Motion detection camera

**Lungoci Luca**

**group 30434**

* **Introduction**

The purpose of this project was to design an application that uses a web camera to detect motion in a room. The images should be transmitted on a web site and a sound should be played when some movement is detected.

This project has some absolute paths ! The files used are in scs\src\assets.

* **How to run the application**

Run the main class of the java project and for the angular project use the command “ng serve” while in the “scs” directory. The webpage should be found at “localhost:4200”.

* **How it works**

Below is how the picture, in the case of a detected motion, is transmitted on the website. Two consecutive frames are stored and each pixel is compared in order to catch a moving object that just entered the supervised scene. A rectangle is drawn around the pixels that changed their value so that the moving object is highlighted.



Here is another example of the result of the application where the part detected as movement in the scene highlighted with red.



* **How it was done**

For this application I separated the code in two parts: one for backend and one for frontend.

The backend part consists in a Java program. Here is where most of the work is done: taking photos through the web camera, comparing frames, highlighting the moving object, playing a sound when motion is detected. The result of the backend is a photo to be displayed on the website by the frontend.

The frontend consists of a Angular project. Its job is to create a website and display the picture. It also refreshed each second to update the picture.

* **The code**

The part I have taken from the internet is the code for playing an audio file and using the web camera:

<https://github.com/sarxos/webcam-capture>

<https://www.geeksforgeeks.org/play-audio-file-using-java/>

The thread of the backend project: The steps that the “run” method performs are:

1. initialize a scanner for reading the configuration file.
2. read the configuration file (the relevant value in that file is the second integer, if it is equal to “1” then the pixels corresponding to the moving object will be drawn with red).
3. take to pictures, send the two pictures to the “ProcessImage” class to be compared and to highlight a potential moving object.
4. Delay the execution by 1 second (if the .jpg file to be displayed on the website is changing too fast then nothing will be seen on the sceen).
5. From the method of the “ProcessImage” class we get a Boolean, alert = true if movement was detected, it is used to know if we play or stop the alarm sound.
6. The modified picture is saved in the file system to be later on displayed on the webpage.

@Override  
public void run() {  
 Webcam webcam = Webcam.*getDefault*();  
 webcam.open();  
  
 while (run.get()) {  
 Scanner scanner = null;  
 try {  
 scanner= new Scanner(file);  
 } catch (IOException e) {  
 System.*out*.println("Io exception");  
 }  
  
  
 int value1 = scanner.nextInt();  
 int displayMode = scanner.nextInt();  
 System.*out*.println(value1 + " <->" + displayMode);  
  
 previousFrame = webcam.getImage();  
 currentFrame = webcam.getImage();  
  
 boolean alert = ProcessImage.*compareFrames*(currentFrame, previousFrame, *differenceBetweenFramesConsideredAsMovement*, displayMode);  
  
 try {  
 Thread.*sleep*(1000);  
 } catch (Exception e) {  
 System.*out*.println("err");  
 }  
  
 if(alert && !previousAlert)  
 sound.play();  
 else if(!alert)  
 sound.stop();  
  
 previousAlert = alert;  
  
 try {  
 ImageIO.*write*(currentFrame, ImageUtils.*FORMAT\_JPG*, new File("C:\\Users\\lungo\\IdeaProjects\\AngularProject\\scs\\src\\assets\\picture.jpg"));  
 } catch (IOException e) {  
 System.*out*.println("Io exception");  
 }  
 }  
}

Code for taking a picture

Webcam webcam = Webcam.*getDefault*();  
webcam.open();

previousFrame = webcam.getImage();  
currentFrame = webcam.getImage();

Code responsible for drawing a rectangle around the moving object:

private static boolean drawRectangle(BufferedImage currentFrame, byte[][] matrix, int displayMode) {  
 int leftLimit, upperLimit, bottomLimit, rightLimit;  
 leftLimit = upperLimit = 999999;  
 rightLimit = bottomLimit = -1;  
  
 for(int x = 0; x < currentFrame.getWidth(); x++) {  
 for (int y = 0; y < currentFrame.getHeight(); y++) {  
 if(matrix[x][y] == 1) {  
  
 if(displayMode == 1)  
 currentFrame.setRGB(x, y, 16711680);  
  
 if(leftLimit > x)  
 leftLimit = x;  
 if(upperLimit > y)  
 upperLimit = y;  
 if(rightLimit < x)  
 rightLimit = x;  
 if(bottomLimit < y)  
 bottomLimit = y;  
 }  
 }  
 }  
  
 System.*out*.println(rightLimit+" "+leftLimit+" "+upperLimit+" "+bottomLimit+" ");  
  
 if( rightLimit < 0 || rightLimit >= currentFrame.getWidth() || leftLimit >= currentFrame.getWidth() || upperLimit >= currentFrame.getHeight() || bottomLimit < 0 || bottomLimit >= currentFrame.getHeight())  
 return false;  
  
 for(int i = leftLimit; i < rightLimit; i++) {  
 currentFrame.setRGB(i, upperLimit, 255);  
 currentFrame.setRGB(i, bottomLimit, 255);  
 }  
  
 for(int i = upperLimit; i < bottomLimit; i++) {  
 currentFrame.setRGB(rightLimit, i, 255);  
 currentFrame.setRGB(leftLimit, i, 255);  
 }  
 return true;  
}

I also wrote a function to remove some pixels considered as movement because the camera sometimes sees some random pixels as movement, scattered around the scene. You can see there “if(neighboursOfPixel >= 3)” , if one changed pixel has 3 unchanged pixels => probably there is no moving object there, just some adjustments made by the camera.

private static byte[][] clearFalseMovement(BufferedImage currentFrame, byte[][] matrix) {  
 for(int x = 0; x < currentFrame.getWidth(); x++) {  
 for (int y = 0; y < currentFrame.getHeight(); y++) {  
 if(matrix[x][y] == 1){  
 byte neighboursOfPixel = 0;  
 try{  
 if(matrix[x-1][y] == 0)  
 neighboursOfPixel++;  
 if(matrix[x+1][y] == 0)  
 neighboursOfPixel++;  
 if(matrix[x][y-1] == 0)  
 neighboursOfPixel++;  
 if(matrix[x][y+1] == 0)  
 neighboursOfPixel++;  
  
 if(neighboursOfPixel >= 3) {  
 matrix[x][y] = 0;  
 }  
 }catch (Exception e) {  
  
 }  
 }  
 }  
 }  
 return matrix;  
}

Below is where the two frames are compared. This function also receives as argument an integer for the difference between pixels that we want to consider as movement. If the value of a pixel changes little from one frame to another, we would not want to assume that an object is moving in our scene. This value was set by me after some modifications and testing.

private static final int *differenceBetweenFramesConsideredAsMovement* = 1160000;

public static boolean compareFrames(BufferedImage currentFrame, BufferedImage previousFrame, int differenceBetweenFramesConsideredAsMovement, int displayMode) {  
 byte[][] matrix = new byte[currentFrame.getWidth()][currentFrame.getHeight()];  
 for(int x = 0; x < currentFrame.getWidth(); x++) {  
 for (int y = 0; y < currentFrame.getHeight(); y++) {  
  
 int dif = Math.*abs*(currentFrame.getRGB(x, y) - previousFrame.getRGB(x, y));  
  
 if (dif > differenceBetweenFramesConsideredAsMovement) {  
 //currentFrame.setRGB(x, y, 16711680);  
 matrix[x][y] = 1;  
 }  
 else  
 matrix[x][y] = 0;  
 }  
 }  
  
 for(int i = 0; i < 5; i++)  
 matrix = *clearFalseMovement*(currentFrame, matrix);  
  
 return *drawRectangle*(currentFrame, matrix, displayMode);  
}