MOTION DETECTION SYSTEM

PROJECT REPORT

By

PAWAN PRIYATHAM (RA2211032010048)
NAMISH. S (RA2211032010062)
BHUMIKA SHARMA (RA2211032010066)

Under the guidance of

Dr. R. LAKSHMINARAYANAN

(Assistant Professor, Department of Networking and Communications, School of Computing)

In partial fulfilment for the Course

of

21CSS201T – ADVANCED PROGRAMMING PRACTICES

in Networking and Communications with

Specialization in Internet of Things

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this B. Tech project report titled **“MOTION DETECTION ALARM”** is the bonafide work of **Pawan Priyatham (RA2211032010048), Namish. S (RA2211032010062) and Bhumika Sharma (RA2211032010066)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

DR. R. LAKSHMINARAYANAN	**DR. ANNAPURANI. K**
SUPERVISOR	**HEAD OF THE DEPARTMENT**
Assistant Professor	Professor
Networking and Communications	Networking and Communications

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

Annexure II

Department of Networking and Communications

SRM Institute of Science & Technology

OWN WORK DECLARATION

Degree/ Course: B. Tech/Computer Science Engineering with specialization in Internet of Things

Student Name: Pawan Priyatham/ Namish. S/ Bhumika Sharma

Registration Number: RA2211032010048/ RA2211032010062/ RA2211032010066

Title of Work: Motion Detection Alarm

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines. We confirm that all the work contained in this assessment is our own except where indicated, and that We have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g., fellow students, technicians, statisticians, external sources).

- Compiled with any other plagiarism criteria specified in the Course handbook / University website.

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my own work, except were indicated by referring, and that I have followed the good academic practices noted above.

RA2211032010048	RA2211032010062	RA2211032010066
08/11/23	08/11/23	08/11/23

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude **to Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr T. V. Gopal**, for his invaluable support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr K. Annapurani Panaiyappan**, Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Panel Head, **Dr. Kayalvizhi Jayavel**, Assistant Professor, and program coordinators **Dr. M. B Mukesh Krishnan**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Ms. Praveena Akki**, Assistant Professor, Networking & Communications, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. R. Lakshminarayanan**, Assistant Professor, Networking & Communications, SRM IST, for providing me with an opportunity to pursue my project under his mentorship. He provided me with the freedom and support to explore the research topics of my interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Networking and Communications Department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Pawan Priyatham [RA2211032010048]

Namish. S [RA2211032010062]

Bhumika Sharma [RA2211032010066]

ABSTRACT

A motion detection alarm system using Python and OpenCV is a software application designed to monitor live video streams from cameras or video files, detect any significant motion within the frames, and trigger alarms or notifications when motion is detected. The system works by capturing consecutive frames, comparing them to identify changes in pixel values, and analyzing these changes to determine if they constitute motion. When motion is detected, the system can activate various responses, such as sounding an alarm, sending email or SMS notifications, or saving video clips of the detected motion. This technology has a wide range of practical applications, including home security, surveillance, and automation, allowing users to enhance the security of their premises and be alerted to potential intrusions or unusual activities in real-time. The Python and OpenCV combination provide a powerful and flexible platform for implementing such motion detection alarm systems, offering the ability to customize and adapt the system to meet specific monitoring and security needs.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 MOTIVATION

Creating a motion detection alarm system with Python and OpenCV serves as an exciting and practical project that addresses real-world security concerns. This project's motivation lies in its potential to enhance home security, surveillance, and automation. By harnessing the power of computer vision, you can develop a system that can detect any unexpected movement in a designated area, instantly triggering alarms, or notifications. Whether safeguarding your home, office, or personal space, this project not only offers a sense of accomplishment but also empowers you with a valuable skill set in computer vision and automation, which can be applied to various other applications. Furthermore, it demonstrates the synergy between cutting-edge technology and everyday security, enabling you to contribute to the ongoing evolution of smart, responsive, and more secure environments.

1.2 OBJECTIVE

The objective of creating a motion detection alarm using Python and OpenCV is to develop a system that can monitor a live video feed from a camera and trigger an alert when it detects any motion. This project involves capturing video frames, processing them with computer vision techniques, and comparing consecutive frames to identify changes in the scene. When significant motion is detected, an alarm or notification is activated, alerting the user to the potential intrusion or movement. This system can have various applications, including home security, surveillance, and automation. By implementing this project, one can gain hands-on experience in image processing, video analysis, and event-driven programming, while enhancing security and monitoring capabilities.

1.3 PROBLEM STATEMENT

Create a motion detection alarm system using Python and OpenCV to enhance security and surveillance in various environments. The system should continuously analyse live video feeds from a camera, identifying and tracking any moving objects within the frame. When motion is detected, the system should trigger an alarm, such as sending email notifications, sounding an audible alert, or activating additional security measures. Users should be able to configure sensitivity levels and define specific regions of interest within the camera's field of view. The primary goal is to develop a cost-effective, open-source solution that can be easily implemented in homes, businesses, or other locations to provide real-time monitoring and alerts for potential security breaches or unusual activity, ultimately increasing safety and peace of mind.

1.4 CHALLENGES

Creating a motion detection alarm system with Python and OpenCV presents several challenges. First, designing an accurate motion detection algorithm that can distinguish between true motion events and false positives due to lighting changes or camera noise is a complex task. Tuning sensitivity parameters and setting appropriate thresholds is essential. Second, optimizing the system for real-time performance can be demanding, especially on resource-constrained devices. Efficiently processing video frames, minimizing latency, and handling various camera resolutions are critical aspects. Additionally, handling different lighting conditions and adapting to environmental changes requires adaptive algorithms. Lastly, implementing a robust notification mechanism, such as email alerts or SMS notifications, to alert users when motion is detected, poses another challenge. Overall, building a reliable motion detection alarm system with Python and OpenCV involves addressing these technical and algorithmic hurdles to ensure its effectiveness and practicality

2. LITERATURE SURVEY

S.no	Paper Title	Author	Year	Publisher	Keywords
1.	An Intelligent Motion Detection Using OpenCV	Dr.Yusuf Perwej, Nikhat Akhtar, Mrs Versha Verma, Shivam Chaturvedi	2022	International Journal of Scientific Research in Science, Engineering and Technology	Delta Frames, OpenCV
2.	A motion detection system in python and OpenCV	Suraiya Praveen, Javeria Shah	2021	ResearchGate	Eliminate noise, moving target
3.	Movement detection using OpenCV	Ankita Rameshwar Mahajan, Vinod Agrawal	2022	International Research Journal of Modernization in Engineering Technology and Science	Counts the movement of the person.

3. REQUIREMENTS ANALYSIS

3.1 Software Requirements

From the given scenario, we draw the following requirements:

1. Python: Ensure that Python is installed on your computer. You can download it from Python's official website.
2. OpenCV: Install OpenCV, a popular computer vision library, using pip.
3. Functional Requirements:
    - Video Capture: The system should be able to capture video frames from the camera in real-time.
    - Motion Detection: Implement a motion detection algorithm to analyze consecutive frames and identify regions where motion occurs.
    - Thresholding: Apply thresholding to differentiate between motion and non-motion areas.
    - Alert System: The system should have an alert mechanism.
    - Alarm Sound: Play a sound when motion is detected.
    - User Interface: Create a simple user interface (GUI or command-line) to start/stop the motion detection system and configure settings.

4. Non-Functional Requirements:
    - Performance: The system should run efficiently and be able to process video frames in real-time.
    - Reliability: It should be reliable, minimizing false alarms while accurately detecting actual motion events.
    - Security: If needed, ensure the system is secure, especially if it involves sending alerts or storing data.
    - Ease of Use: Make the system user-friendly, with clear instructions and error handling.

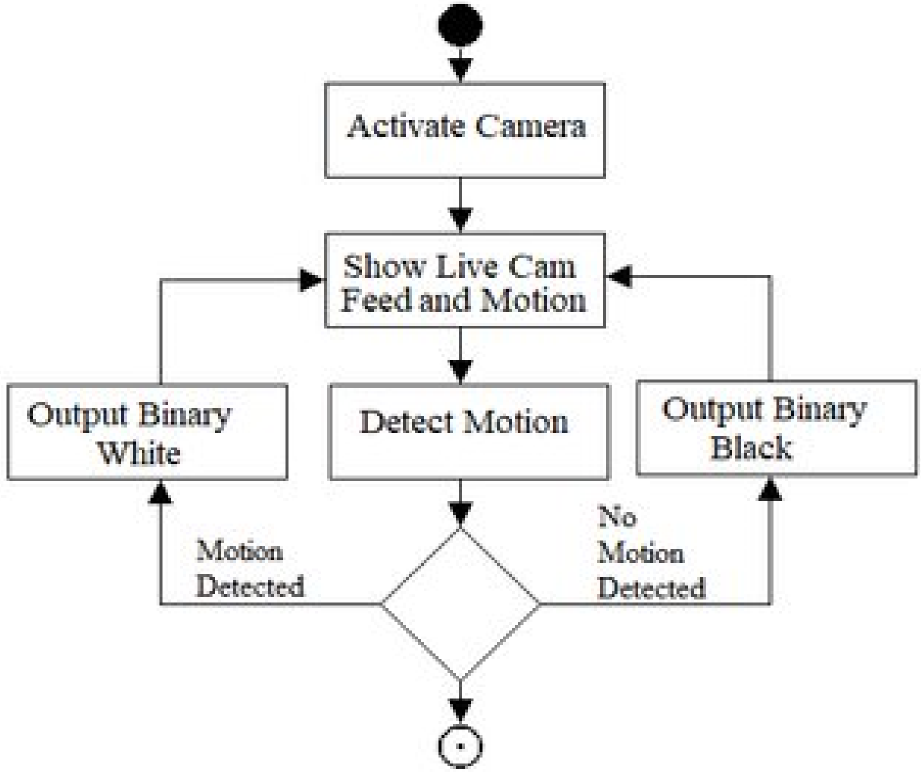
3.2 Hardware Requirement

1. Performance: The system should run efficiently and be able to process video frames in real-time.
2. Reliability: It should be reliable, minimizing false alarms while accurately detecting actual motion events.
3. Security: If needed, ensure the system is secure, especially if it involves sending alerts or storing data.
4. Ease of Use: Make the system user-friendly, with clear instructions and error handling.

4. ARCHITECTURE AND DESIGN

4.1 Architecture

The architecture is as follows:



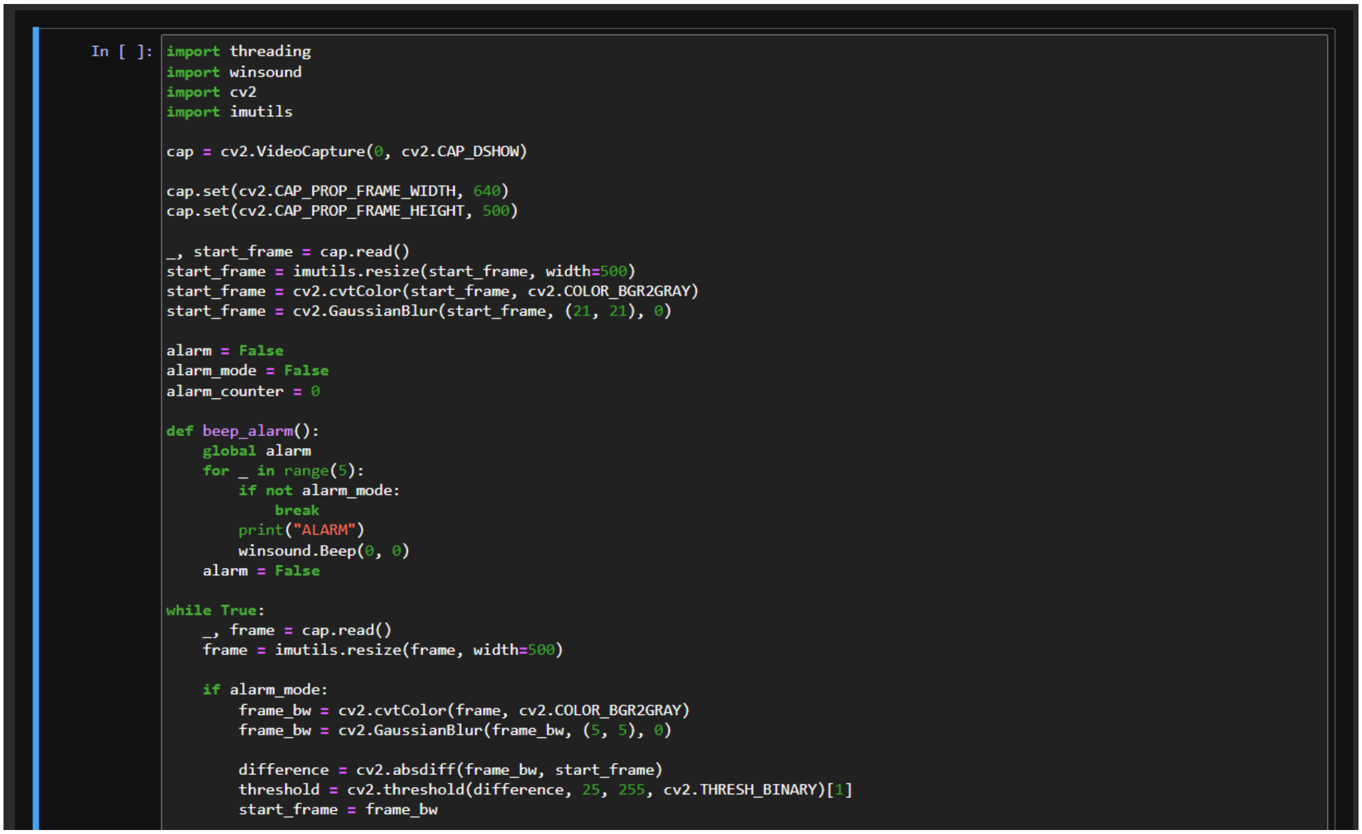
The architecture consists of the following management modules:

- Threading Module
- Win sound Module
- Cv2 (OpenCV) Module
- Imutils Module

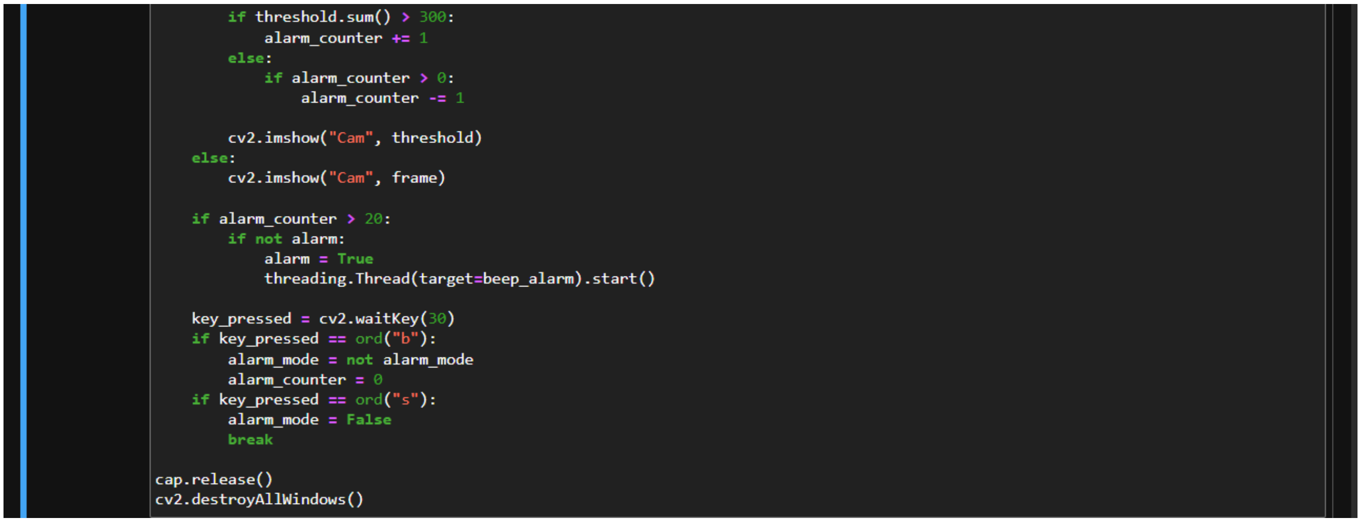
These modules are interconnected with each other.

5. IMPLEMENTATION

PROGRAM:

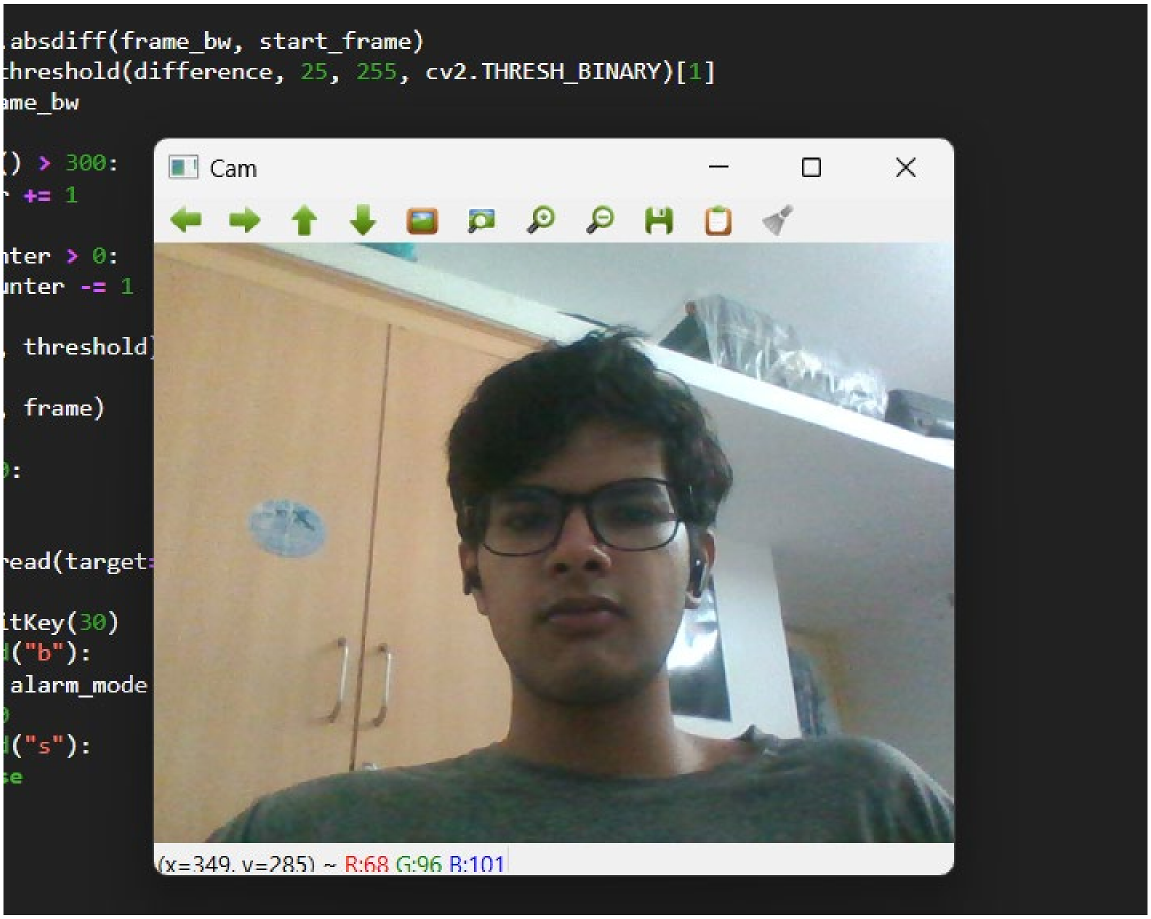


```
In [ ]: import threading
        import winsound
        import cv2
        import imutils

        cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

        cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
        cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 500)

        _, start_frame = cap.read()
        start_frame = imutils.resize(start_frame, width=500)
        start_frame = cv2.cvtColor(start_frame, cv2.COLOR_BGR2GRAY)
        start_frame = cv2.GaussianBlur(start_frame, (21, 21), 0)

        alarm = False
        alarm_mode = False
        alarm_counter = 0

        def beep_alarm():
            global alarm
            for _ in range(5):
                if not alarm_mode:
                    break
                print("ALARM")
                winsound.Beep(0, 0)
            alarm = False

        while True:
            _, frame = cap.read()
            frame = imutils.resize(frame, width=500)

            if alarm_mode:
                frame_bw = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                frame_bw = cv2.GaussianBlur(frame_bw, (5, 5), 0)

                difference = cv2.absdiff(frame_bw, start_frame)
                threshold = cv2.threshold(difference, 25, 255, cv2.THRESH_BINARY)[1]
                start_frame = frame_bw
```

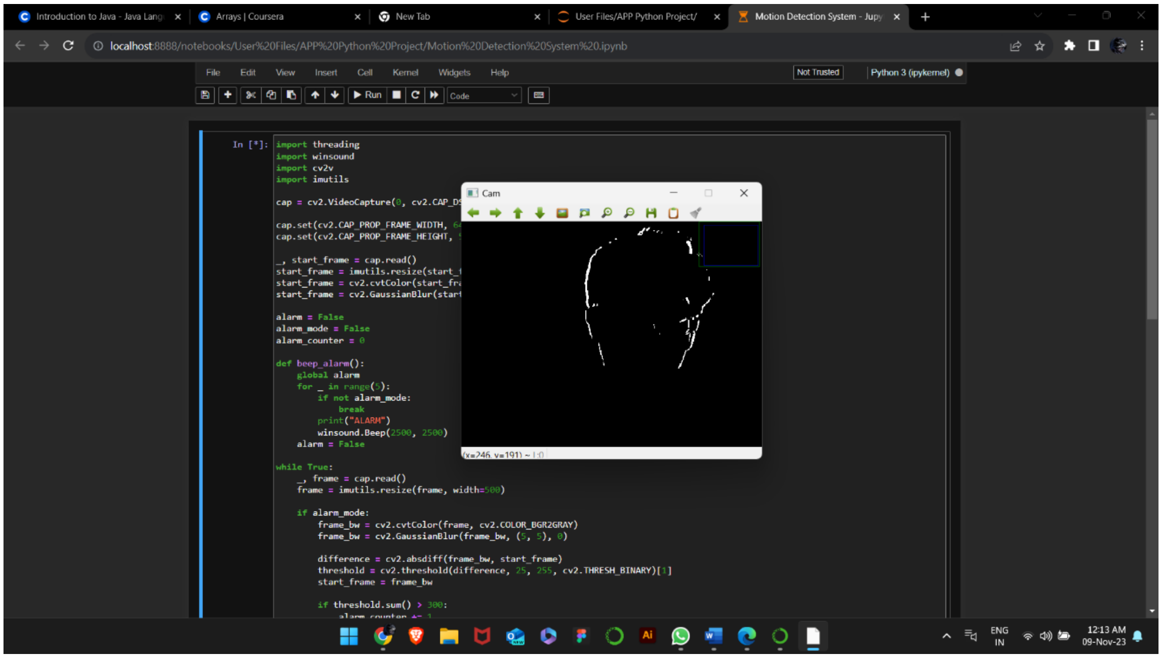


```
        if threshold.sum() > 300:
            alarm_counter += 1
        else:
            if alarm_counter > 0:
                alarm_counter -= 1

        cv2.imshow("Cam", threshold)
    else:
        cv2.imshow("Cam", frame)

    if alarm_counter > 20:
        if not alarm:
            alarm = True
            threading.Thread(target=beep_alarm).start()

    key_pressed = cv2.waitKey(30)
    if key_pressed == ord("b"):
        alarm_mode = not alarm_mode
        alarm_counter = 0
    if key_pressed == ord("s"):
        alarm_mode = False
        break

cap.release()
cv2.destroyAllWindows()
```

OUTPUT:

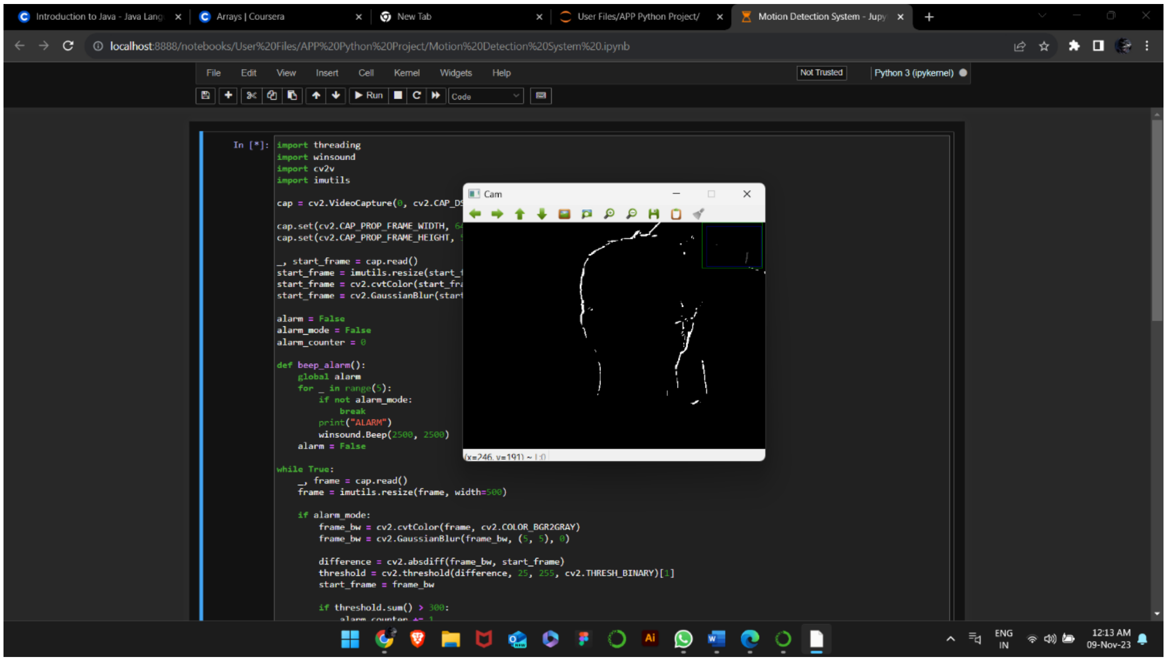


```
.absdiff(frame_bw, start_frame)
hreshold(difference, 25, 255, cv2.THRESH_BINARY)[1]
ame_bw

) > 300:
 += 1

nter > 0:
unter -= 1

 threshold)

 frame)

):


ead(target

tKey(30)
("b"):
 alarm_mode
)
("s"):
e
```



Introduction to Java - Java Lang
Arrays | Coursera
New Tab
User Files/APP Python Project/
Motion Detection System - Jupy
localhost:8888/notebooks/User%20Files/APP%20Python%20Project/Motion%20Detection%20System%20.ipynb
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Cam
(x=246, y=191) ~ L:0
ENG
IN
12:13 AM
09-Nov-23

```
In [*]: import threading
        import winsound
        import cv2v
        import imutils

        cap = cv2.VideoCapture(0, cv2.CAP_DS

        cap.set(cv2.CAP_PROP_FRAME_WIDTH, 64
        cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 

        _, start_frame = cap.read()
        start_frame = imutils.resize(start_
        start_frame = cv2.cvtColor(start_fra
        start_frame = cv2.GaussianBlur(start

        alarm = False
        alarm_mode = False
        alarm_counter = 0

        def beep_alarm():
            global alarm
            for _ in range(5):
                if not alarm_mode:
                    break
                print("ALARM")
                winsound.Beep(2500, 2500)
            alarm = False

        while True:
            _, frame = cap.read()
            frame = imutils.resize(frame, width=500)

            if alarm_mode:
                frame_bw = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                frame_bw = cv2.GaussianBlur(frame_bw, (5, 5), 0)

                difference = cv2.absdiff(frame_bw, start_frame)
                threshold = cv2.threshold(difference, 25, 255, cv2.THRESH_BINARY)[1]
                start_frame = frame_bw

                if threshold.sum() > 300:
                    alarm_counter += 1
```


Introduction to Java - Java Lang
Arrays | Coursera
New Tab
User Files/APP Python Project/
Motion Detection System - Jupy
localhost:8888/notebooks/User%20Files/APP%20Python%20Project/Motion%20Detection%20System%20.ipynb
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Cam
(x=246, y=191) ~ L:0
ENG
IN
12:13 AM
09-Nov-23

```
In [*]: import threading
        import winsound
        import cv2v
        import imutils

        cap = cv2.VideoCapture(0, cv2.CAP_DS

        cap.set(cv2.CAP_PROP_FRAME_WIDTH, 64
        cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 

        _, start_frame = cap.read()
        start_frame = imutils.resize(start_f
        start_frame = cv2.cvtColor(start_fra
        start_frame = cv2.GaussianBlur(start

        alarm = False
        alarm_mode = False
        alarm_counter = 0

        def beep_alarm():
            global alarm
            for _ in range(5):
                if not alarm_mode:
                    break
                print("ALARM")
                winsound.Beep(2500, 2500)
            alarm = False

        while True:
            _, frame = cap.read()
            frame = imutils.resize(frame, width=500)

            if alarm_mode:
                frame_bw = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                frame_bw = cv2.GaussianBlur(frame_bw, (5, 5), 0)

                difference = cv2.absdiff(frame_bw, start_frame)
                threshold = cv2.threshold(difference, 25, 255, cv2.THRESH_BINARY)[1]
                start_frame = frame_bw

                if threshold.sum() > 300:
                    alarm_counter += 1
```

6. CONCLUSION

In conclusion, developing a motion detection alarm system using Python and OpenCV offers a powerful and versatile solution for enhancing security and monitoring in various environments. By harnessing the capabilities of computer vision, this system can automatically detect and alert users to any unexpected motion within a specified area, making it valuable for both residential and commercial applications. Python and OpenCV's ease of use and extensive libraries allow for the creation of a robust alarm system that can be customized to meet specific requirements.

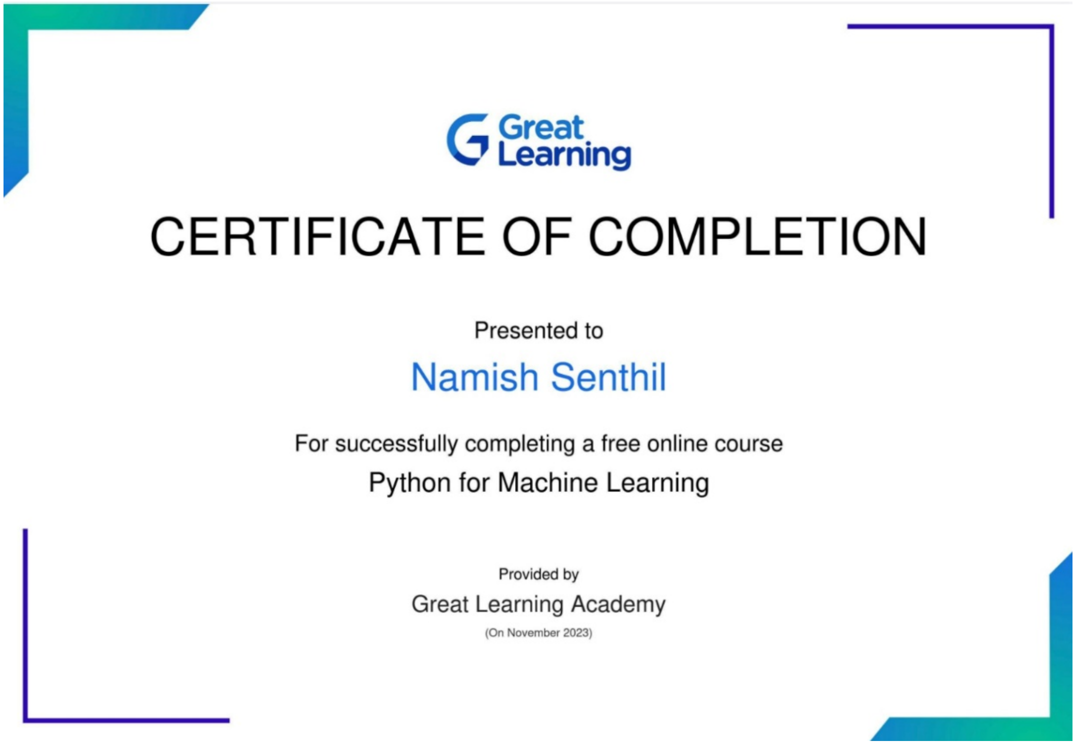
The core functionality of the motion detection alarm relies on continuously capturing video frames from a camera source, comparing consecutive frames to identify regions of change, and triggering alerts when motion is detected. The system's adaptability is further bolstered by the inclusion of features such as thresholding, user-defined regions of interest, and configurable alert options, including sound alarms and visual notifications. Logging capabilities also enable the system to record timestamps and relevant information about detected motion events, which can be useful for security or surveillance purposes.

With its capacity to operate efficiently and accurately in real-time, the motion detection alarm system showcases the strength of Python and OpenCV in computer vision applications. By adhering to privacy and legal considerations and providing clear documentation, this system can serve as a valuable tool for enhancing security, automation, and monitoring, making it an asset for anyone seeking to implement an intelligent motion detection solution.

7. REFERENCES

- https://www.researchgate.net/publication/359368941_An_Intelligent_Motion_Detection_Using_OpenCV
- https://www.researchgate.net/publication/350538660_A_Motion_Detection_System_in_Python_and_Opencv
- https://www.irjmets.com/uploadedfiles/paper/issue_5_may_2022/22939/final/fin_irjmets1652552307.pdf
- Sezgin M, Sankur B, ―Survey over Image Thresholding Techniques and Quantitative Performance
- Evaluation‖. Journal of Electronic Imaging, 13: 146-165, (2004).
- N. Ramesh, J. H. Yoo, and I. K. Sethi, ―Thresholding based on histogram approximation‖, IEEE Proc.
- Vision Image Signal Process. 142(5), 271–279, (1995).
- T. W. Ridler and S. Calvard,‖ Picture Thresholding Using an Iterative Selection Method‖, IEEE
- Transactions On Systems, Man, And Cybernetics, Vol. sMC-8, NO. 8, AUGUST (1978).

8. PROOF OF CERTIFICATION



Great Learning

CERTIFICATE OF COMPLETION

Presented to

Namish Senthil

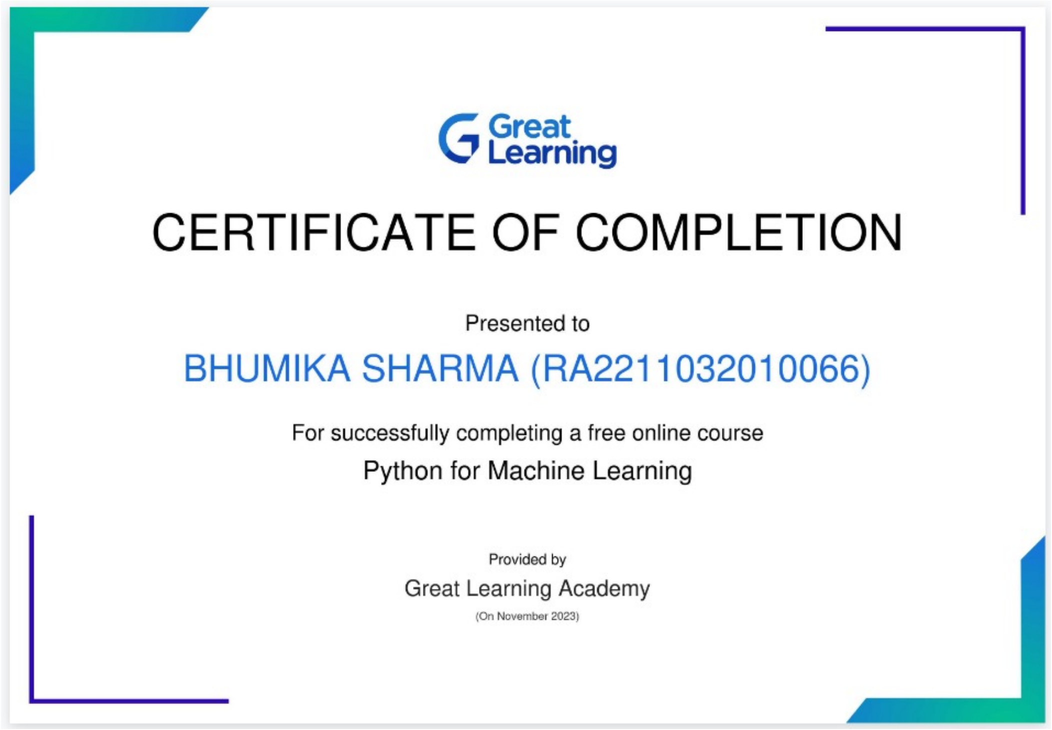
For successfully completing a free online course

Python for Machine Learning

Provided by

Great Learning Academy

(On November 2023)



Great Learning

CERTIFICATE OF COMPLETION

Presented to

BHUMIKA SHARMA (RA2211032010066)

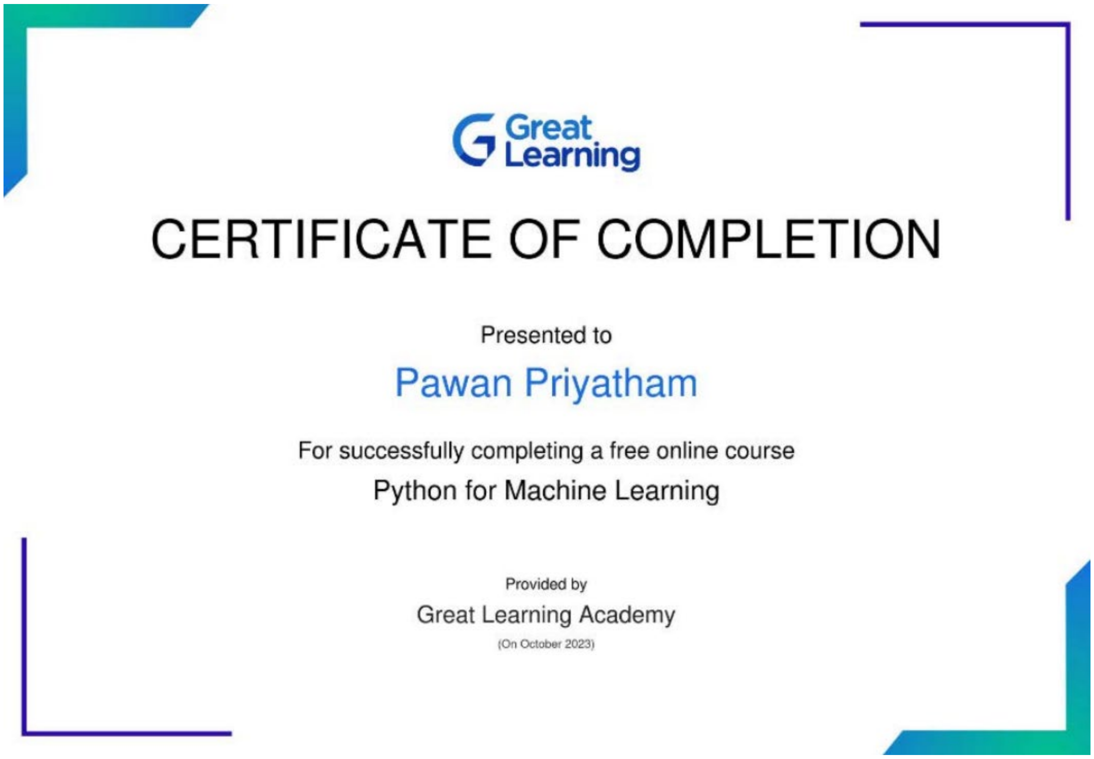
For successfully completing a free online course

Python for Machine Learning

Provided by

Great Learning Academy

(On November 2023)



Great Learning

CERTIFICATE OF COMPLETION

Presented to

Pawan Priyatham

For successfully completing a free online course

Python for Machine Learning

Provided by

Great Learning Academy

(On October 2023)