Department of Computer Science
Technical University of Cluj-Napoca

Structure of Computer Systems

Laboratory activity 2019-2020

Project title: Surveillance System

Name: Mateiu Bianca
Group: 30434
Email: mateiu.bianca@gmail.com

Contents

Chapter 1

Research

The project I want to develop is an online Surveillance System, that uses the webcam. The functioning is simple: the system processes each frame captured by the camera, and when it detects motion, it reports it. The system will also generate a video file, containing only the parts of the recording that had some sort of movement.

Video surveillance is no new technology, and it has been around for a while. It has proven itslef useful in many sectors of our society, but lately, it has been a significant increase in the development of home securtiy systems. My project was developed with the same purpose in mind.

After some research concerning the available tools and technologies, I have decided to use OpenCV for video processing, and Flask for the Web Application.

1.1 OpenCV

OpenCV was started at Intel, in 1999, and the first release came out in 2000. It supports a wide variety of programming languages, among wich *Python*, which I have decided to use.[1]

Among OpenCV's application areas we can find:[3]

- facial recognition system
- gesture recognition
- human-computer interaction
- motion tracking

We can see that the problem we are dealing with is similar to that of motion tracking, only it is more simplified. Instead to locating and tracking a moving object, we simply have to detect if *something* moved in the scene, and then trigger some actions, such as saving the frames to a video file.

I have installed OpenCV using the package installer for Python, *pip*, by running the command `pip install opencv-python`

1.2 Flask

Flask is a simple, lightweight WGSI(*Web Server Gateway Interface*) web application framework. [2], used for building websites in Python. I chose to use this framework to stream the video recorded by the web camera. I have installed it using pip as well, with the command `pip install Flask`

Chapter 2

Analysis and design

In order to create a Video Surveillance System, the first thing we require is, of course, a video camera. For this project, I will be using the Web camera of my laptop, but if the project were to be developed further, an actual security camera would be more suited for the job.

The next step is to access the frames captured by this camera, and we can easily do this using OpenCV. Also using OpenCV, we must compare consecutive frames, and find the difference between them. When this difference is high, we can assume that motion has been found. We will use the standard deviation of the difference between frames, which is is a measure of the amount of variation or dispersion of a set of values.

If motion is detected, than the frame will be written to a '.avi' video file, and saved on the local hard-drive. Also, we can see when the system detects motion beacuse the program will write a text on the video streamed on the Web Application.

In regards of the Web Application, this will have a home page, containing only a button, that will start the Video Surveillance when it is pressed. The pressing of this button also concludes to the generation of a new '.avi' file on the hard drive of the computer, containing the frames that captured movement. When the user clicks the button, he will be redirected to another page, containing the video streaming. This page will also have a button, that will redirect the user back to the home page.

Chapter 3

Implementation

The project contains two python files:

1. camera
2. main

3.1 camera.py

This file contains the declaration of the *VideoCamera* class, that deals with capturing the video, reading and comparing frames, detecting movement and writing the video file containing motion.

The first step is to import cv2, and start the video capturing:

```
    self.cap = cv2.VideoCapture(0)
```

This line captures the video stream from camera source, while the parameter, 0, refers to the first camera.

The most important method of this file is *computeDifference*. I will go throught this method line by line, and I will also show through pictures the output of each line:

```
    _, self.frame1 = self.cap.read()
    gray1 = cv2.cvtColor(self.frame1, cv2.COLOR_BGR2GRAY)

     _, self.frame2 = self.cap.read()
    gray2 = cv2.cvtColor(self.frame2, cv2.COLOR_BGR2GRAY)
```

The first step is to read 2 frames, and convert them to grayscale. In picture (a) from 3.1, we can observe the frame captured by the video camera (without conversion). The reason we convert them to grayscale is beacuse it for the method we will use next to compute the absolute difference between frames, grayscale images are required.

```
    dist = cv2.absdiff(gray1, gray2)
```

The next step, as I said, is to compute the absolute difference between the two images.(See 3.1 picture (b)) The output of this function will be an *intensity image*, an image that will have values between 0 and 1, so it will not be just black and white. So we will use the *threshold* function to obtain a black and white image(3.1 picture (c)):

```
    _, self.thresholdImg = cv2.threshold(dist, 20, 255, cv2.THRESH_BINARY)
```

What this function does is go over every pizel value, and if it is greater than a threshold value (second parameter, 20 in our case), it is assigned one value (white), else it is assigned another value (black). Next, we want to reduce the noise in the image, by applying a filter:

```
    self.thresholdImg = cv2.GaussianBlur(dist, (9, 9), 0)
    _, self.thresholdImg = cv2.threshold(self.thresholdImg, 20, 255,
                                                          cv2.THRESH_BINARY)
```

After applying the filter the output is again an intensity image(3.1 picture (d)), so we need to apply the threshold function again(3.1 picture (e)). Lastly, we calculate the standar deviation of the image, and return it(3.1 picture (f)):

```
    _, stDev = cv2.meanStdDev(self.thresholdImg)
    return stDev
```

In the method *getFrame*, which is called from the *main* file, the value returned from the previous method is copared to a treshold deviation chosen by us (the lower the value, the greater the sensitivity of the motion detection system), and if it is higher, than we consider that motion was detected.

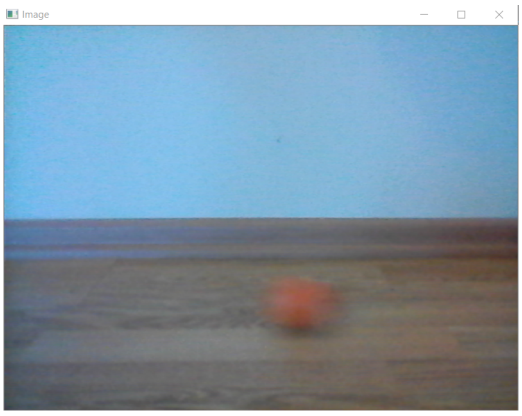
If motion was detected, than write both frames to the video file (so the saved video will look as the streamed one), and write the text: 'REC' and 'MOTION DETECTED' on the image that will be returned from this method, to signal the user that the video is being recorded, and that motion was detected. In order to write text on the frames, we use another method from OpenCV, *putText*.

Lastly, encode the frame to 'jpeg' format, and return it.

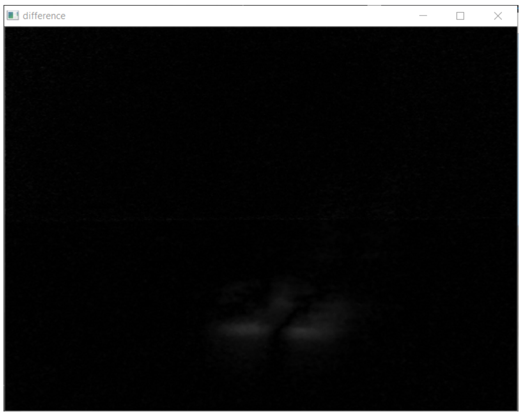
3.2 main.py

This file deals with the Web app page generation, and is the file that has to be run in order to start the project.

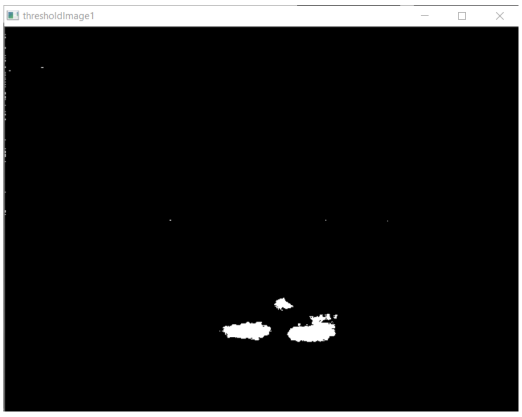
The Web app has 2 pages: '/' and '/recording' 3.2, which render 2 different html template files, stored in the *templates* directory. In the html file rendered in the '/recording' file, there is an *img* tag, that has as source the url for another route defined in *main.py*: '/video_feed'. This method return a Response, containing the frame returned from the VideoCamera class.



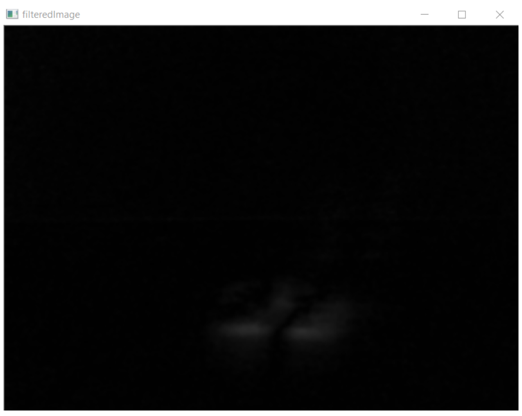
(a) Image captured from video camera.



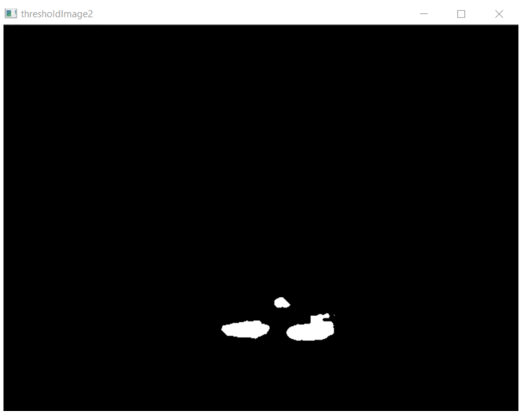
(b) Difference image between 2 frames.



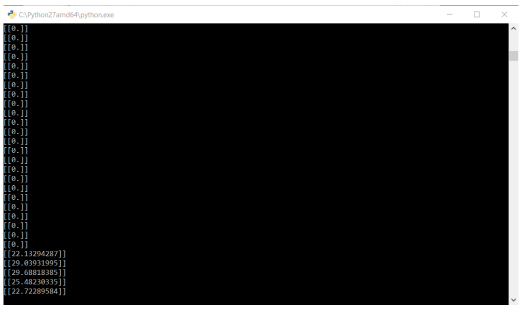
(c) Threshold image.



(d) Filtered image.



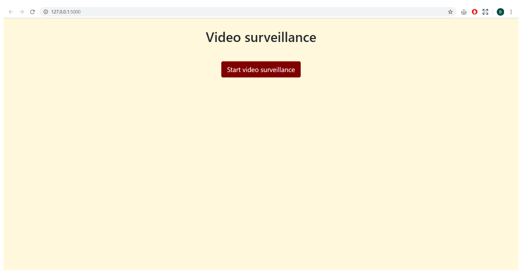
(e) Second threshold image.



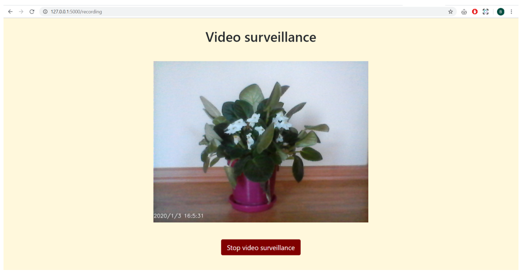
```
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[22.13294287]]
[[29.03931995]]
[[29.68818385]]
[[25.48230335]]
[[22.72289584]]
```

(f) Output of standard deviation.

Figure 3.1: Example of program: rolling ball target



(a) Home page.



(b) Recording page.

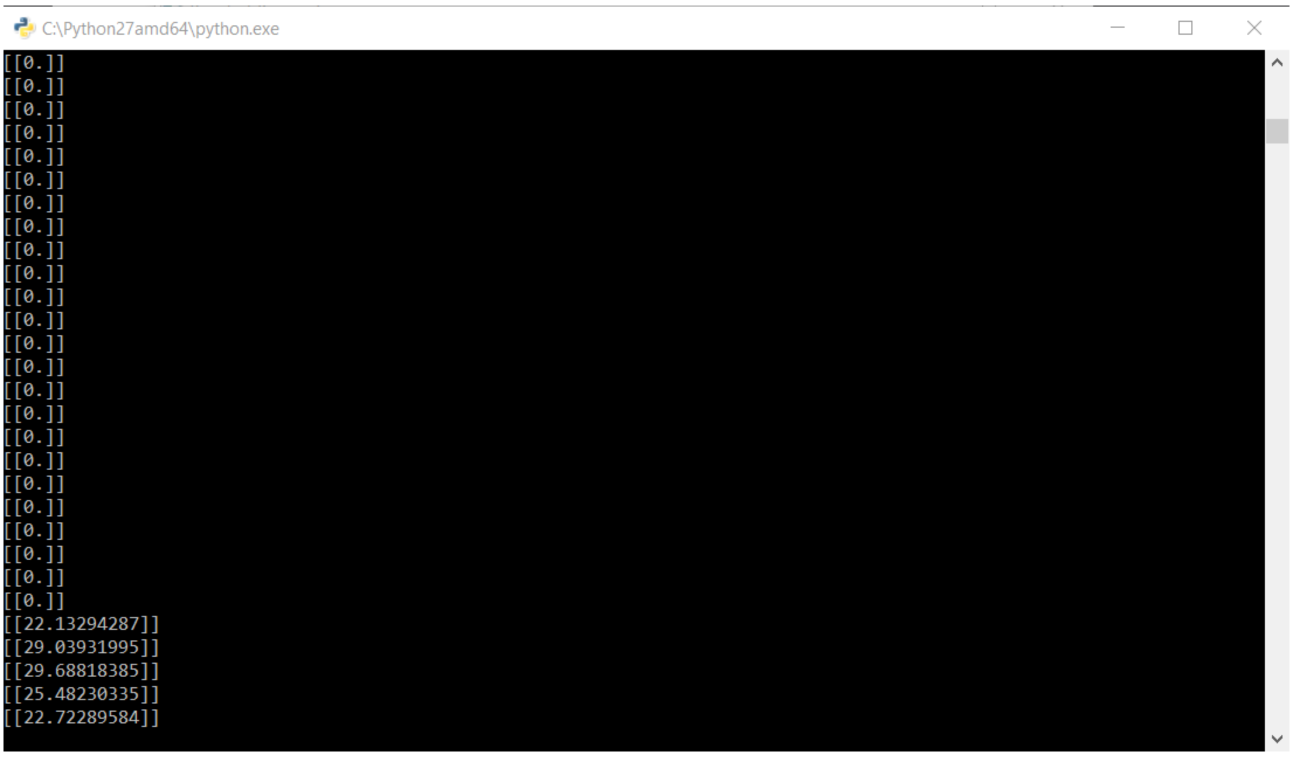
Figure 3.2: Web application layout

Chapter 4

Testing and validation

In regards to the testing of the program, I have run several experiments with it, that generated what I consider to be satisfying results. I have also experimented with different values for the OpenCV functions, that made the system more or less sensitive to movement, and the current configuration I believe to be the best one.

The corectness of this program can also be observed in picture 4.1, in the output values of the standard deviation. We can see that before the ball started rolling, the frame was a static one, therefore the output was 0. When the ball started rolling, we can see the standard deviation started having positive values.



```
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[0.]]
[[22.13294287]]
[[29.03931995]]
[[29.68818385]]
[[25.48230335]]
[[22.72289584]]
```

Figure 4.1: Standard deviation values

Bibliography

[1] Opencv-python tutorials. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html.

[2] The pallets projects. https://www.palletsprojects.com/p/flask/.

[3] Wikipedia contributors. Opencv — Wikipedia, the free encyclopedia, 2019. [Online; accessed 3-January-2020].