Monitoring System with Web Camera

Crisan Oana Andra

Group 30431/1

1. Introduction

The purpose of this project is to develop and implement a system that enables monitoring a certain area captured by the laptop's webcam on the Internet. The user will be informed via an alarm in the event of an unexpected circumstance (such as the presence of a moving object in the room).

Everyone may use this system because it is affordable and compatible with any laptop or computer that has a webcam. Safety is undoubtedly one of the most essential aspects for a person, so why not bring it into the comfort of your own room?

This device will be simulated in the IDE provided by Eclipse or IntelliJ and then an Internet page will pop up. The webcam's view will be visible to the user. An audible signal will be activated in the event of any strange movement.

Java typically doesn't offer simple access to the computer's hardware. Because of this, utilizing Java to access the webcam could be challenging. Taking this into consideration, I chose to use Python. A cross-platform library called OpenCV allows us to create real-time computer vision applications. The primary areas of focus are image processing, video capture, and analysis, which includes features for object and face detection. This library processes the digital images, and it stores them in a special data type: “Mat”.  It represents an n-dimensional array and is used to store image data of grayscale or color images, voxel volumes, vector fields, point clouds, tensors, histograms, etc.

This class comprises of two data parts: the **header** and a **pointer**

* **Header** − Contains information like size, method used for storing, and the address of the matrix (constant in size).
* **Pointer** − Stores the pixel values of the image (Keeps on varying).

Motion detection is the process of identifying a shift in an object's position in relation to its environment or a shift in the environment in relation to an object. Either mechanical or electronic techniques can be used to accomplish it. Motion perception refers to the process when it is carried out by living organisms. Motion detectors are frequently included as part of systems that carry out tasks automatically or notify users when motion is detected in a particular region. An occupancy sensor picks up on a person or object entering or leaving a certain area. As game controllers, motion controllers are also employed with video game consoles. A camera can also allow the body's movements to be used for control, such as in the Kinect system.

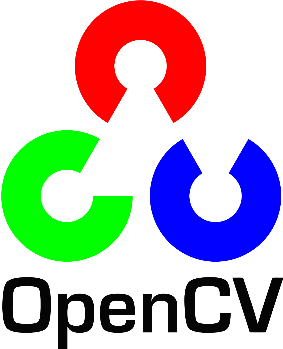
Microsoft's range of motion-sensing input devices known as Kinect was originally made available in 2010. The devices typically include RGB cameras, infrared projectors, and detectors that map depth using time of flight or structured light calculations. These components can be utilized, among other things, for real-time gesture identification and body skeleton detection. Additionally, they have microphones that can be utilized for voice control and speech recognition.

The idea underlying motion detection in this application is rather straightforward. Checking for differences between the frames is required. That refers to comparing the pixel values between the current frame and the previous one.

To notify the user of any significant background changes, a Java sound file will be played. Using multiple Java packages, such as sun.audio, javax.sound, etc., makes this task simple to do.

1. Bibliographic study

The OpenCV library, which enables us to build real-time computer vision applications and gives us access to our computer's webcam, must be one of the most significant tools utilized in this project.



A library of programming functions with a focus on real-time computer vision is called OpenCV (Open-Source Computer Vision Library). It was first created by Intel and afterwards sponsored by Willow Garage and Itseez (which was later acquired by Intel).

In addition to the obvious requirement of a camera, we also need a few more items in order to display a picture from one. JavaFX will be used to display the photos. We need a mechanism to convert an OpenCV Mat to a JavaFX Image because we'll be displaying the photos our camera takes in an ImageView.

This library processes the digital images, and it stores them in a special data type: “Mat”.  It represents an n-dimensional array and is used to store image data of grayscale or color images, voxel volumes, vector fields, point clouds, tensors, histograms, etc.

This class comprises of two data parts: the **header** and a **pointer**

* **Header** − Contains information like size, method used for storing, and the address of the matrix (constant in size).
* **Pointer** − Stores the pixel values of the image (Keeps on varying).

The next step is to translate our Mat into bytes, which are then translated into an image object. The camera view will be streamed to a JavaFX Stage first. The library will then be initialized using the loadShare() function. To display the image, we'll then build a stage containing a VideoCapture and an ImageView. To set the image, we also need to establish an AnimationTimer. Finally, the Mat is converted to an Image via our getCapture method.

Now that a window has been created, the program must live-stream the camera's view to the imageView window. We need a way to determine whether movements have taken place by comparing each pixel between the current image and the motion buffer image.

Diagram

Description automatically generated

1. Analysis and Design

By analyzing the algorithm for motion detection, we see that in order to design the device we need to transform the information from RGB to grayscale, because that’s how the OpenCV library works. The data might be stored in NumPy type arrays since they are faster, and they can only contain one type of data. This type also allows us various operations, such as calculating the mean value which we will need later.

Anyway, the most important part is calculating the difference between the mean value of pixels between the current frame and the previous one. If this difference is bigger that a given value (it depends on how sensitive to movement we want our detector to be, but it has to be greater than zero) then it means that there is movement, and the system should notice the user by beeping.

When the user is done using the application, he/she may press the “Q” button on the keyboard and that should turn off the application immediately. Also, the user can check the time when there was movement by opening the text file that was created in the project’s folder.

Diagram

Description automatically generated

Success Scenarios:

* When motion is detected:

1. Open the application
2. The application detects movement
3. Hear the sound
4. Close the application by pressing ‘Q’
5. Check the text file with the times of the movement

* When motion is not detected:

1. Open the application
2. No movement is detected
3. Close the application (it is also specified in the console that here was no movement detected)

Fail Scenario:

* This might happen if the camera is not detected
* The user will be informed about this via an error message

1. Implementation

The implementation of this project seemed complicated at first, but it’s actually very simple and short. The main part consists of a while loop in which some functions repeat themselves until the ‘Q’ key is pressed.

At first, the current frame is set in the variable frame which has the following parameters (height, width and channels (=3 RGB)) and is a special type from the OpenCV library. Then, we need to transform the information contained in frame from RGB to grayscale in order to save it in the “NumPy” type array. We calculate the difference between the last 2 frames to find out if they are different and we reinitialize the last frame with the current frame.

Text

Description automatically generated

The next step is to check if the motion is greater than a chosen value. This value depends on how sensitive we want our sensor to be. I personally chose 0.3. If there is motion detected we set the “detected\_motion” to True and we call the following functions which were implemented above the loop: playSound(), writeInFile() and takeSnapshot().

Text

Description automatically generated

An additional verification is necessary since somehow for the first ever frame it always detects motion, so we simply skip it.

Graphical user interface, text

Description automatically generated

Functions:

writeInFile() – this function writes the times when motion is detected in a text file which can be found in the project’s folder

Graphical user interface, text

Description automatically generated

takeSnapshot() – this function uses a special command “imWrite” from the OpenCV library which saves in the project’s folder a picture of the current frame each time motion is detected

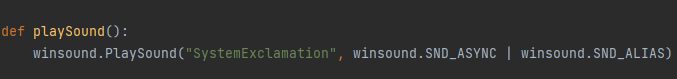
Text

Description automatically generated

playSound()

- this function uses a function from the “winsound” library in order to make a noise when there is motion

- one of the parameters of this function is “winsound.SND\_ASYNC” which makes it possible for the program not to stop playing the sound until is finished at each iteration of the while loop (so basically this prevents the program form lagging and makes working with threads unnecessary)



1. Results and Testing

When we run the project, a frame that shows the images from camera will pop up. If our system detects motion, the user will hear a beep.

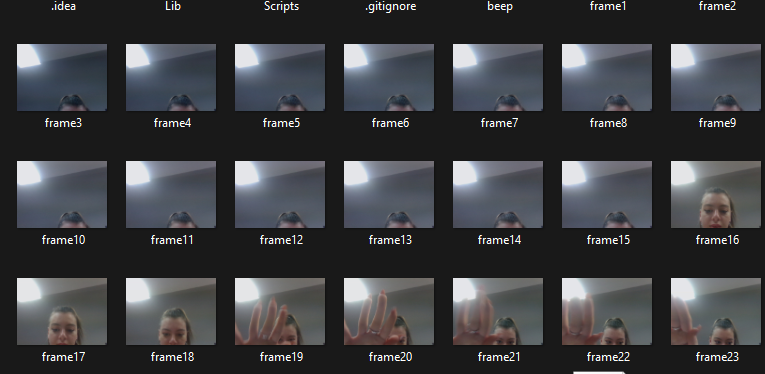
A group of people in a room

Description automatically generated with medium confidence

Text

Description automatically generated

When the motion is detected, some screenshots with the moving object will be presented in the project’s folder:



Also, the time of the motion will be presented in a text file:

Graphical user interface, text

Description automatically generated with medium confidence

1. Conclusion

To conclude, working on this project has thought me how to use the OpenCV library and to learn more about how frames work. Of course, that this system could be much more developed by taking care of the sensitivity of the motion detector. Initially, I wanted to work in Java, but I realized that Python is a much more fit programming language in order to implement these kinds of requirements.

7.Bibliography

<https://www.geeksforgeeks.org/python-opencv-capture-video-from-camera/>

<https://subscription.packtpub.com/book/application-development/9781785283932/3/ch03lvl1sec28/accessing-the-webcam>

<https://www.kdnuggets.com/2022/08/perform-motion-detection-python.html>