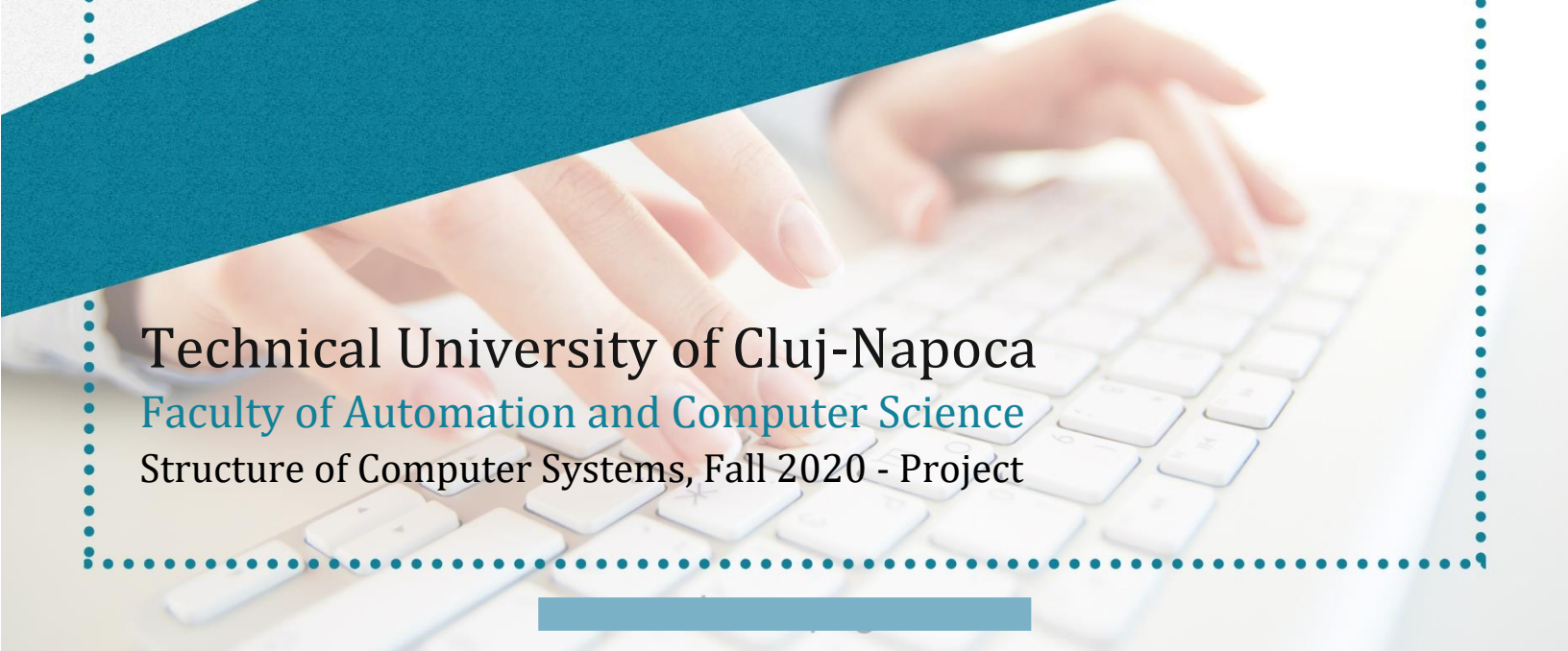
MOTION DETECTION SYSTEM

Using Python & OpenCV Contours

Author: Benedek Balázs

2020



Technical University of Cluj-Napoca
Faculty of Automation and Computer Science

Structure of Computer Systems, Fall 2020 - Project

Contents

1. Introduction

1.1 Context

The goal of this project is to design, implement and test the motion detector system, using image processing by **Python** and **OpenCV** Contours. In many applications based on machine vision, motion detection is used. What is this? It is the detection of the change in the position of an object with respect to its surroundings and vice-versa. *For example*, when we want to count the people that pass by a certain place or how many cars have passed through a gate, we use motion detection. In all these cases, the first thing we have to do is to extract the people or vehicles that are in the scene. There are different techniques, methods, or algorithms that enable this kind of system. I chose the methods used in **OpenCV** and **Computer Vision**.

This system can be used most of the cases in Security scenarios, like detecting if in a museum, during the night there are movements or not, there are burglars or not and so on. This initial project can be connected either to a smarthouse or just to a phone to get messages, notifications about motion in front of a camera. **Internet-Of-Things (IoT)** enthusiasts can even deploy this program on **Raspberry Pi** server and do wonders.

During this documentation, there will be specified how the motion detection algorithm works and how to adjust motion detection parameters for our own needs.

1.2 Specifications

The system will be simulated in **Visual Studio Code (vs code)** using the internal terminal for testing and running the project. There are more ways to test the project's functionality, taking into consideration that there are two videos added to the folder, namely **"frozen.mp4"** and **"papaUT.mp4"**, where the motion detector can work properly, or the simpliest way is to use the **internal camera (using Port 0)** (or a connected WebCam, or the Raspberry Pi Camera) and analysing the real-time video. After running the code four new window will appear on screen (**Gray frame** (blur + grayscale), **Difference Frame** (difference of intensities), **Threshold Frame** (white pixels by difference frame) and **Color Frame** (video with contour around moving objects)).

1.3 Objectives

Videos can be treated as stack of pictures called **frames**. Here I am comparing different frames (pictures) to the first frame which should be static (*No movements initially*). We compare two images by comparing the intensity value of each pixels. In python we can do it easily as we can observe during this documentation and during testing & analising the code (project). The requirements for the project functionality are the followings: **Python or Python3**, **OpenCV** libraries, and **Pandas** libraries, all of these must be installed to be able to run the project properly.

2. Bibliographic study

For detecting motion, we will use the **OpenCV** module. We start with a baseline image, which is the frame captured without any moving object inside it. As soon as the camera fires, the first image is set to our baseline image, which means that we expect no moving object when our program first starts. Next, when somebody enters the frame, certain pixels in that frame will be different. We deduce this difference using the **"cv2.absdiff"** method.

In this project to detect and track motion, there are the following steps that we will achieve:

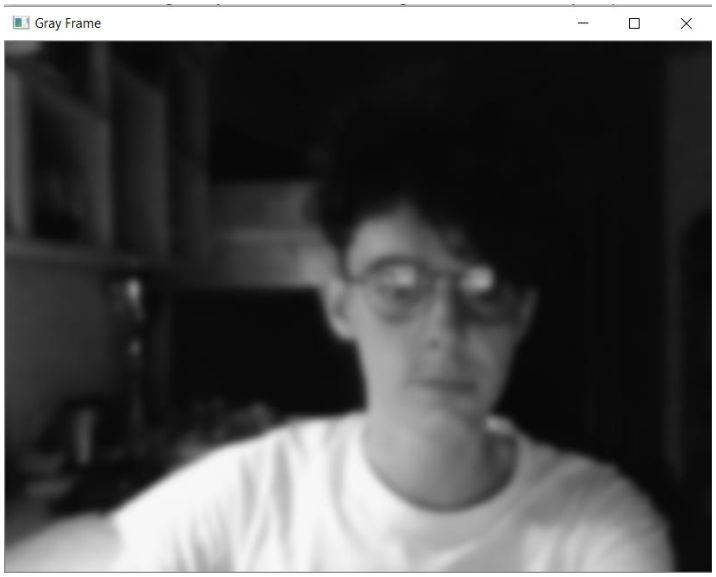
1. Capture the baseline frame (with no object)
    a. Convert the frame to Gray
    b. Smoothen the frame to remove noise
2. Capture the new frame (with object)
    a. Convert the frame to Gray
    b. Smoothe the frame to remove noise
3. Calculate the difference between the two frames
    a. If the difference is greater than the treshold, assume motion is detected
    b. Else assume no motion is detected

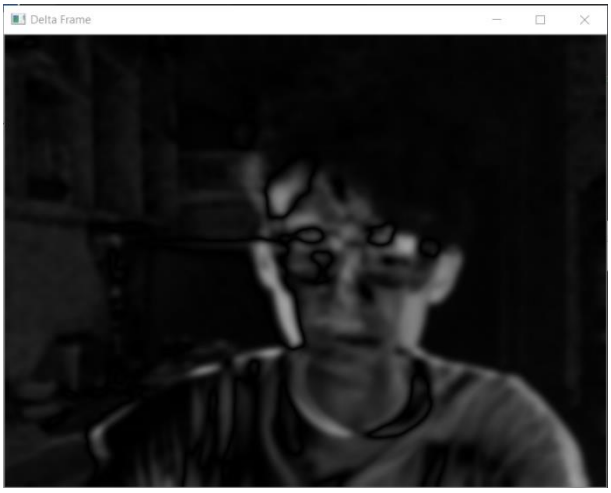
3. Analisys

To solve the motion detector activity, there are 4 windows (frames) used to transform the webcam video into an workspace of being able to track moving objects:

3.1. Gray Frame

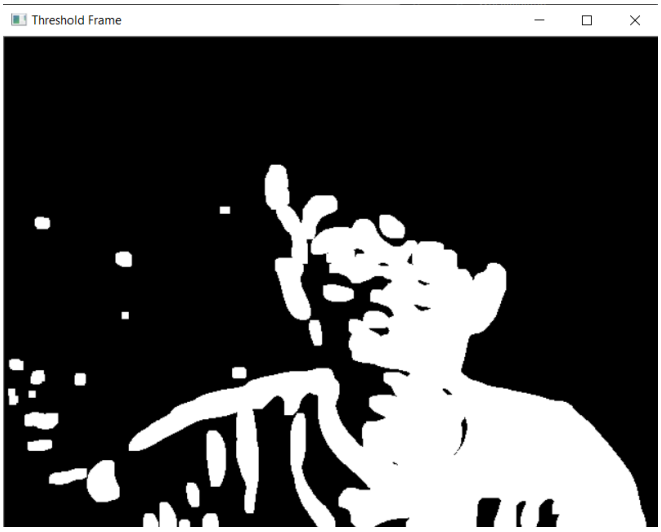
First we will convert the image to gray scale and soften(blur) the image using a Low Pass Filter (LPF). A LPF is generally used in image processing to smoothen the image (eg:- used in skin smoothening, background blurring) and a High Pass Filter(HPF) is used to sharpen the image (eg:- sharpening of eyes and other details). If you ever used photo editing tools like Photoshop/GIMP etc., you must be familiar with this. This helps to increase the accuracy by removing the noise. We are using the GaussianBlur for this purpose. An image after GaussianBlur looks like below.





3.2. Difference Frame

The next step is to deduce the difference between the baseline and the new frame. We pass the two images to the cv2.absdiff() function.

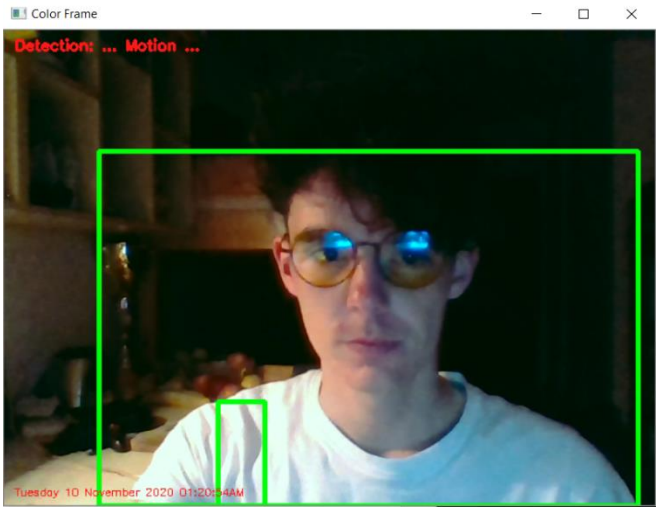


3.3. Threshold Frame

The resulting pucture from the Difference Frame is converted into a binary image using a method called Image Thresholding, meaning, if a particular pixel value is greater than a certain threshold (specified by us here as 35), it will be assigned the value for White (255) else Black(0). Now we have an image which has only 2 types of pixels (pure black or pure white, nothing in between).

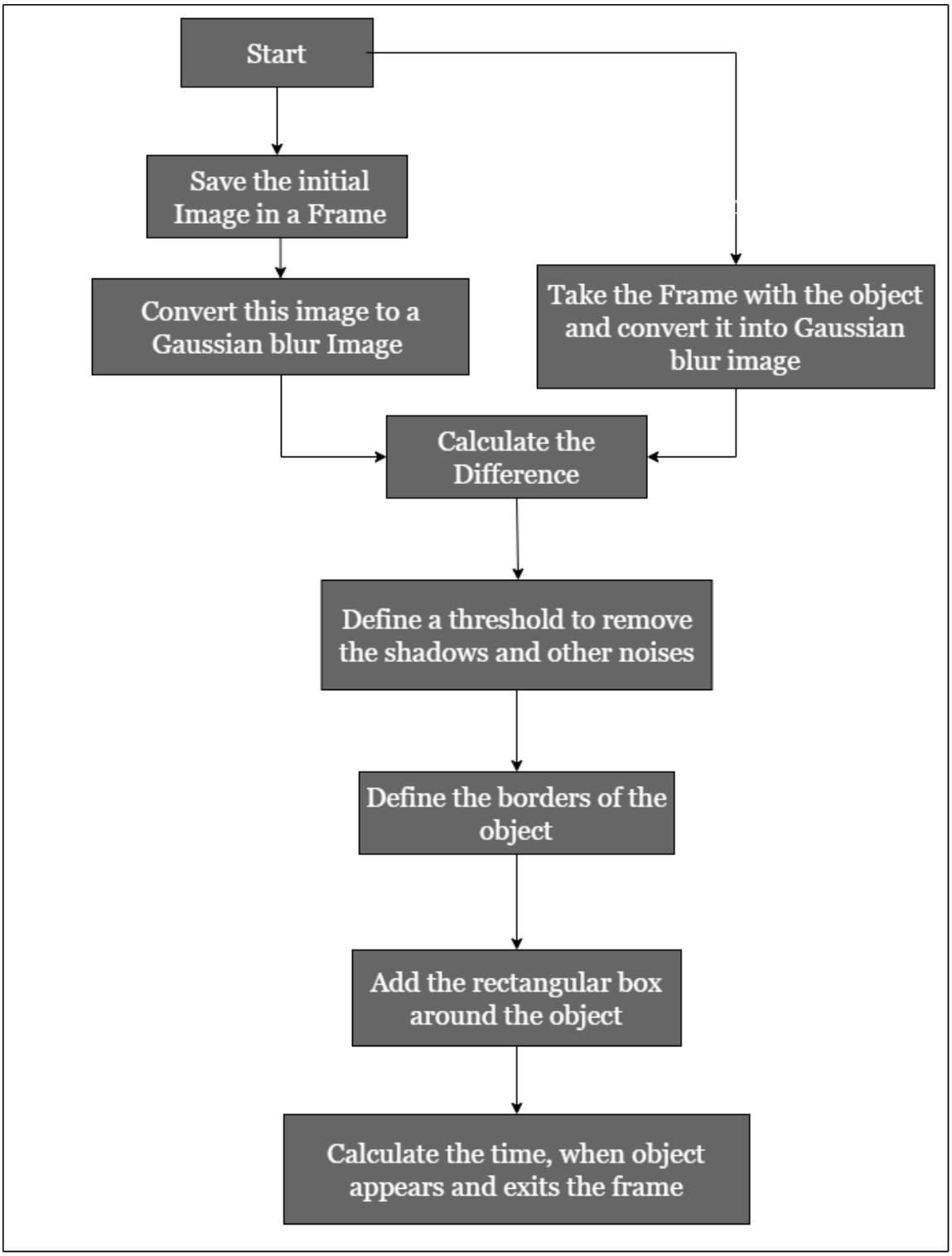
3.4. Color Frame

In this frame we can see the color images in color frame along with green contour around the moving objects.



4. Design

The script **"motion-detector.py"** utilizes the openCV library to capture video from the default webcam of the loacal machine. and we basically store the first frame, which ideally would capture the static background, and then check every coming frame captured from the camera to see if the difference (delta) in all pixels surpasses the set threshold. If it does, we draw rectangular contours on the frame to indicate where the detected objects are.



Listing 5: Solution logic

5. Implementation

5.1 Video capture

Step 1: Capture the video frames from web camera.

```
video = cv2.VideoCapture (0, cv2.CAP_DSHOW)
```

Step 2: Create a backup of live image.

5.2 Motion Detection

Step 1: Capture the video frame of the area to be surveillance called color_frame.

```
color_frame = video.read()
```

Step 2: Convert the the colored video to grayscale, in variable called gray_frame.

```
gray_frame = cv2.cvtColor (color_frame, cv2.COLOR_BGR2GRAY)
```

Step 3: Convert the grayscale image to a blured one, named Gaussian Blur.

```
gray_frame = cv2.GaussianBlur (gray_frame, (21, 21), 0)
```

Step 4: Apply Absdiff() on gray_frame and first_frame (the frame at the beginning to corelate to the evolution of frame) resulting the delta_frame.

```
# capturing only the first gray frame
    if first_frame is None:
        first_frame = gray_frame
        continue

    # creating a delta frame and a threshold frame
    delta_frame     = cv2.absdiff (first_frame, gray_frame)
```

Step 5: if the threshold_frame has a positive value, then motion is detected, else there is no motion.

```
threshold_frame = cv2.threshold (delta_frame, 30, 255, cv2.THRESH_BINA
RY)[1]
```

5.3 Image Filtering

Step 1: Dilate the image to resize the shape of threshold_frame.

```
#dilating the threshold frame and finding pixel contours in it

    threshold_frame = cv2.dilate (threshold_frame, None, iterations=3)
```

After dilation the image becomes patchy at edges. Hence to smooth the appearance Gaussian filter is applied. To identify motion, the motion causing pixels are segregated using binary threshold.

5.3 Detection

Step 1: Reconstruct the estimated motion area in color (BGR)

Step 2: Find contours of all the detected blobs and draw on color_frame

Step 3: Reject smaller blobs by selecting appropriate threshold.

```
# finding the contour area and bounding the end points in a rectangle
    for contour in cnts:
        if cv2.contourArea (contour) < 2000:
            continue
        status = 1

        (x, y, w, h) = cv2.boundingRect (contour)
        cv2.rectangle (color_frame, (x,y), (x + w, y + h), (11, 255, 1), 3)

        text = "... Motion ..."
```

```
# drawing the text and the current time on the frame
    ct = current_time.strftime("%A %d %B %Y %I:%M:%S%p")
    cv2.putText(color_frame, 'Detection: {}'.format(text), (10, 20), cv2.FONT_HER
SHEY_SIMPLEX, 0.5,  (25, 25, 255), 2)
    cv2.putText(color_frame, ct, (10,  color_frame.shape[0]  -  10), cv2.FONT_HER
SHEY_SIMPLEX, 0.35, (25, 25, 255), 1)


    # displaying all 4 different frames
    cv2.imshow("Grey Frame"     , gray_frame)
    cv2.imshow("Delta Frame"    , delta_frame)
    cv2.imshow("Threshold Frame", threshold_frame)
    cv2.imshow("Color Frame"    , color_frame)

    key = cv2.waitKey (33)
    if key == 27 or key == ord ('q'):
        break
```

```
video.release()
cv2.destroyAllWindows()
```

6. Testing and validation

To test this project first of all we need to have the python or python3 installed.

Step 1: Install the opencv: pip install opencv-python

Step 2: Install the pandas: pip install pandas

To run the project use: python motion-detector.py or python3 motion-detector.py

To be sure that the project works well, I used to different videos, chosen from youtube where the picture is pretty clear to be able to make differences between pixels. My webcam's picture is very noisy, thus the program had some difficulties to differentiate these noisy pixels from the actual movements.

7. Conclusions

7.1 Future Scopes

More complex algorithms are necessary to **detect motion** when the **camera itself is moving**, or when the motion of a specific object must be detected in a field containing other movement which can be ignored. An example might be **a painting surrounded by visitors in an art gallery.**

7.2 Summary

After I finished this **project**, I reckon that my mind has widened while I was able to dive into the Python implementation, improving **my python skills**, learning new techniques, and working more beautiful than until now. There is no doubt this project was a good continuing for our set of experiences.

During this project I have got more confidence to work with opencv. I could strengthen as well my affirmation that starting work from time is a right way to have more hours to think more, because modelling the project in a correct way from the beginning helps you to speed-up the working stages. Facing minor or major problems, I realized that having time for research has the benefit of the capability to be absorbed in surfing the internet, to improve our acquaintance and to be pleased by yourself.

8. Fact

Samuel Bango

The first motion sensor was invented in the year **1950** by Samuel Bango named as a burglar alarm. He applied the basics of a radar to ultrasonic waves – a frequency to notice fire or robber and that which human beings cannot listen to. The Samuel motion sensor is based on the principle of "*Doppler Effect*".

9. Bibliography

- https://www.youtube.com/watch?v=-ZrDjwXZGxI
- https://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/
- https://likegeeks.com/python-image-processing/
- https://www.udemy.com/course/opencv-project/