Motion Detection using Image Processing

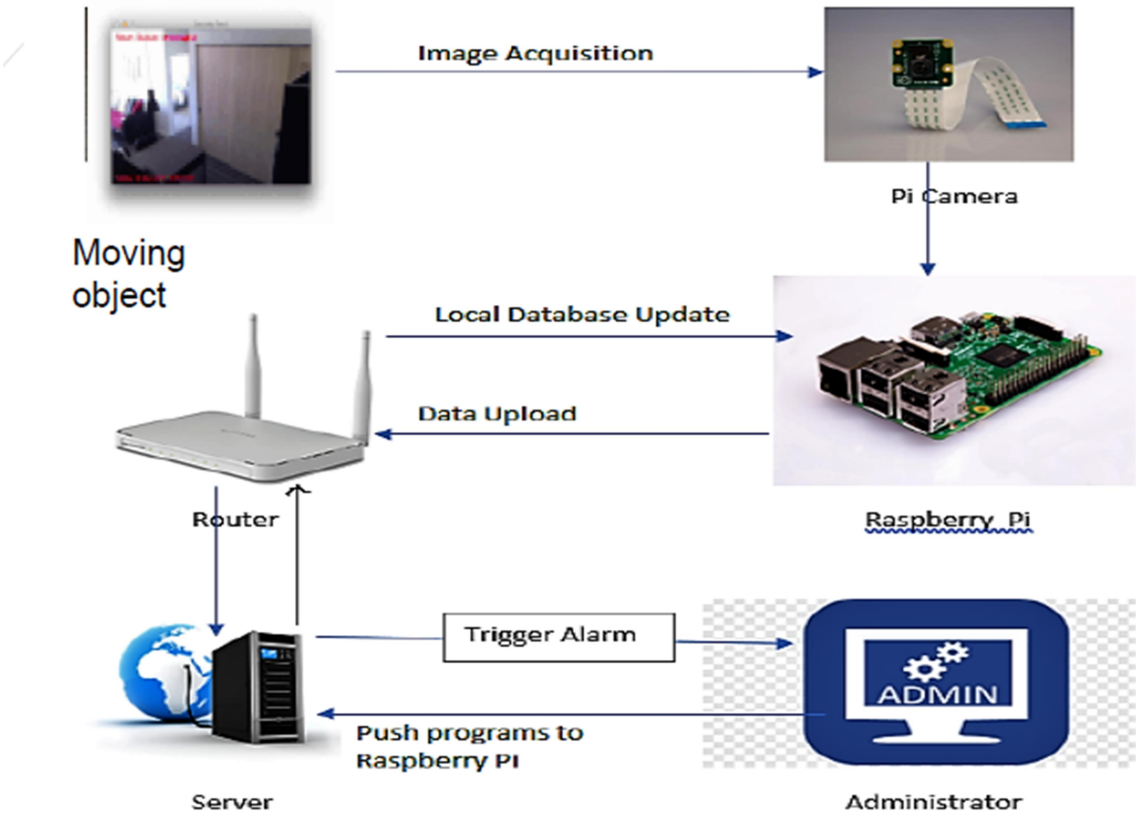


Table of Contents

List of Figures

Chapter 1 Introduction

1.1 Need of the system / Motivation

There is the number of premises such as security server room, several government buildings, banks, etc., which needs security in smart cities. They need to be monitored against intruders 24*7 or for a particular period. This module has been developed as a part of the Smart City Security Surveillance System to keep an eye on the unauthenticated movement.

The system is implemented such that video acquisition is made through the Raspberry PI camera, and the live stream is processed continuously. People's motion is being observed, and depending upon the type of motion, the intruder will be identified. When an intruder is identified, this information is sent to the concerned security personnel via mail, and an alert mechanism is triggered, which is an alarm in our case. Also, snapshots and the time is recorded and will be saved in the database for the future.

1.2 Scope

The system could be implemented on building's entrances, offices, banks, vaults, and other places where intruder/defaulter is allowed.

1.3 Overview of existing system

The system uses a high-resolution camera for video acquisition, and live stream is processed over Raspberry Pi. The live stream is converting it into

frames and performing operations for the detection of motion. The motion has been detected, and it has been displayed on the screen with a boundary box corresponding to any object's movement. It is also hosting the live stream and saving the images with a time stamp.

Chapter 2 Proposed System

2.1 MOTION DETECTION PROCESS

In case of thefts and unauthorized gate entries, organizations that want security can use this motion detection model for motion detection. This model will detect any motion, such as living object motion and non-living object motion. It provides motion-detected frames with time stamping to the organization. Motion detected frames could also be live-streamed.

1. The first frame of our video file will contain no motion and just background — therefore, we can model our video stream's background using only the first frame of the video.

2. Now we can start processing our frame and preparing it for motion analysis. We will first resize it down to have a width of 500 pixels — there is no need to process the large, raw images straight from the video stream. We will also convert the image to grayscale since color has no bearing on our motion detection algorithm. Finally, we will apply Gaussian blurring to smooth our images.

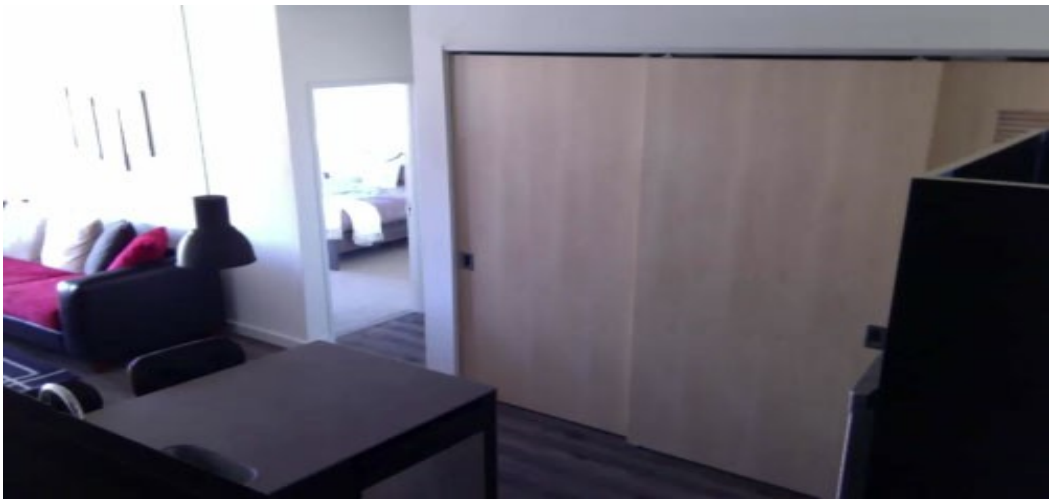


Figure 1 The above frame satisfies the assumption that the first frame of the video is simply the static background — no motion is taking place.

3. Now that we have our background modeled via a variable, we can utilize it to compute the difference between the initial frame and subsequent new frames from the video stream.

4. Computing the difference between two frames is a simple subtraction, where we take the absolute value of their corresponding pixel intensity differences.

5. **delta = |background_model** – **current_frame|**.

6. The background of the image is black. However, regions that contain motion (such as the region of walking someone through the room) are much lighter.

7. We will then threshold the frame delta to reveal regions of the image that only have significant pixel intensity values. If the delta is less than 25, we discard the pixel and set it to black (i.e., background). If the delta is greater than 25, we will set it to white (i.e., foreground).

$$Rk(x,y) = fk(x,y) - B(x,y)$$

$$Dk(x,y) = \begin{cases} 1 \; background \; Rk(x,y) > T \\ 0 \qquad target \; Rk(x,y) \leq T \end{cases}$$

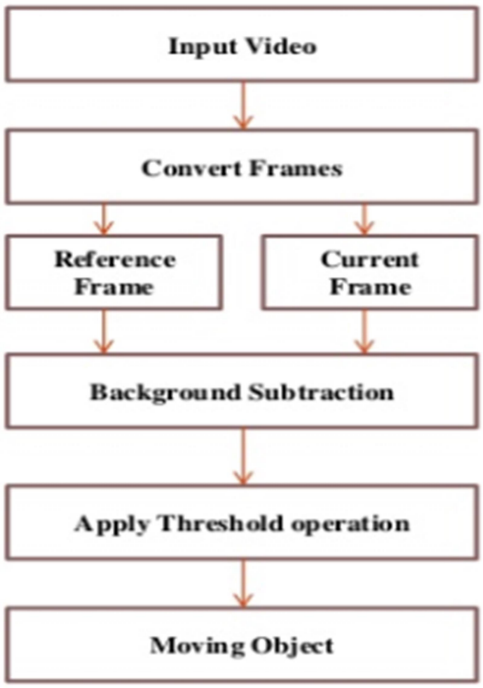


Figure 2 Describe the flow of Algorith.

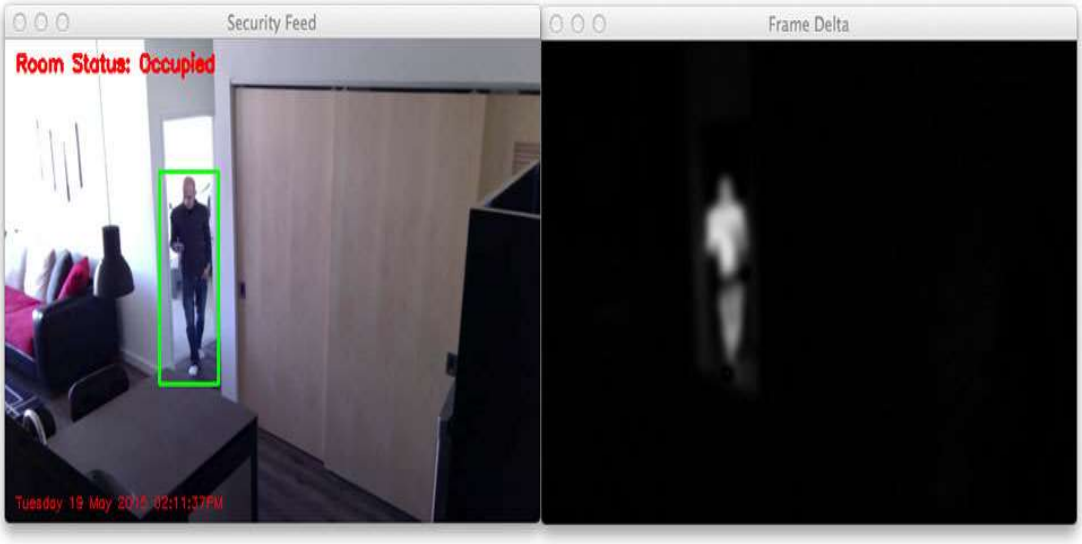


Figure 3 An example of the frame delta, the difference between the original first frame and the current frame.

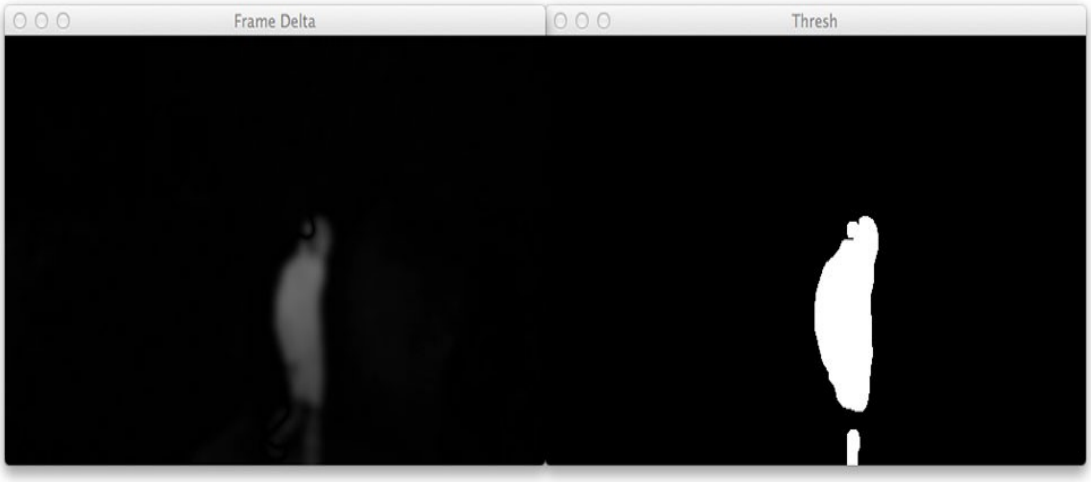


Figure 4 Thresholding the frame delta image to segment the foreground from the background.

8. Note that the image's background is black, whereas the foreground (and where the motion is taking place) is white. Given this thresholded image, it is simple to apply contour detection to find these white regions' outlines.
9. We start looping over each of the contours, where we will filter the small, irrelevant contours. If the contour area is more extensive than our supplied --min-area, we will draw the bounding box surrounding the foreground and motion region. We will also update our text status string to indicate that the target area is “Occupied” and generate the signal in the form of voice, and the motion is detected.

Chapter 3 Design Assumptions and Dependencies

3.1 Assumptions of the system can be listed as follows

1. Users are expected to look at augmented view provided on the screen.
2. Camera should not have any flaws on its lens.
3. There should not be any obstacle between the camera and the object.
4. Camera is connected to the raspberry pi that does the processing and viewing.
5. Camera will be at normal height at 45 degree angle.

3.2 Design Goals and Guidelines

There are two major goals relevant to the implementation. These are the performance and accuracy of the system.

- **Performance**: Performance is one of the two major issues for our system. The system should have a high performance such that the user should see the augmented view of the area.
- **Simplicity**: Regarding the user interface, an important design goal is keeping it simple. The user interface should not be so complicated. Besides, the user should not need to do anything for basic usage of the system.
- **Security**: Since the system will be used online and by a single user, security needs to maintain, such as interference and hack-proof network data transmission.
- **Availability**: System will be available as long as the computer running on is available.
- **Maintainability**: There will not be so much change in the future. Therefore, maintainability is not a big issue.

3.3 Data organization

The system will be organized using the repository model. In our system, data will flow between the main repository and sub-systems. The input to the system is the image frame captured from the webcam. This data will be passed to its local repository and distributed between sub-systems by the central repository. As output, the system will show the augmented view. The general structure of the system organization in terms of data management can be seen in the diagram below:

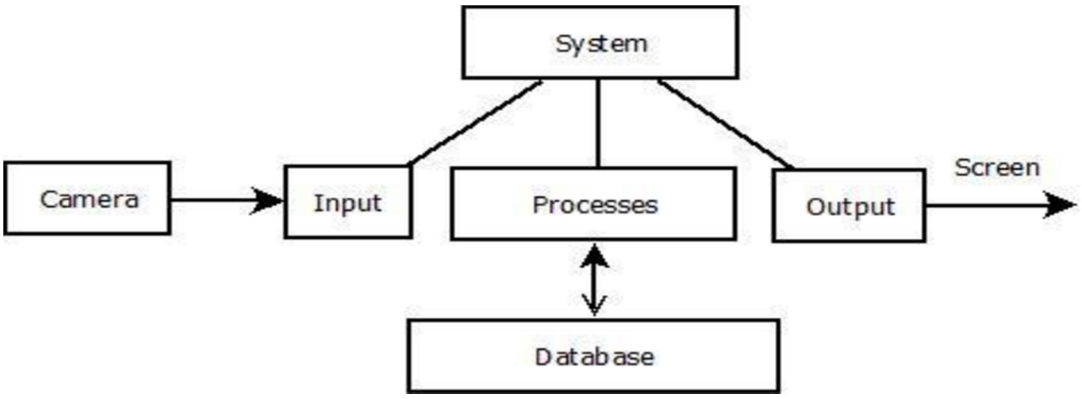


Figure 5 Model Block diagram

3.4 Use-case Diagram

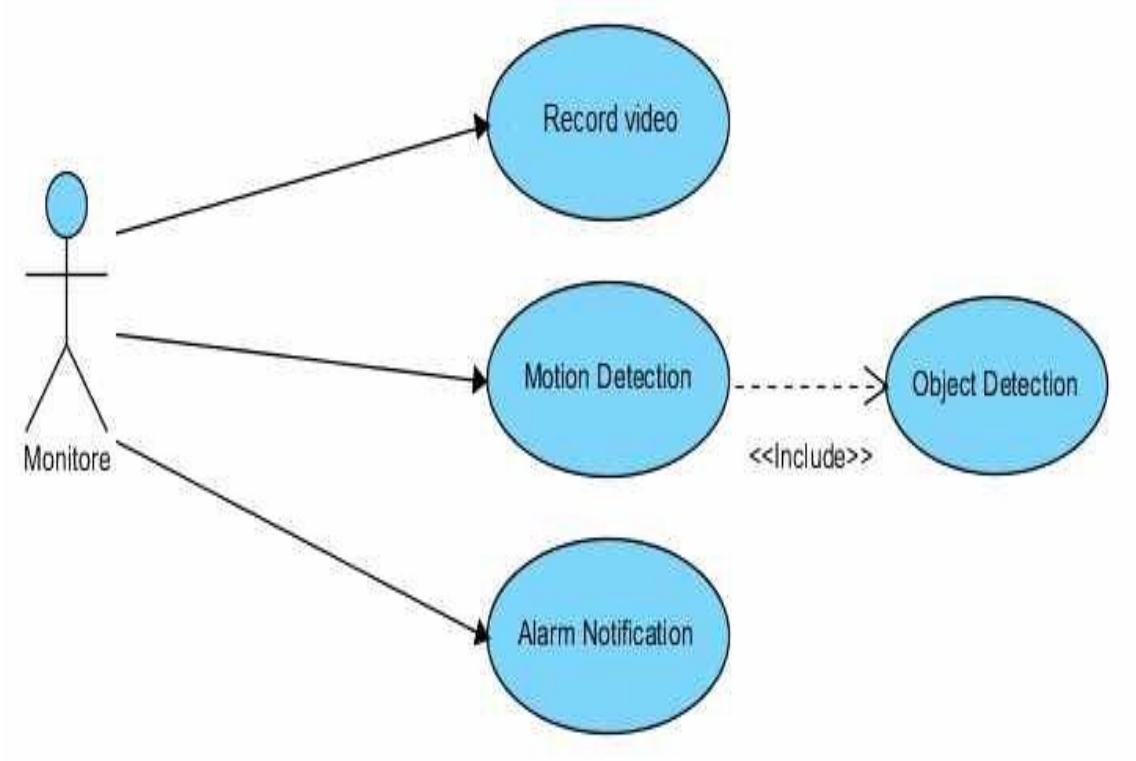


Figure 6: Use case diagram of the model.

3.5 Flow Chart Diagram

Our project's methodology is motion detection performed on the video stream received by the cameras is described in the following diagram.

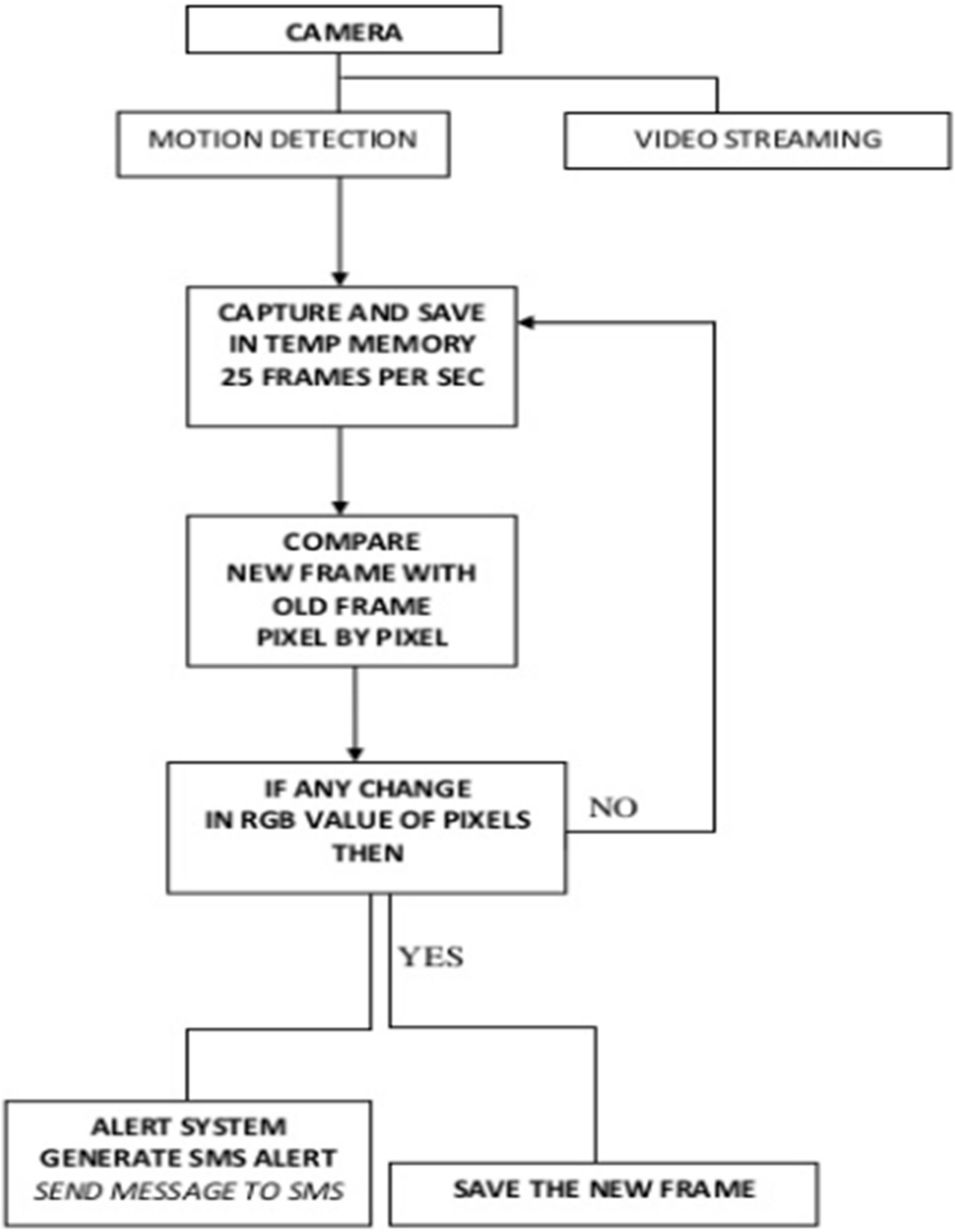


Figure 7: Flow chart of the model.

3.6 Communication interface

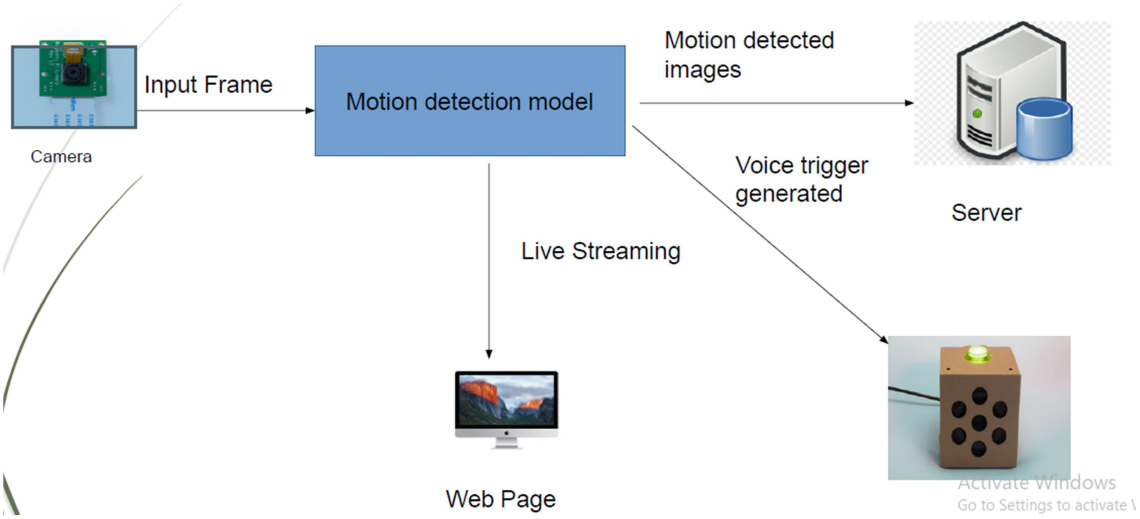


Figure 8: communication interface.

3.7 Hardware interface

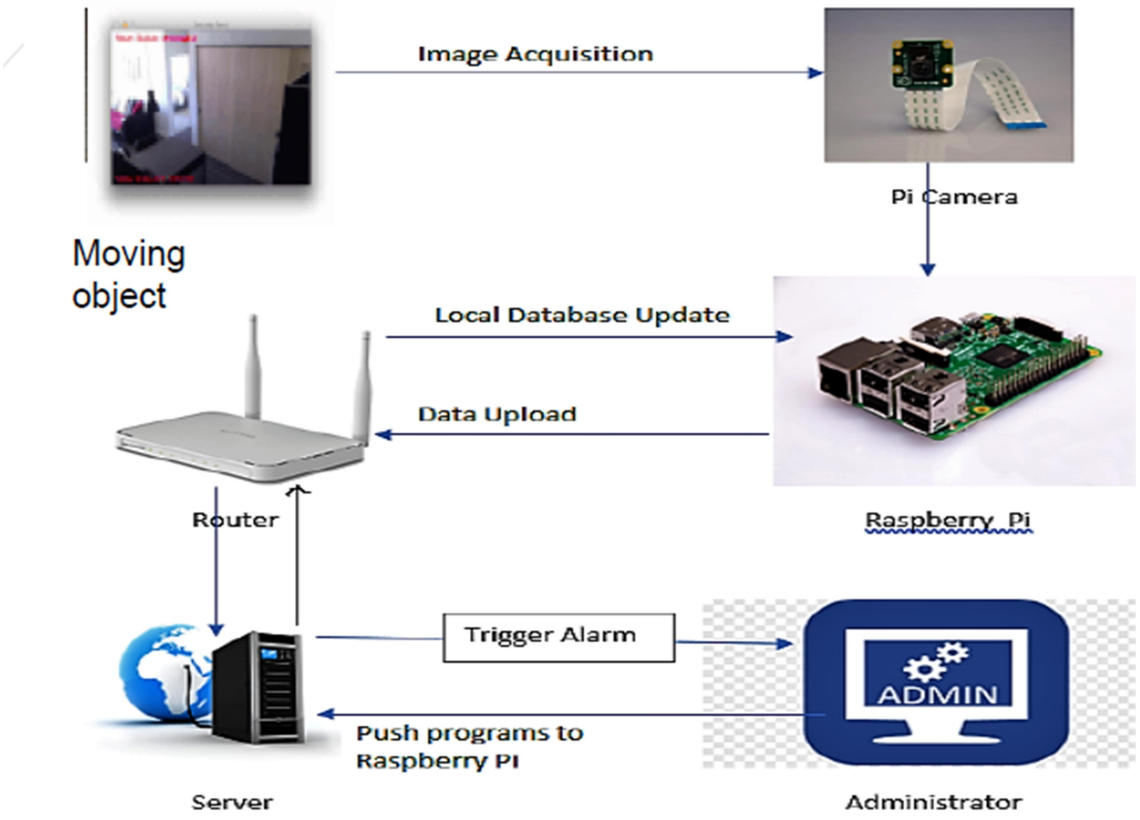


Figure 9: Hardware interface.

Chapter 4 Operating Environment

4.1 Software requirement

- Raspbian operating system.
- OpenCV 3
- Python 3.5
- Flask 1.1.2
- Werkzeug 1.0.1
- Numpy
- MySQL
- Database Server

4.2 Hardware requirement

- Raspberry Pi 4.
- Camera.
- Memory card.

Chapter 5 System Features

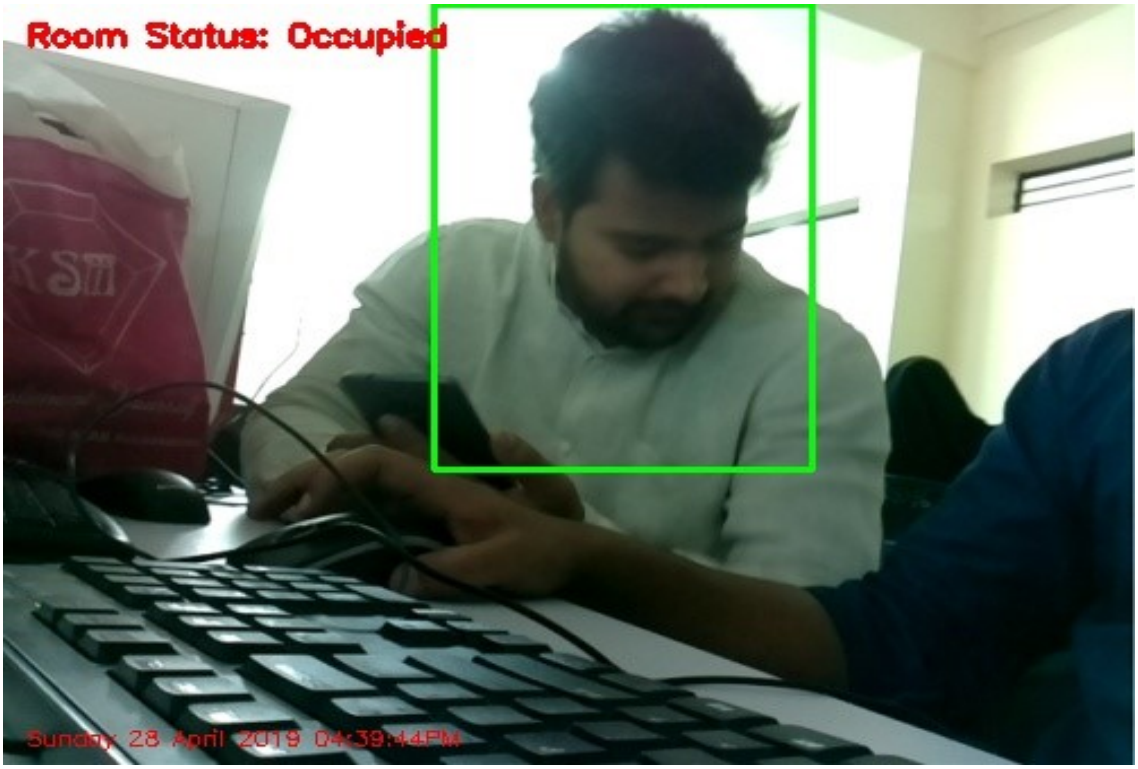
5.1 Automatic recording

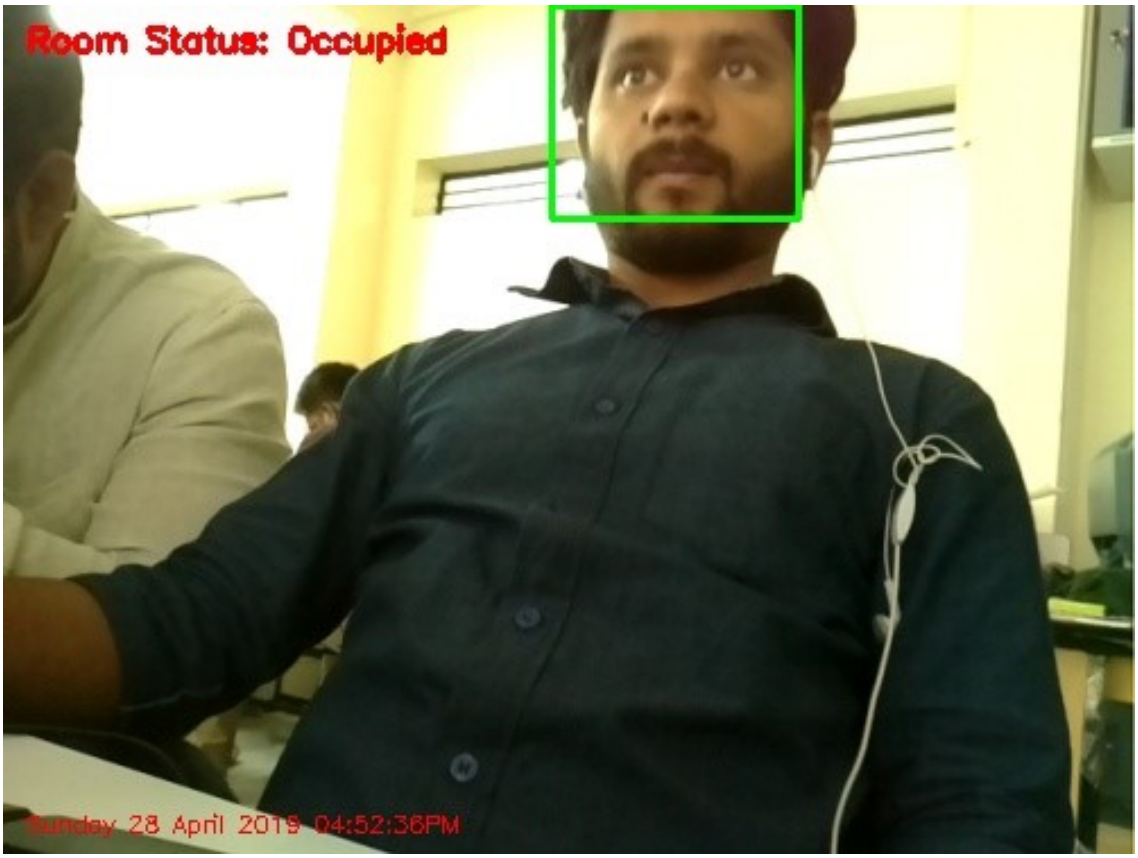
Automatically record the streaming for that amount of time when objects' motion is happing and store it in the database at the server.

5.2 Live Streaming

Motion detected frames would be streaming live on the web browser with a bounding box associated with the current time.

Chapter 6 Results




Room Status: Occupied
Sunday 28 April 2019 04:52:36PM