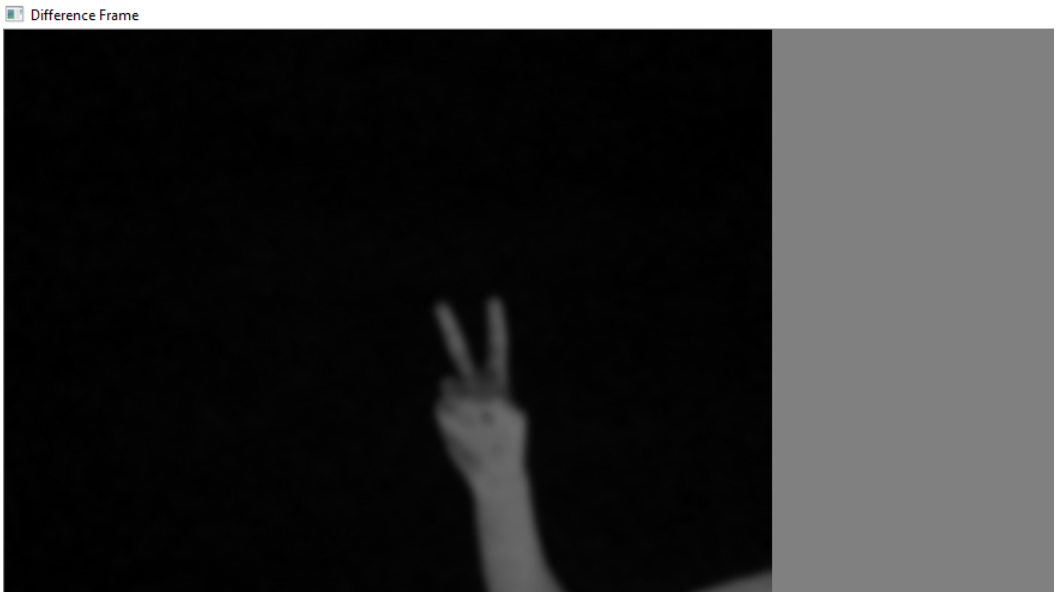
(1) Gray Frame: The first video frame that will open in the output screen is the gray frame. The image displayed inside the gray frame is displayed in the grayscale or gray shade (That's why this frame is named the gray frame), and it is also a bit blur. As we know, RGB (Red, Green & Blue) image frames have three intensity values. But, there is only one intensity value in the gray picture frame, as displayed here. This makes it even easier to capture the intensity value differences for the object present in the gray frame.

A resultant output window for the gray picture frame is displayed as the output of the program:



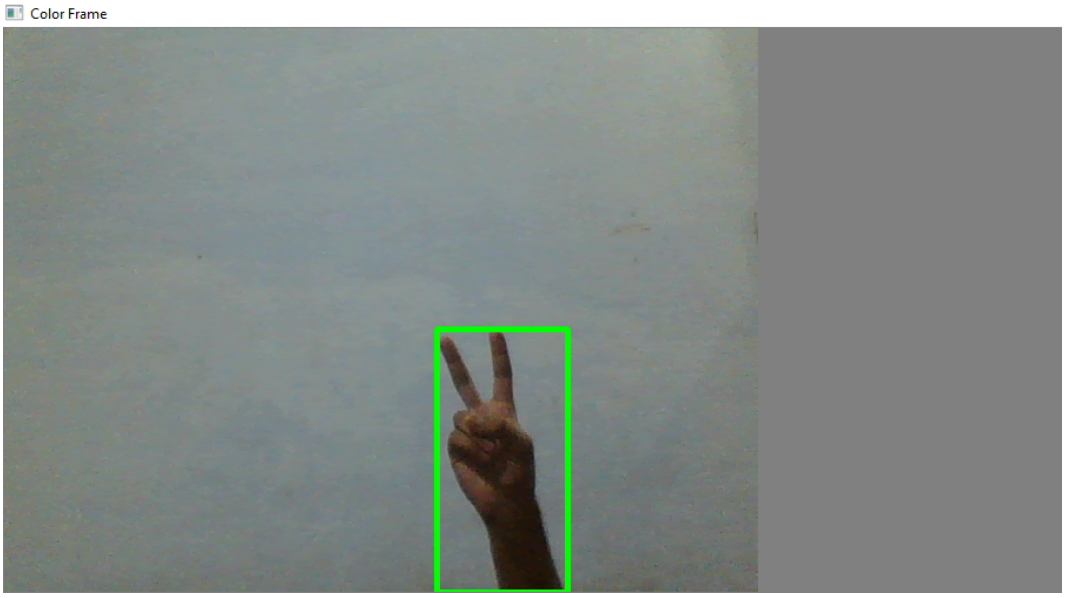
(2) Difference Frame: Instead of naming this frame as a black & white frame, we have named this frame as a difference frame because we will capture the difference of intensity values in the initial static frame & current frame through this video frame. Through this frame, we can detect the motion in the frame as well as movements of objects in the frame using the difference of intensity values in the initial static frame and current frame.

Following is the resultant difference frame displayed to us in the output screen:



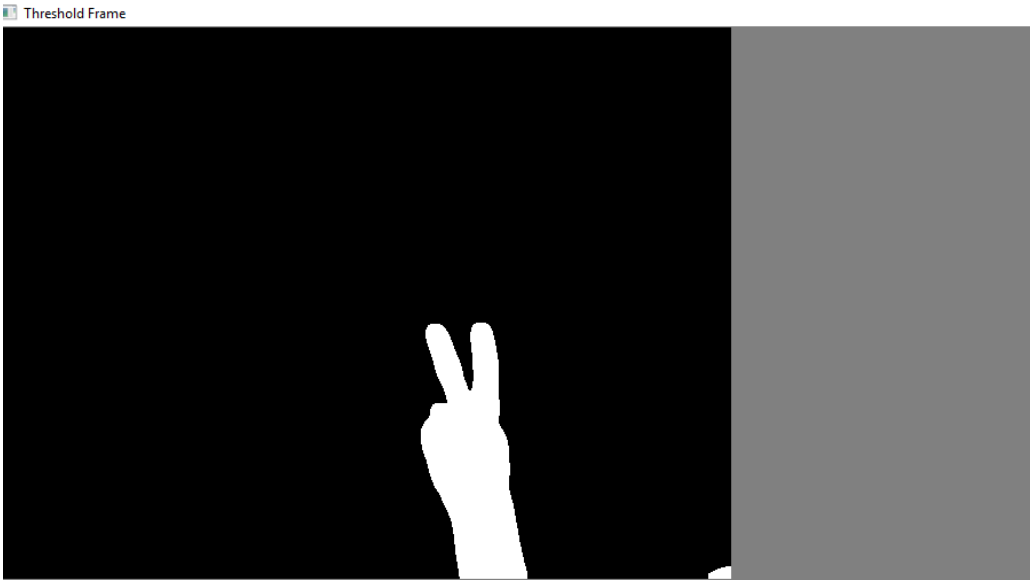
(3) Colour frame: This is the colour frame where images are displayed in the colour shade format (as we usually see in a camera or system's webcam). We capture the colour images in the color frame, and the movements of the objects present in the colour frame are highlighted with the green rectangle around the area where movement is happening in the frame.

A resultant output window for the color picture frame is displayed as the output of the program:



(4) Threshold Frame: This is the GaussianBlur picture filter frame, the filter which we applied in the program after the gray picture filter. As we defined in the program, when the intensity of the pixel changes with a difference of more than 30, then the pixel will be highlighted with the white color. In the opposite case, when the pixel's intensity changes with a difference of less than 30, then the pixel difference will be highlighted with the black color.

Following is the resultant Threshold frame displayed to us in the output screen:



4. IMPLEMENTATION

```
# Python program to implement
# Webcam Motion Detector

# importing OpenCV, time and Pandas library


import cv2, time, pandas

# importing datetime class from datetime library

from datetime import datetime

# Assigning our static_back to None

static_back = None

# List when any moving object appear

motion_list = [ None, None ]

# Time of movement

time = []
```

```
# Initializing DataFrame, one column is start
# time and other column is end time


df = pandas.DataFrame(columns = ["Start", "End"])

# Capturing video

video = cv2.VideoCapture(0)

# Infinite while loop to treat stack of image as video

while True:
    # Reading frame(image) from video


    check, frame = video.read()

    # Initializing motion = 0(no motion)

    motion = 0

    # Converting color image to gray_scale image

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Converting gray scale image to GaussianBlur
    # so that change can be find easily

    gray = cv2.GaussianBlur(gray, (21, 21), 0)

    # In first iteration we assign the value
    # of static_back to our first frame


    if static_back is None:
        static_back = gray
        continue

    # Difference between static background
    # and current frame(which is GaussianBlur)

    diff_frame = cv2.absdiff(static_back, gray)

    # If change in between static background and
    # current frame is greater than 30 it will show white

color(255)
```

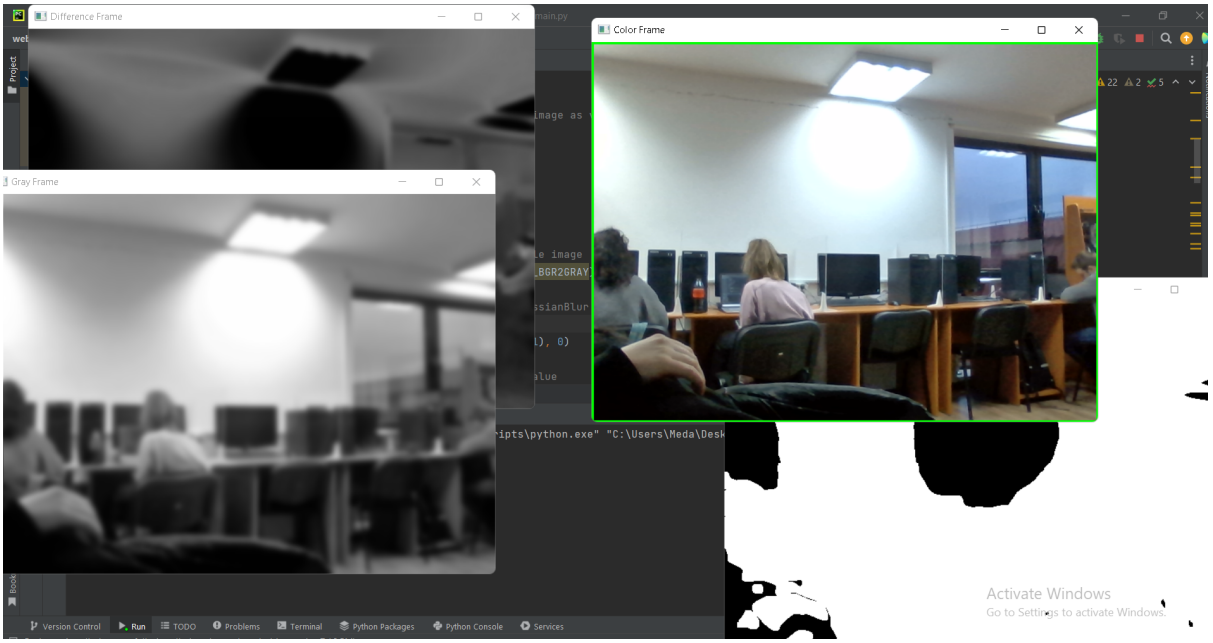
```
    thresh_frame = cv2.threshold(diff_frame, 30, 255,
cv2.THRESH_BINARY)[1]
    thresh_frame = cv2.dilate(thresh_frame, None, iterations = 2)

    # Finding contour of moving object

    cnts,_ = cv2.findContours(thresh_frame.copy(),
                      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    for contour in cnts:
        if cv2.contourArea(contour) < 10000:
            continue
        motion = 1

        (x, y, w, h) = cv2.boundingRect(contour)
        # making green rectangle around the moving object


        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0),
3)

    # Appending status of motion
    motion_list.append(motion)

    motion_list = motion_list[-2:]

    # Appending Start time of motion


    if motion_list[-1] == 1 and motion_list[-2] == 0:
        time.append(datetime.now())

    # Appending End time of motion


    if motion_list[-1] == 0 and motion_list[-2] == 1:
        time.append(datetime.now())

    # Displaying image in gray_scale

    cv2.imshow("Gray Frame", gray)

    # Displaying the difference in currentframe to
    # the staticframe(very first_frame)


    cv2.imshow("Difference Frame", diff_frame)
```

```
    # Displaying the black and white image in which if
    # intensity difference greater than 30 it will appear white

    cv2.imshow("Threshold Frame", thresh_frame)

    # Displaying color frame with contour of motion of object

    cv2.imshow("Color Frame", frame)

    key = cv2.waitKey(1)

    # if q entered whole process will stop
    if key == ord('q'):

        # if something is movingthen it append the end time of
movement
        if motion == 1:
            time.append(datetime.now())
        break

# Appending time of motion in DataFrame

for i in range(0, len(time), 2):
    df = df.append({"Start":time[i], "End":time[i + 1]},
ignore_index = True)

# Creating a CSV file in which time of movements will be saved


df.to_csv("Time_of_movements.csv")

video.release()

# Destroying all the windows


cv2.destroyAllWindows()
```

5.Testing and validation

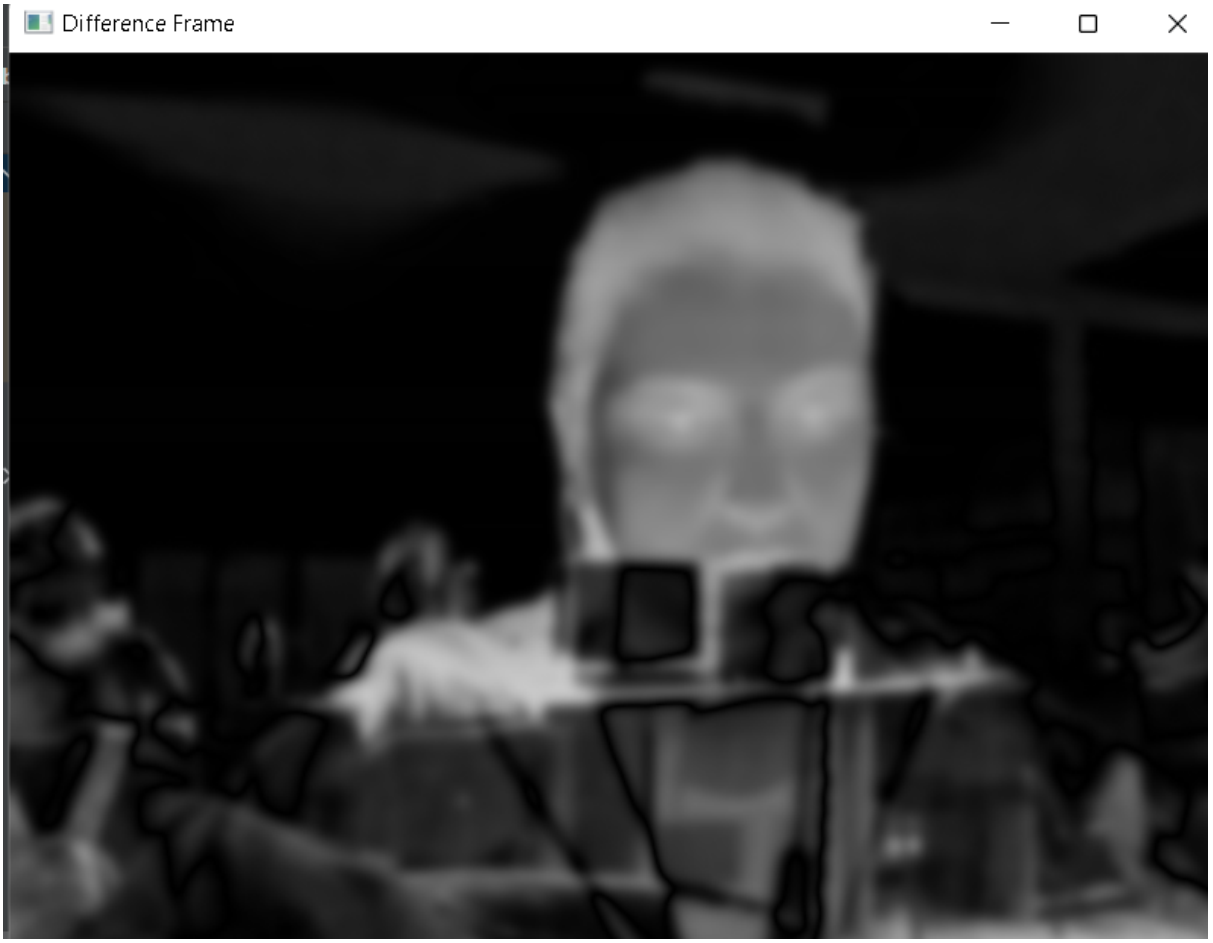
The following happens when the code is run:

- 4 windows open ; the threshold frame, the difference frame, the gray frame and the color frame:



- Due to the quality of the camera, the images (video) are not the best quality and sometimes because of the light a green rectangle will cover the whole of the color frame window.
- keep in mind: different cameras or webcams used in different lighting will yield different results(it means that the threshold would have to be constantly adjusted) - a better quality webcam would work the best
- My camera sees movement because of its poor quality (also poor quality lighting)
- When testing i used a small lamp to help with the light

- when the camera notices movement of an object: in the difference frame you will be able to see that the original frame remains ‘stuck’ in the background. That ghostly image in the back is the old frame used to compare with a new one to detect movement



- Here can be observed; there is an image of chairs from the background, and then the image of a person that moved

6. CONCLUSIONS

The target of this project WAS to create a program able to process video information from a webcam (i.e.the program must identify a moving object and track its position). Videos can be treated as a stack of pictures called frames. different frames(pictures) will be compared to the first frame which should be static(No movements initially). We compare two images by comparing the intensity value of each pixel (a pixel's intensity is its **brightness**).

I liked in this project the challenging aspect of it, and the fact that I got to look deeper into image processing, and see how some inner operations are done, that we think are easy and we take for granted. The most challenging task in my opinion was

understanding the problems my camera had. Finally, from all the papers I've read and the examples I've seen I've come up with my own version of it, implementing it as I would on paper

7.Bibliography

Webcam Motion Detector in Python - Javatpoint
General functions — pandas 1.5.2 documentation (pydata.org)
WebCam Motion Detector in Python - GeeksforGeeks