2023-2024

Camera detection system

Mariș Radu-Ioan

Technical University of Cluj-Napoca

Camera monitoring system: motion detection

Table of Contents

1. Introduction

1.1 Context

The goal of this project is to design a camera system that is capable of supervising a location and via Internet alert the user if any unexpected situation might occur, such as in a room that supposedly is empty an object moving is detected.

This monitoring system can be used by people who need a simple way to keep watch over their house, garden or other type of propriety that they might own, as well as a baby monitoring device.

1.2 Specifications

The system employs OpenCV and Python to capture camera input, utilizing a physical camera like the one integrated into a laptop. Django is chosen as the web framework to host the website. The integration of OpenCV and Python facilitates camera input detection, while Django is used for web hosting. The overall system combines these technologies to gather camera input and process the information.

1.3 Objectives

Design and implement a way to detect the camera input, detect the movement using a series of matrixes and draw the perimeter of a rectangle over the moving object for better visualization.

When a moving object is detected, the website will indicate this with a red rectangle covering the perimeter of the entire camera input. So, when something is moving, the website will show both the rectangle around the object that should not be there and the red rectangle around the entire camera input.

2. Bibliographic study

OpenCV

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 the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

Python

Python is a high-level, versatile programming language known for its readability and simplicity. Created with a focus on code readability and ease of use, Python supports multiple programming paradigms, making it suitable for various applications. Its extensive standard library and vibrant community contribute to its popularity, enabling developers to create diverse software quickly and efficiently, from web applications and data analysis scripts to artificial intelligence and machine learning projects. Python's clean syntax and dynamic typing make it accessible for beginners while offering advanced features for experienced programmers.

Django

Django is a high-level web framework for building dynamic and scalable web applications using Python. It follows the model-view-controller (MVC) architectural pattern, emphasizing code reusability and rapid development. Django provides a robust set of tools and features, including an Object-Relational Mapping (ORM) system for database interactions, a templating engine for efficient HTML rendering, and a built-in administrative interface for managing web application content. With a focus on "batteries-included" philosophy, Django simplifies common web development tasks, allowing developers to focus on building features rather than dealing with low-level details. Its versatility and adherence to best practices make it a popular choice for developing secure and maintainable web applications.

Python & Django

Python and Django integration combines the powerful capabilities of the Python programming language with the robust web development framework provided by Django. This integration allows developers to seamlessly leverage Python's versatility for tasks such as camera input processing using OpenCV, while Django handles the web application's backend structure. The synergy between Python and Django streamlines the development process, providing an efficient and scalable solution for hosting websites with sophisticated functionalities, such as camera input detection and processing.

3. Analysis

To effectively manage the project workload throughout the semester, I have established a table to track the implementation schedule for various features. This table outlines the specific features that will be addressed in each weekly session, ensuring a systematic and organized approach to project development.

Week planification for the implementations:

Week 1-2:
I am conducting a search for bibliographic studies and other relevant sources to gather information and insights that will contribute to the progress of the project.

Week 3-4:
I have installed a computer vision (CV) library and I tried to get myself familiar with its functionalities, exploring methods to integrate it seamlessly with Python for effective usage.

Week 5-6:
I make the functionality off the application such that the laptop camera is detecting motion using Python.

Week 7-8:
I develop a “hello word” application to get familiar with the Django environment.

Week 9-10:
I combine the functionality of the application (code in Python) with the Django environment, integrating them seamlessly.

Week 11-12:
I polish the project and the documentation according to the feedback adding minor improvements.

4. Implementation

The way I went about detecting motion is similar to a surveillance camera. My camera is placed in a static position and takes a picture of the environment. Then, it will compare each frame of the live video with the picture used as reference, and if anything is different it will signal it with a rectangle.

First transformation to the image is a grayscale transformation in order to detect the light intensity of the image. In order to detect only relevant movement I used a gaussian blur to transform the first image.

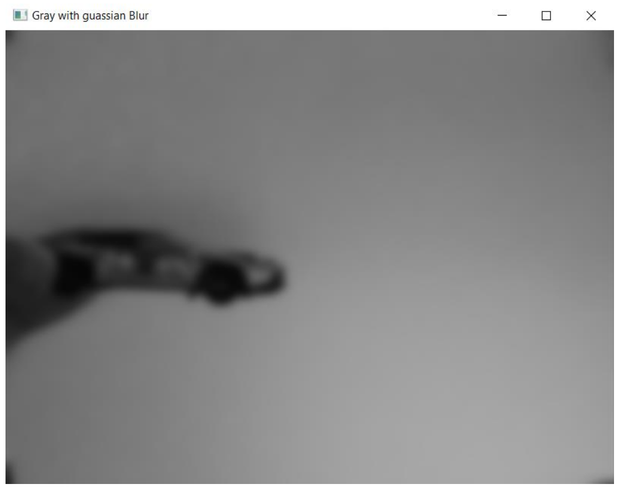


Fig. Gray with gaussian blur

After that, I generate the difference frame. This is done by comparing the initial image with the current frame from the camera. If there is a difference in light values.

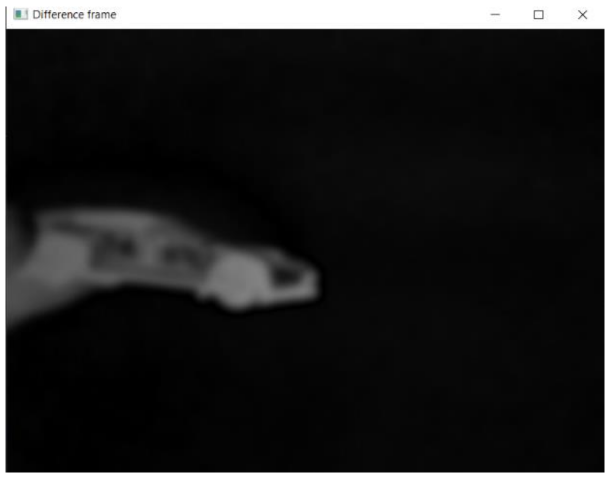


Fig. Difference frame

The next transformation is the threshold generation. For this, we set a threshold and if there is a light value above that value, then it will be converted to white, thus having only 2 values to work with: black and white.

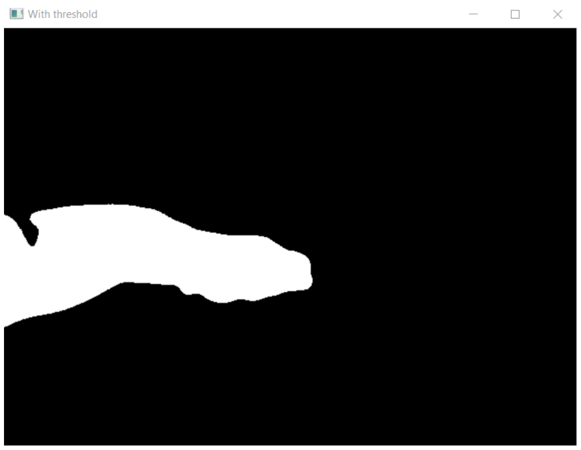


Fig. Threshold

The final transformation is applied in the current frame of the camera input. By using the threshold and OpenCV it detects the perimeter of all the white objects and generates bounding coordinates for that object. Then I draw rectangles at the coordinates from the threshold on the current frame of the input and get the final image that is shown to the user.

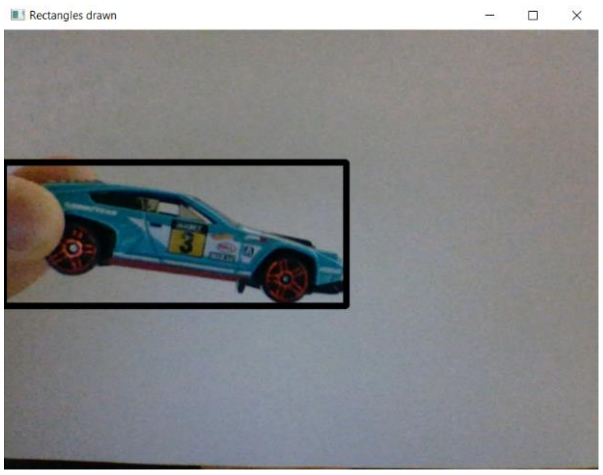


Fig. Final output

5. Testing and validation

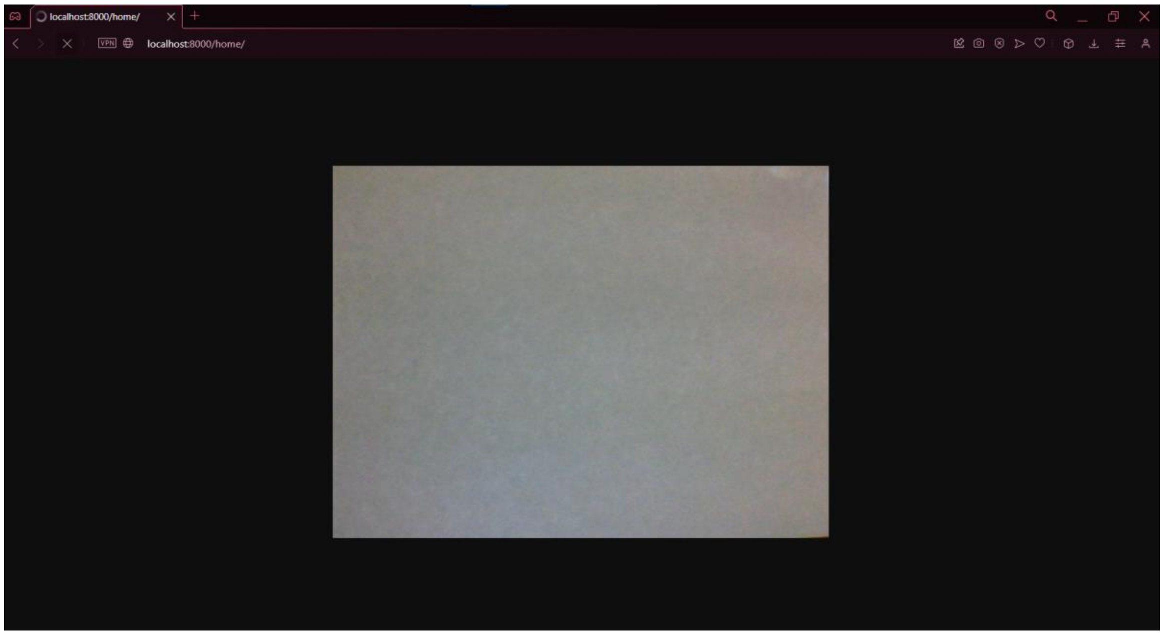
In order to run the application, you need to a command in terminal. This command will execute the “manage.py” file in python and will run a server. The specific command is:

“python manage.py runserver”

This will run a Django webserver at the following web address:

“localhost/8000/home”

It will display your camera input in the center of the page and it will start to detect any change since the opening of the page. If any change is detected, a rectangle overlay will be be displayed over the place where motion is detected. Also, a red rectangle overlay will be encapsulating the entire camera input like in the images bellow:



<u>Fig. Initial state</u>

A piece of paper is put in front of the camera for demonstration purposes. At the beginning, only the sheet of paper is show and no movement, so only the camera input is displayed. As long as the website is open, the camera will continue to search for any movement.

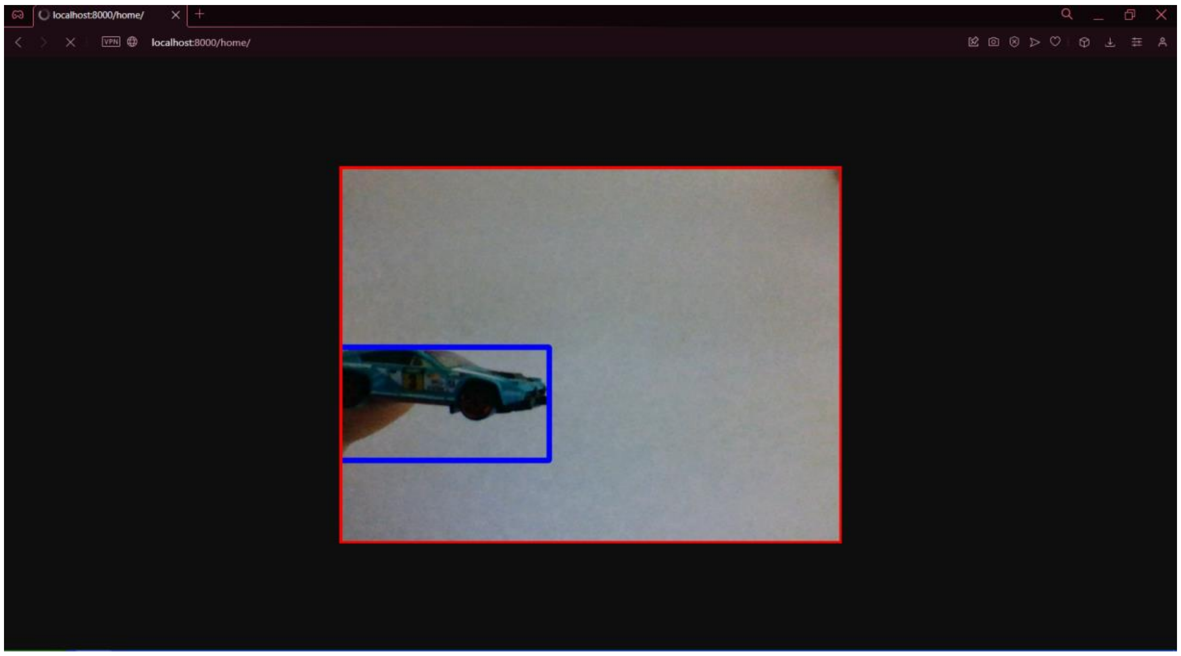


Fig. Movement detected

The camera detected movement and with the blue rectangle overlay the object that is detected is contoured and a red rectangle will appear around the whole image. In this example, the car is detected and will appear on the display with the blue rectangle surrounding it, indicating it’s position.

6. Conclusions

The Camera Detection System project successfully achieved its objectives of designing and implementing a camera monitoring system with motion detection capabilities. By using OpenCV, Python, and Django, the system captures camera input, processes motion detection, and presents the information through a web interface. The step-by-step implementation plan ensured a systematic development approach, from the exploration of relevant technologies such as Python and Django.

The motion detection algorithm, inspired by surveillance camera principles, identifies changes in the camera input and highlights them with a blue rectangle, providing a visual representation of the detected movement. The testing and validation phase demonstrated the system's functionality, showcasing its ability to promptly identify and display movement, encapsulating the entire camera input with a red rectangle for enhanced visualization.

For future developments the user-interface can get better. A dedicated refresh button is planned for the next iteration of the project. At current time, for a refresh you need to refresh the whole page.

Also, I would like to implement sound or other factors that could alert users that something is moving. At the moment only visual stimuli are used and I believe sounds could improve the awareness of the motion.

7. Bibliography

1. https://en.wikipedia.org/wiki/OpenCV
2. https://masteringdjango.com/django-tutorials/mastering-django-structure/
3. https://www.geeksforgeeks.org/erosion-dilation-images-using-opencv-python/
4. https://docs.djangoproject.com/en/5.0/
5. https://docs.python.org/3/
6. https://docs.opencv.org/4.x/