VIDEO SURVEILLANCE SYTEM USING WEB CAMERAS

Stioiu Denis-Adrian

TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

Contents

1.Introduction

1.1 Context

The goal of this project is to design and implement a surveillance system using a web camera, and motion detection. The application will be connected on the local server of a computer and anyone with the IP of the network can connect to the application. If any motion is detected by the camera the users will be by a message on the video. The user will receive an SMS when the system starts streaming.

The application can be used by people who want to supervise their home, and check if there are people who try to trespass their property.

1.2 Specifications

The application will be programmed in the PyCharm IDE provided by Jetbrains, using the OpenCV library to detect motion and work with the web camera. The camera will be connected to a LAN socket in order to stream live and the project uploaded on a local server.

1.3 Objectives

Design and implement a Survaillance System that detects motion from the live stream of the camera and alerts the user if there is motion detected along with a warning . The camera will send a stream of images and the application will check if there is any motion, and alert the user.

2. Bibliographic study

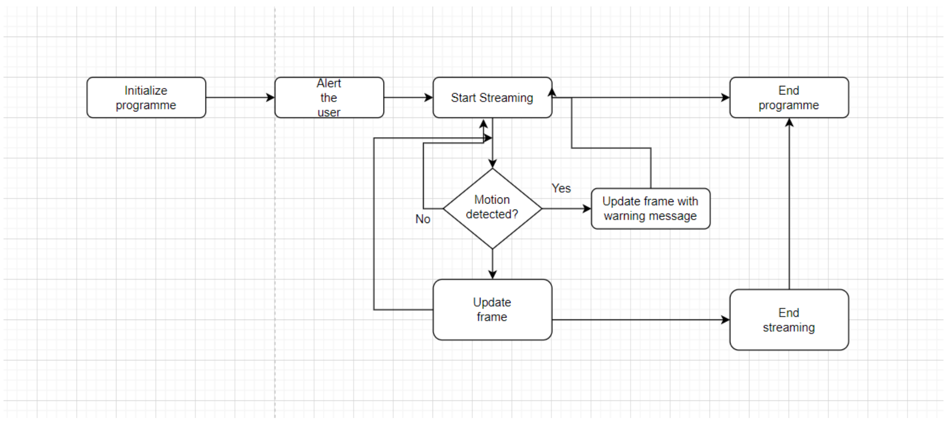
OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. The algorithms of the library can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, etc.

The background of the video stream is largely static and unchanging over consecutive frames of a video. Therefore, if we can model the background, we monitor it for substantial changes. If there is a substantial change, we can detect it, this change normally corresponding to a motion on our video. The background subtraction involves calculating a reference image, subtracting each new frame from this image, and thresholding the results.

OpenCV is used to calculate absolute frame deltas against the most recent saved frame and the current frame. The frame deltas are then passed through a threshold filter, and bounding boxes are drawn around contours of the thresholded frame. A sufficiently large bounding box derived from the contours of a thresholded frame delta image is considered movement.

3. Analisys

3.1 Motion Detection



The streaming of the video is performed by encoding the frames to a JPG format that will be continuously uploaded to the server. OpenCV is implemented on the camera to process the image before streaming. The approaches used are background subtraction, bounding rectangles, and applying text to the video.

When the system starts streaming the users will be alerted through an SMS that will contain the link to the website.

The background subtraction obtained frame delta is compared to a threshold value of 255 to identify the difference in image regions corresponding to motion. Contour detection is applied to those regions, and once the motion is detected the objects moving will be bounded to a rectangle.

4. Design

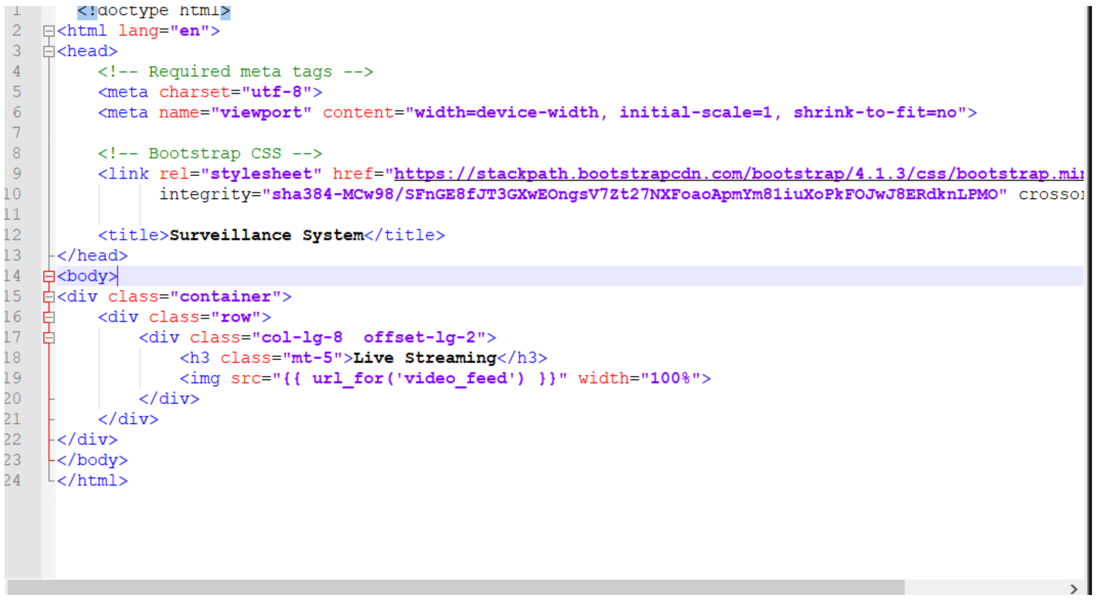
The application is streamed using the Flask framework. Flask is a micro web framework for Python, an is classified as micro because it does not require particular tools or libraries. The streaming of the video is done by continuously uploading a new frame to the page, and rendering a basic HTML page, through a page component id.

The SMS is sent with the help of Twilio API, where a user makes an account on their platform and receives a phone number to use for sending alerts, notifications, etc. When finishing the registration each account has some security tokens (Account Sid and Auth token), and we will create a template message that will be sent at the start of the application in the first three seconds. This is done by saving the time in seconds when the application started, and the current time, and if the difference between them is less than three seconds an SMS will be sent. I chose the 3 second interval in order not to spam the user when the application is streaming.

The motion detection part is integrated in the video stream sent to the server. We use OpenCV to capture the video streamed by a web cam(in this project the laptop’s web cam), we set its width, height and fps(frames per second). We firstly read two frames one after another, and compute their absolute difference (the difference between the matrix of pixels of each frame, and save as the separate frame). The next part is converting the difference obtained into gray scale, in order to be easier interpreted. Then we will apply Gaussian Blur to the frame in order to find the changes easier. Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function. It is used to reduce the details in a picture(in our case the frame) and reduce the image noise. If the change in the background and current frame is greater than a threshold value of 30, then will show white color. The threshold image will be dilated in order to find better contours. Now, we traverse the list of contours and apply bounding rectangles for contours with a surface on the camera greater than 5000 pixels (in order to avoid adding contours to smaller objects), as 5000 would be an area big enough for a motion of a human. Then if there is motion there will be a warning on the video stream saying “Movement detected”. The next step is to update the frame and check again.

5. Implementation

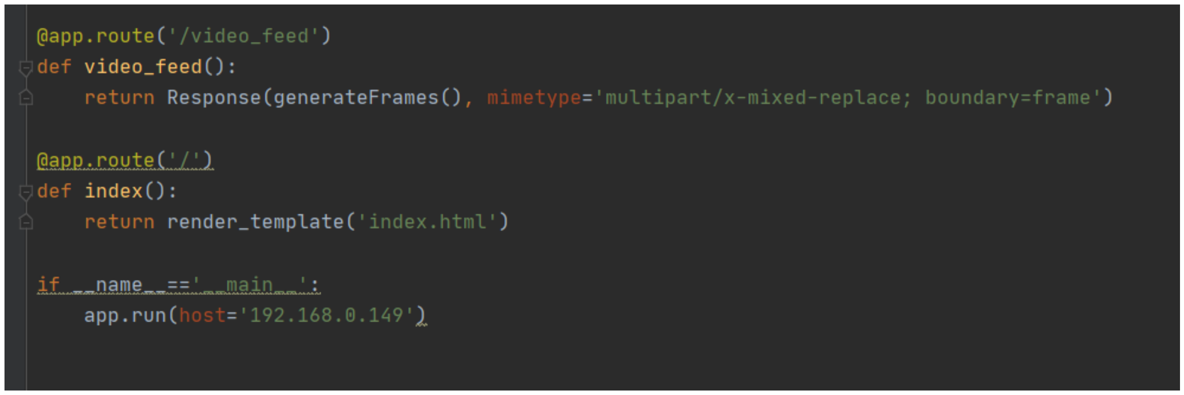
.



```html
<!doctype html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.mi
          integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crosso

    <title>Surveillance System</title>
</head>
<body>
<div class="container">
    <div class="row">
        <div class="col-lg-8  offset-lg-2">
            <h3 class="mt-5">Live Streaming</h3>
            <img src="{{ url_for('video_feed') }}" width="100%">
        </div>
    </div>
</div>
</body>
</html>
```

The Flask framework updates a simple html page. As it can be observed the file only contains the CSS elements for uploading the video, the title page, and a class that contains a source for the URL we will stream, with the id ‘video_feed’).

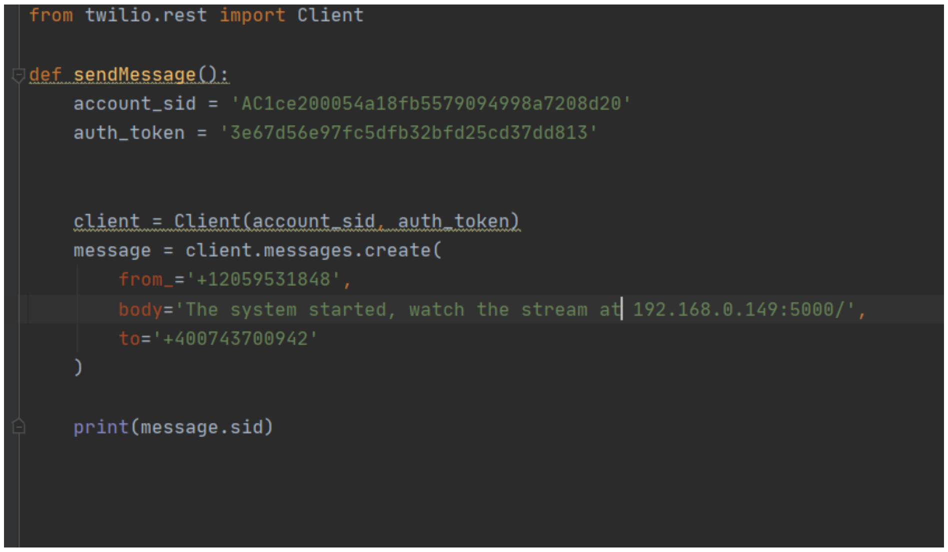
In the Python code we initialize the app using the command app = Flask(__name__) that will generate a web application.



```python
@app.route('/video_feed')
def video_feed():
    return Response(generateFrames(), mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/')
def index():
    return render_template('index.html')

if __name__=='__main__':
    app.run(host='192.168.0.149')
```

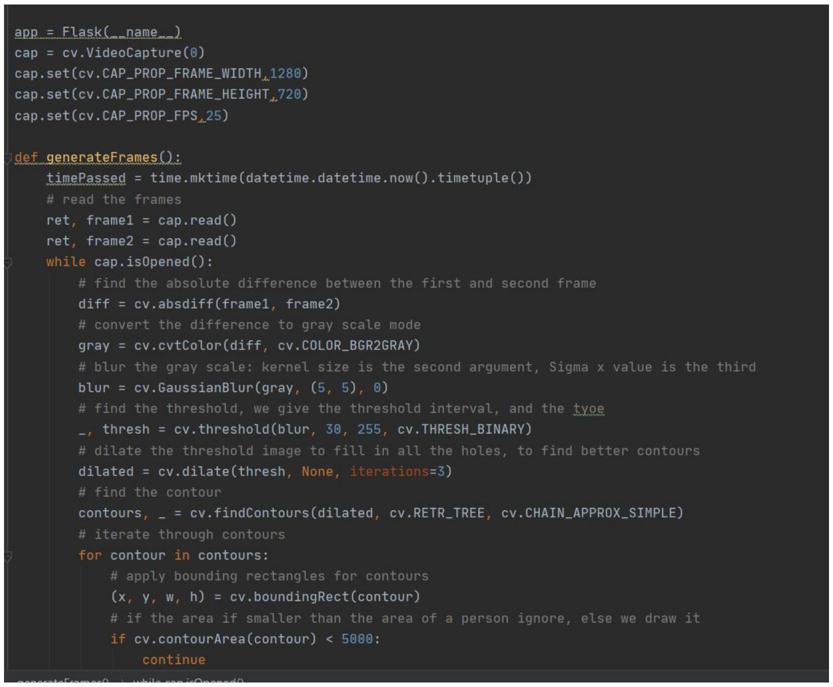
Using the function video_feed we update the source in the HTML code with the generated frames in the function generateFrames (we will discuss about it at the image processing part). The function sends a response to the URL with the video stream.

We render the index.html page in the function index(), and by specifying the IPv4 of the hosting device in the main we will host the application on this address, and it will be visible for all the persons in the network.

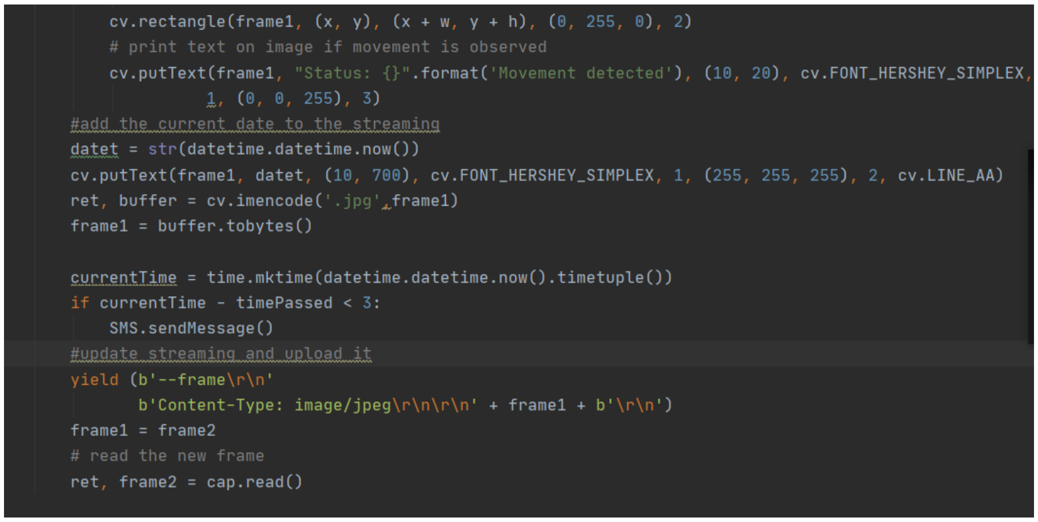


```python
from twilio.rest import Client

def sendMessage():
    account_sid = 'AC1ce200054a18fb5579094998a7208d20'
    auth_token = '3e67d56e97fc5dfb32bfd25cd37dd813'


    client = Client(account_sid, auth_token)
    message = client.messages.create(
        from_='+12059531848',
        body='The system started, watch the stream at 192.168.0.149:5000/',
        to='+400743700942'
    )

    print(message.sid)
```

The SMS is sent by using the private security tokens provided by Twilio, and specified in the variables account_sid and auth_token. With these we connect to the client(which is the account on the platform), and using the client we send the message from the phone number assigned to each account to the receiving person, along with the text. Message.sid is the code of the message and is printed in order to verify that the message has been sent.



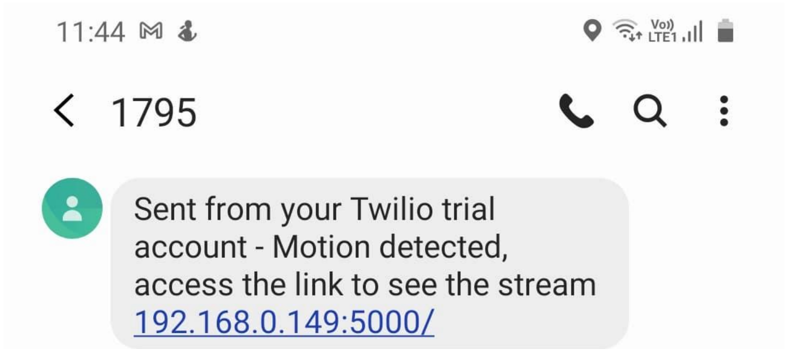
```python
app = Flask(__name__)
cap = cv.VideoCapture(0)
cap.set(cv.CAP_PROP_FRAME_WIDTH,1280)
cap.set(cv.CAP_PROP_FRAME_HEIGHT,720)
cap.set(cv.CAP_PROP_FPS,25)

def generateFrames():
    timePassed = time.mktime(datetime.datetime.now().timetuple())
    # read the frames
    ret, frame1 = cap.read()
    ret, frame2 = cap.read()
    while cap.isOpened():
        # find the absolute difference between the first and second frame
        diff = cv.absdiff(frame1, frame2)
        # convert the difference to gray scale mode
        gray = cv.cvtColor(diff, cv.COLOR_BGR2GRAY)
        # blur the gray scale: kernel size is the second argument, Sigma x value is the third
        blur = cv.GaussianBlur(gray, (5, 5), 0)
        # find the threshold, we give the threshold interval, and the tyoe
        _, thresh = cv.threshold(blur, 30, 255, cv.THRESH_BINARY)
        # dilate the threshold image to fill in all the holes, to find better contours
        dilated = cv.dilate(thresh, None, iterations=3)
        # find the contour
        contours, _ = cv.findContours(dilated, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
        # iterate through contours
        for contour in contours:
            # apply bounding rectangles for contours
            (x, y, w, h) = cv.boundingRect(contour)
            # if the area if smaller than the area of a person ignore, else we draw it
            if cv.contourArea(contour) < 5000:
                continue
```



```python
            cv.rectangle(frame1, (x, y), (x + w, y + h), (0, 255, 0), 2)
            # print text on image if movement is observed
            cv.putText(frame1, "Status: {}".format('Movement detected'), (10, 20), cv.FONT_HERSHEY_SIMPLEX,
                       1, (0, 0, 255), 3)
        #add the current date to the streaming
        datet = str(datetime.datetime.now())
        cv.putText(frame1, datet, (10, 700), cv.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv.LINE_AA)
        ret, buffer = cv.imencode('.jpg',frame1)
        frame1 = buffer.tobytes()

        currentTime = time.mktime(datetime.datetime.now().timetuple())
        if currentTime - timePassed < 3:
            SMS.sendMessage()
        #update streaming and upload it
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame1 + b'\r\n')
        frame1 = frame2
        # read the new frame
        ret, frame2 = cap.read()
```

The function to generate the stream implements the motion detection part and streams it to the server. The function has a while loop that runs as long as the programme can take data from the web camera. We also add the current time on the stream. Each frame is converted into a JPG format and sent continuously to the server and updates with the speed of 20 FPS.

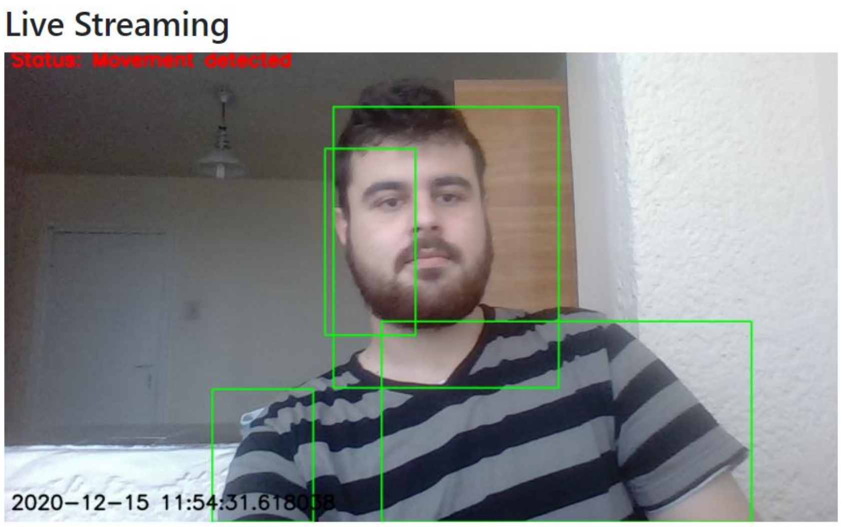
6. Testing and validation



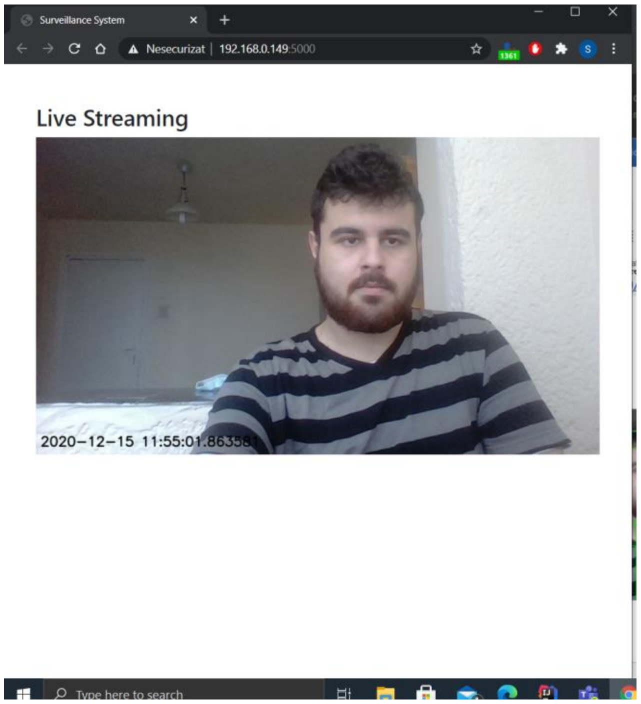
11:44

1795

Sent from your Twilio trial account - Motion detected, access the link to see the stream 192.168.0.149:5000/



Live Streaming



7. Conclusion

The goal of this project was to design and implement a Video surveillance system hosted on a local server, that detects the motion and sends a warning to the user, and a notification of when the system started.

I liked this project because it was the first time working with image processing tools, and learned more about it and how to host an application on a local server. It was interesting finding the solutions from the papers I have read and trying to implement them in my own version. I learned to work with automated sms and Twilio API.

8. Bibliography

[1] https://docs.opencv.org/4.5.0/d1/dc5/tutorial_background_subtraction.html

[2]https://towardsdatascience.com/video-streaming-in-web-browsers-with-opencv-flask-93a38846fe00

[3] https://www.twilio.com/docs/tutorials?filter-product=SMS&filter-language=Python

[4] https://pypi.org/project/Flask/