VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

BURGLAR DETECTION USING CLOSED-CIRCUIT TELEVISION

A Project Report

submitted by

Kandra Ksheeraj	(19BCE0829)
Kandala Sai Krishna	(19BCE0832)
Nakka Deepak	(19BCE0499)

CSE4019 - Image Processing
Slot: A1+TA1

School of Computer Science and Engineering
VIT, Vellore

December 2021

SUBMITTED TO
Prof. NATARAJAN P
SCOPE, VIT VELLORE

Contents

Abstract

Introduction

Literature Survey

Techniques used

Requirements

Results: Implementation, Code

Conclusion & Future Work

References

Abstract

Burglary is one of the most pervasive forms of crime perpetrated against the general public. It's also known as Breaking and Entering or Home Invasion, and it's defined as an unauthorised entry into another's property with the goal of committing a crime there. The manner in which detectives and police officers investigate this crime could mean the difference between solving the case and allowing burglaries to continue in the community, and burglaries have some of the lowest clearance rates in police investigations.

Computer vision allows computers and systems to extract useful data from visual inputs such as digital photos and videos. Most people have used the cv2.videocapture() function to read from a webcam or a video recording from a disc, but only a few people are aware of how simple it is to stream a video from a URL, which is usually an IP camera's URL. Using OpenCV's cv2, stream a live video from your phone's camera. VideoCapture() function in your PC and do all sorts of image processing on the spot. When someone enters your room, this serves as a surveillance system, recording video samples. You may easily expand this system by replacing the mobile camera with multiple IP cameras.

In recent years, there has been an increase in the number of thefts. This creates a dangerous environment where people live in fear. The issue of home security is a topic of concern in today's society. The present conventional intruder detection systems are quite costly, and there is a possibility of false alarms. This problem is solved by integrating OpenCV and a mobile phone to construct a framework that can accurately identify an intruder while filtering out the motions generated by moving objects. If an intruder is detected, the system saves the footage to local storage.

Introduction

The presence of a human safety officer is not a completely effective or dependable security method. In such situations, our framework detects intruders and safeguards the property of the owner. We can reduce robberies by implementing this approach. As a result, we will be able to react quickly so that no damage to our homes occurs. We can track down the interloper if we have the interloper's facial details and file a police report against him. The camera is installed in a room that can continually capture video in the planned framework. When a gatecrasher enters the room, our framework will go through a client-side process. Using OpenCV and Python, we built a framework that is entirely dependent on motion detection.

Existing model

Many monitoring and security technologies are already in place in today's environment. They either require a human safety officer or installation work, both of which may result in false alarms. In certain surveillance systems, an intruder is spotted using video recording cameras and the captured video is stored on an external storage disc, necessitating a significant expenditure in order to regulate, store, and monitor the actions. However, an owner must manually study the tape to locate the gatecrasher, which may result in time waste because he must watch the full recording to find the interloper and may overlook minor facts while doing so. The second method is a radar-based system that identifies an intruder by emitting microwaves or radiofrequency waves that bounce off any object in its path using a radar system. The intruder alarming system, on the other hand, incorporates electric alarms that are assigned to inform the owner when an intruder arrives. Passive infrared motion detection systems, ultrasonic motion detection systems, vibration sensor systems, and other systems are only a few examples.

Proposed system

For detecting the intruder, the proposed system uses an IP webcam programme that needs to be loaded on a mobile phone and python code. Both will function as client-server systems. The video is continuously collected and filtered on the client-side. The filtered data is then delivered to the server. It also includes a time and date stamp to show the status of the intruder. If the framework detects an intruder, it begins recording the current frame and displays a level of patience. The recording procedure will finish when the patience level reaches zero, and the recorded footage will be saved to local storage.

Literature Survey

[1] M. Piccardi (October 2004). "Background subtraction techniques: a review". IEEE International Conference on Systems, Man and Cybernetics 4. pp. 3099–3104.doi:10.1109/icsmc.2004.1400815 b
[2] Yoshida, T., 2004. Background differencing technique for image segmentation based on the status of reference pixels." International Conference on Image Processing ICIP'04, pages 3487–3490, Singapore, October 24-27, 2004
[3] Elhabian, S., K. El-Sayed and S. Ahmed, 2008 Moving object detection in spatial domain using background removal techniques - State-of-Art. Recent Pat on Computer Sci., 1(1): 32-54
[4] Papert, The summer vision project, Massachusetts Institute of technology, 1996
[5] Efstration Graves, A brief history of computer vision [Internet]. 2016 [updated 05/05- 2016, cited 05/05-2016]. Available at: http://www.egavves.com/a-brief-historyofcomputer-vision/#sthash.b1miJmj3.dpbs
[6] Roqueiro, Petrushin, Counting people using video cameras, Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA, 2006
[7] Havaei1, Jodoin1, Efficient interactive brain tumour segmentation as within-brain kNN classification. University Sherbrooke, Canada. 2014
[8] Mobileye, Mobileye - Our Vision. 2016 [updated 04/25- 2016, cited 25/04-2016]. Available at: http://www.mobileye.com/
[9] Vargas, M., F. Barrero and J.M. Milla, 2008. An Enhanced Background Estimation Algorithm for Vehicle
Detection in Urban Traffic Video Proceedings of the 11 International IEEE Conference on Intelligent Transportation Systems, Beijing, China, October 12-15, 2008.
[10] Shanker, Lai, Android porting concepts, Electronics Computer Technology (ICECT), 2011 3rd International Conference on, Kanyakumari, 2011, pp. 129-133.

Techniques used

Background Subtraction

The most popular Background subtraction algorithms are:

1. BackgroundSubtractorMOG() algorithm: It is one of the background subtraction algorithms which uses the concept of the gaussian mixture.

2. BackgroundSubtractorMOG2() algorithm: It is the same concept as BackgroundSubtractorMOG() but the major difference it can provide stability even in high light intensity conditions and also stability while identifying the shadows in each frame.

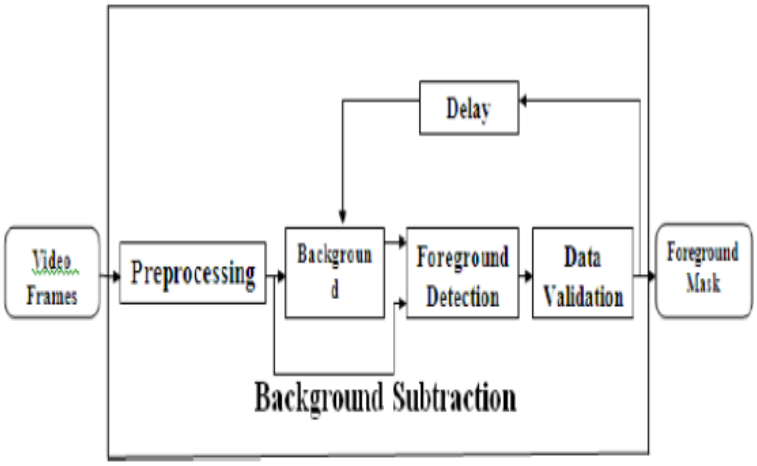
3. Geometric multigrid: It makes use of the statistical method and per pixel Bayesian segmentation algorithm.

The process becomes more complicated when there is a shadow without the actual image or appearance of the object. Simple background subtraction algorithms will consider the moving or still shadows as foreground which reduces the accuracy of the system. The solution for the above complications will be solved openCV.js which can simple and easy to use. The below constructor is the solution. The cv2.createBackgroundSubtractorMOG2() takes in 3 contentions:

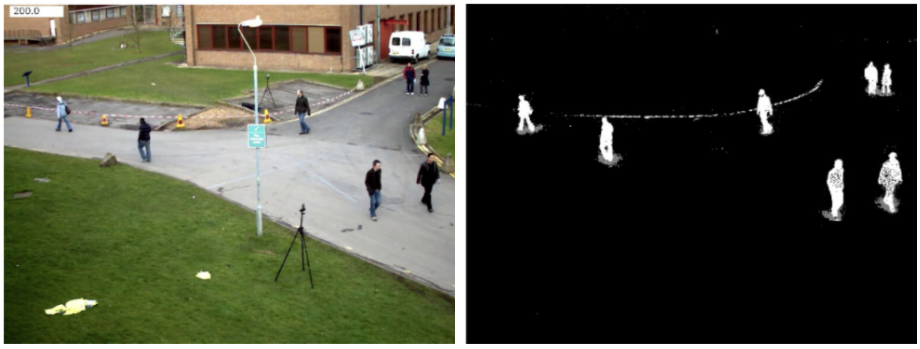
1. detectsShadows: Now this calculation can likewise recognize shadows, on the off chance that we pass in detectShadows=True argument in the constructor. The capacity to identify and dispose of shadows will give us smooth and powerful outcomes. Empowering shadow discovery somewhat diminishes speed.

2. history: This is the number of edges that are utilized to make the foundation model, increment this number if your objective item frequently stops or stops briefly.

3. varThreshold: This edge will help you sift through commotion present in the casing, increment this number if there are heaps of white spots on the edge. In spite of the fact that we will likewise utilize morphological activities like disintegration to dispose of the commotion

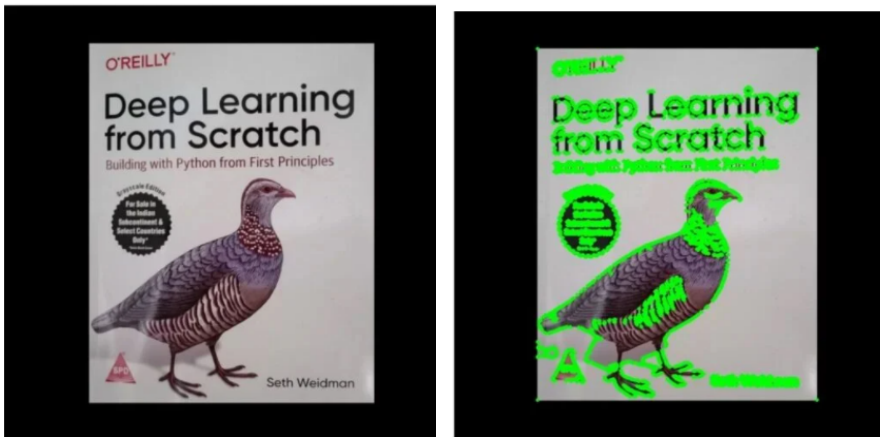


It is a technique for separating out foreground elements from the background and is done by generating a foreground mask. This technique is used for detecting dynamically moving objects from static cameras. Background subtraction technique is important for object tracking. There are several techniques for background subtraction. The foremost step of the method comprises of generation of background frames using statistical information of an initial set of frames not containing any targets. The generated background frame is made adaptive by continuously updating the background using the motion information of the scene.



Contour Detection

Detect the borders of objects, and localize them easily in an image. Applications such as image-foreground extraction, simple-image segmentation, detection and recognition. The background subtraction method followed by a clutter rejection stage ensure the detection of foreground objects. The next step comprises of detection of contours and distinguishing the target boundaries from the noisy background. This is achieved by using the Canny edge detector that extracts the contours followed by a *k*-means clustering approach to differentiate the object contour from the background contours. The post-processing step comprises of morphological edge linking approach to close any broken contours and finally, flood fill is performed to generate the silhouettes of moving targets.



OpenCV

Computer vision is a space of computer programming which bases on making PC's fit for interpreting pictures. It got out during the '70s when Martin Minsky asked is a student to interface a computer to a camera and get the PC to evaluate what it saw [4]. During this time the state of the art of computer vision transformed into reality. Today CV is consistently used with AI which is set up to recognize certain features or things like remembering the portion of people. cv2.VideoCapture() can be used to

- **Using Live camera feed:** You pass in an integer number i.e. 0,1,2 etc e.g. cap = cv2.VideoCapture(0), now you will be able to use your webcam live stream. The number depends upon how many USB cams you attach and on which port.
- **Playing a saved Video on Disk:** You pass in the path to the video file e.g. cap = cv2.VideoCapture(Path_To_video).
- **Live Streaming from URL using Ip camera or similar:** You can *stream from a URL e.g. cap = cv2.VideoCapture(protocol://host:port/video)* **Note:** that each video stream or IP camera feed has its own URL scheme.
- **Read a sequence of Images:** You can also read sequences of images, e.g. GIF.

Python

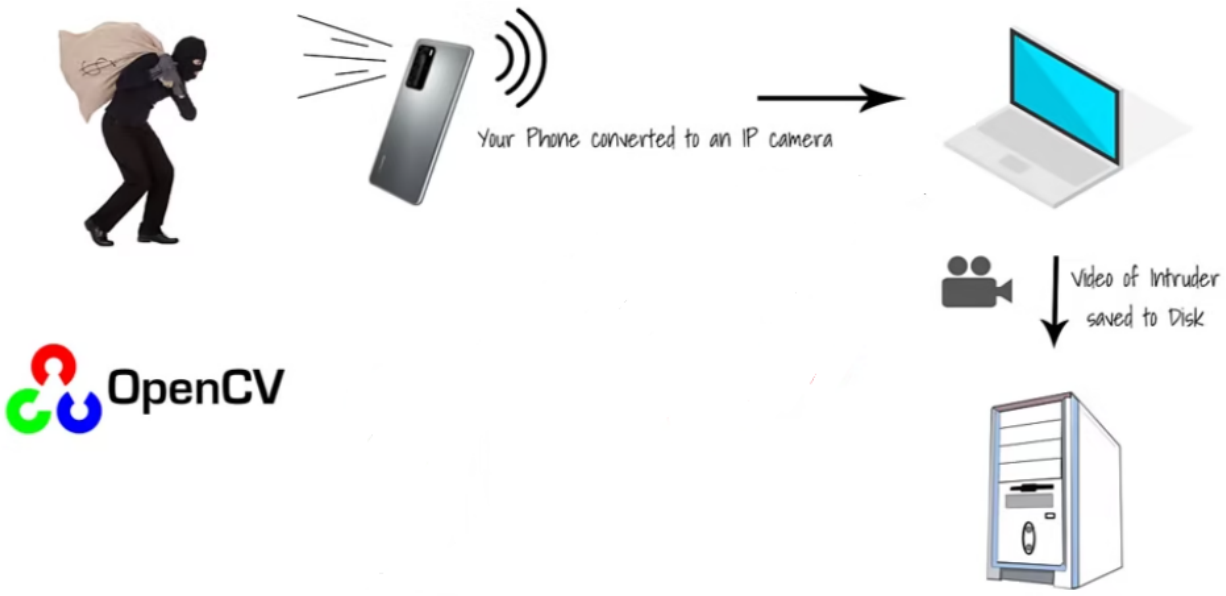
Python is a globally reputed language that is proposed to be significantly coherent. It is an interpreted language that resembles PERL and PHP. It is very interactive in nature and has a user-friendly interface. It is set to be object-oriented which maintains the procedure of programming that optimizes code inside objects.

We will be splitting the project into parts that have their own functionality.
1. Accessing the Live stream from your telephone to
OpenCV.
SMS.
2. Building a Motion Detector with Background Subtraction and Contour recognition.
3. Making the Final Application.

Requirements

- Operating System: Windows 10, Mac OS 11.2.2
- IP Webcam app on mobile or wifi smart CCTV cams
- TOOL: The external storage device or computer storage
- Jupyter notebook or google collab
- System: Intel 1.90 GHz
- A Mobile Phone of RAM: 2GB or more
- WIFI support 5 MP
- Camera Storage Space: 4GB
- Libraries
    1. Numpy
    2. Open
    3. Time
    4. Datetime
    5. deque

Architecture Diagram



Code

Burglar Detection

In [1]:
```
# Import the required libraries
import numpy as np
import cv2
import time
import datetime
from collections import deque
```

Reading from your Phone's IP Camera

In [2]:
```
# Set Window normal so we can resize it
cv2.namedWindow('frame', cv2.WINDOW_NORMAL)

# Note the starting time
start_time = time.time()

# Initialize these variables for calculating FPS
fps = 0
frame_counter = 0

# Read the video steram from the camera
cap = cv2.VideoCapture('http://192.168.31.160:8080/video')

while(True):

    ret, frame = cap.read()
    if not ret:
        break

    # Calculate the Average FPS
    frame_counter += 1
    fps = (frame_counter / (time.time() - start_time))

    # Display the FPS
    cv2.putText(frame, 'FPS: {:.2f}'.format(fps), (20, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255),1)

    # Show the Frame
    cv2.imshow('frame',frame)

    # Exit if q is pressed.
    if cv2.waitKey(1) == ord('q'):
        break

# Release Capture and destroy windows
cap.release()
cv2.destroyAllWindows()
```

Background Subtraction

In [3]:
```
# load a video
cap = cv2.VideoCapture('sample_video.mp4')

# Create the background subtractor object
foog = cv2.createBackgroundSubtractorMOG2( detectShadows = True, varThreshold = 50, history = 2800)

while(1):

    ret, frame = cap.read()
    if not ret:
        break

    # Apply the background object on each frame
    fgmask = foog.apply(frame)

    # Get rid of the shadows
    ret, fgmask = cv2.threshold(fgmask, 250, 255, cv2.THRESH_BINARY)

    # Show the background subtraction frame.
    cv2.imshow('All three',fgmask)
    k = cv2.waitKey(10)
    if k == 27:
        break

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

Contour Detection with Background Subtraction

In [4]:

```
# initlize video capture object
cap = cv2.VideoCapture('sample_video.mp4')

# you can set custom kernel size if you want
kernel= None

# initilize background subtractor object
foog = cv2.createBackgroundSubtractorMOG2( detectShadows = True, varThreshold = 50, history = 2800)

# Noise filter threshold
thresh = 1100

while(1):
    ret, frame = cap.read()
    if not ret:
        break

    # Apply background subtraction
    fgmask = foog.apply(frame)

    # Get rid of the shadows
    ret, fgmask = cv2.threshold(fgmask, 250, 255, cv2.THRESH_BINARY)

    # Apply some morphological operations to make sure you have a good mask
  # fgmask = cv2.erode(fgmask,kernel,iterations = 1)
    fgmask = cv2.dilate(fgmask,kernel,iterations = 4)

    # Detect contours in the frame
    contours, hierarchy = cv2.findContours(fgmask,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

    if contours:

        # Get the maximum contour
        cnt = max(contours, key = cv2.contourArea)


        # make sure the contour area is somewhat hihger than some threshold to make sure its a person and not some noise.
        if cv2.contourArea(cnt) > thresh:

            # Draw a bounding box around the person and label it as person detected
            x,y,w,h = cv2.boundingRect(cnt)
            cv2.rectangle(frame,(x ,y),(x+w,y+h),(0,0,255),2)
            cv2.putText(frame,'Person Detected',(x,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.3, (0,255,0), 1, cv2.LINE_AA)


    # Stack both frames and show the image
    fgmask_3 = cv2.cvtColor(fgmask, cv2.COLOR_GRAY2BGR)
    stacked = np.hstack((fgmask_3,frame))
    cv2.imshow('Combined',cv2.resize(stacked,None,fx=0.65,fy=0.65))

    k = cv2.waitKey(40) & 0xff
    if k == ord('q'):
        break

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

Function for detecting if a person is present or not

In [5]:

```
def is_person_present(frame, thresh=1100):

    global foog

    # Apply background subtraction
    fgmask = foog.apply(frame)

    # Get rid of the shadows
    ret, fgmask = cv2.threshold(fgmask, 250, 255, cv2.THRESH_BINARY)

    # Apply some morphological operations to make sure you have a good mask
    fgmask = cv2.dilate(fgmask,kernel,iterations = 4)

    # Detect contours in the frame
    contours, hierarchy = cv2.findContours(fgmask,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

    # Check if there was a contour and the area is somewhat higher than some threshold so we know its a person and not noise
    if contours and cv2.contourArea(max(contours, key = cv2.contourArea)) > thresh:

            # Get the max contour
            cnt = max(contours, key = cv2.contourArea)

            # Draw a bounding box around the person and label it as person detected
            x,y,w,h = cv2.boundingRect(cnt)
            cv2.rectangle(frame,(x ,y),(x+w,y+h),(0,0,255),2)
            cv2.putText(frame,'Person Detected',(x,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.3, (0,255,0), 1, cv2.LINE_AA)

            return True, frame


    # Otherwise report there was no one present
    else:
        return False, frame
```

Sample Script for recording Testing videos

In [ ]:

```
import numpy as np
import cv2

cap = cv2.VideoCapture('video.mp4')

# This is how we define our codec
fourcc = cv2.VideoWriter_fourcc(*'XVID')
width = int(cap.get(3))
height = int(cap.get(4))

# Save our video at 20.0 FPS with the defined dimensions, Note my default camera has these dimensions, yours might be differe
out = cv2.VideoWriter(r'sampletestnew2.mp4',fourcc, 15.0, (width, height))

while(True):
    ret, frame = cap.read()
    if not ret:
        break

    # Save this frame
    out.write(frame)

    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the video writer Object
out.release()
cap.release()
cv2.destroyAllWindows()
```

Final Application

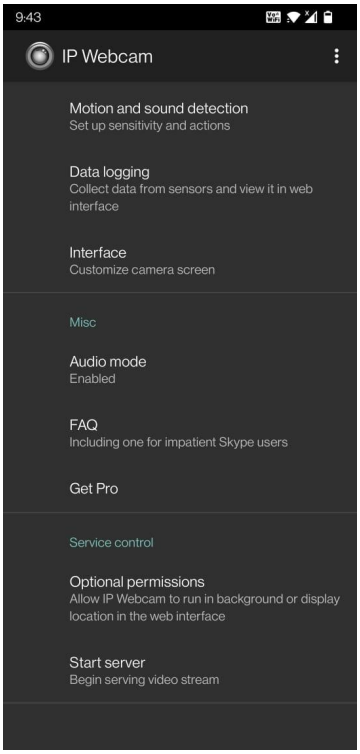
In [7]:

```
#time.sleep(15)

# Set Window normal so we can resize it
cv2.namedWindow('frame', cv2.WINDOW_NORMAL)

cap = cv2.VideoCapture('http://192.168.31.160:8080/video')

# Get width and height of the frame
width = int(cap.get(3))
height = int(cap.get(4))

info_dict = eval(data)

# Initialize the background Subtractor
foog = cv2.createBackgroundSubtractorMOG2( detectShadows = True, varThreshold = 100, history = 2000)

# Status is True when person is present and False when the person is not present.
status = False

# After the person disapears from view, wait atleast 7 seconds before making the status False
patience = 7

# We don't consider an initial detection unless its detected 15 times, this gets rid of false positives
detection_thresh = 15

# Initial time for calculating if patience time is up
initial_time = None

# We are creating a deque object of length detection_thresh and will store individual detection statuses here
de = deque([False] * detection_thresh, maxlen=detection_thresh)

# Initialize these variables for calculating FPS
fps = 0
frame_counter = 0
start_time = time.time()


while(True):

    ret, frame = cap.read()
    if not ret:
        break

    # This function will return a boolean variable telling if someone was present or not, it will also draw boxes if it
    # finds someone
    detected, annotated_image = is_person_present(frame)

    # Register the current detection status on our deque object
    de.appendleft(detected)

    # If we have consectutively detected a person 15 times then we are sure that soemone is present
    # We also make this is the first time that this person has been detected so we only initialize the videowriter once
    if sum(de) == detection_thresh and not status:
            status = True
            entry_time = datetime.datetime.now().strftime("%A, %I-%M-%S %p %d %B %Y")
            out = cv2.VideoWriter('outputs/{}.mp4'.format(entry_time), cv2.VideoWriter_fourcc(*'XVID'), 15.0, (width, height)

    # If status is True but the person is not in the current frame
    if status and not detected:

        # Restart the patience timer only if the person has not been detected for a few frames so we are sure it was'nt a
        # False positive
        if sum(de) > (detection_thresh/2):
```

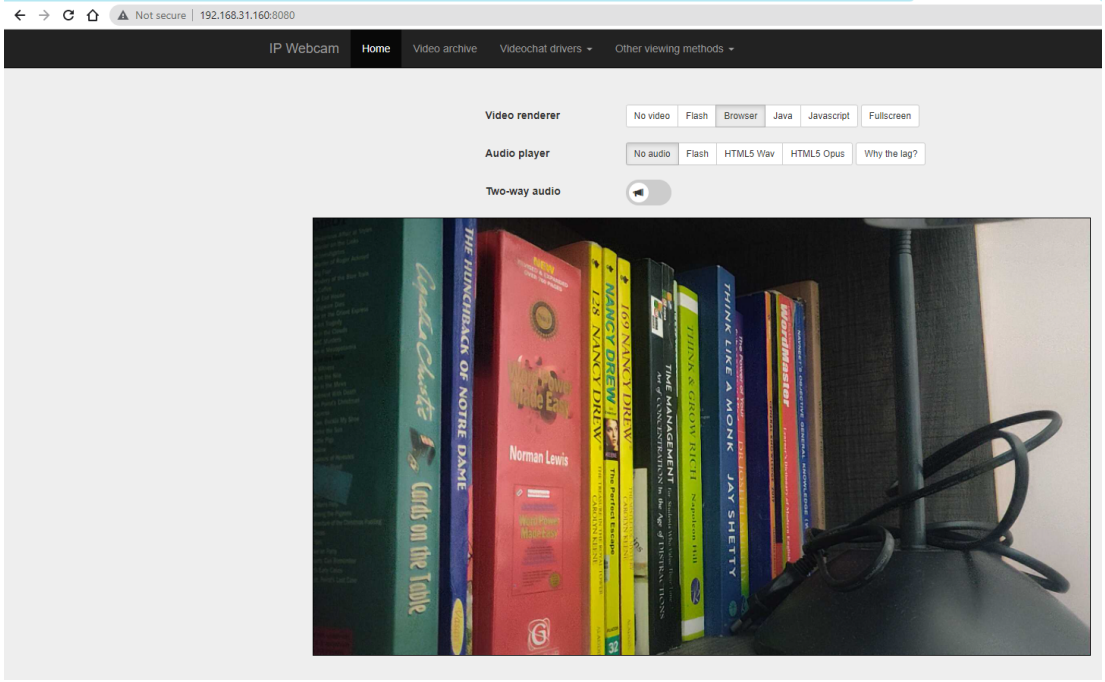
```
            if initial_time is None:
                initial_time = time.time()

        elif initial_time is not None:

            # If the patience has run out and the person is still not detected then set the status to False
            # Also save the video by releasing the video writer and send a text message.
            if  time.time() - initial_time >= patience:
                status = False
                exit_time = datetime.datetime.now().strftime("%A, %I:%M:%S %p %d %B %Y")
                out.release()
                initial_time = None

    # If significant amount of detections (more than half of detection_thresh) has occured then we reset the Initial Time.
    elif status and sum(de) > (detection_thresh/2):
        initial_time = None

    # Get the current time in the required format
    current_time = datetime.datetime.now().strftime("%A, %I:%M:%S %p %d %B %Y")

    # Display the FPS
    cv2.putText(annotated_image, 'FPS: {:.2f}'.format(fps), (510, 450), cv2.FONT_HERSHEY_COMPLEX, 0.6, (255, 40, 155),2)

    # Display Time
    cv2.putText(annotated_image, current_time, (310, 20), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 255),1)

    # Display the Room Status
    cv2.putText(annotated_image, 'Room Occupied: {}'.format(str(status)), (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.6,
                (200, 10, 150),2)

    # Show the patience Value
    if initial_time is None:
        text = 'Patience: {}'.format(patience)
    else:
        text = 'Patience: {:.2f}'.format(max(0, patience - (time.time() - initial_time)))

    cv2.putText(annotated_image, text, (10, 450), cv2.FONT_HERSHEY_COMPLEX, 0.6, (255, 40, 155) , 2)

    # If status is true save the frame
    if status:
        out.write(annotated_image)

    # Show the Frame
    cv2.imshow('frame',frame)

    # Calculate the Average FPS
    frame_counter += 1
    fps = (frame_counter / (time.time() - start_time))


    # Exit if q is pressed.
    if cv2.waitKey(30) == ord('q'):
        break

# Release Capture and destroy windows
cap.release()
cv2.destroyAllWindows()
out.release()
```

Results: Implementation

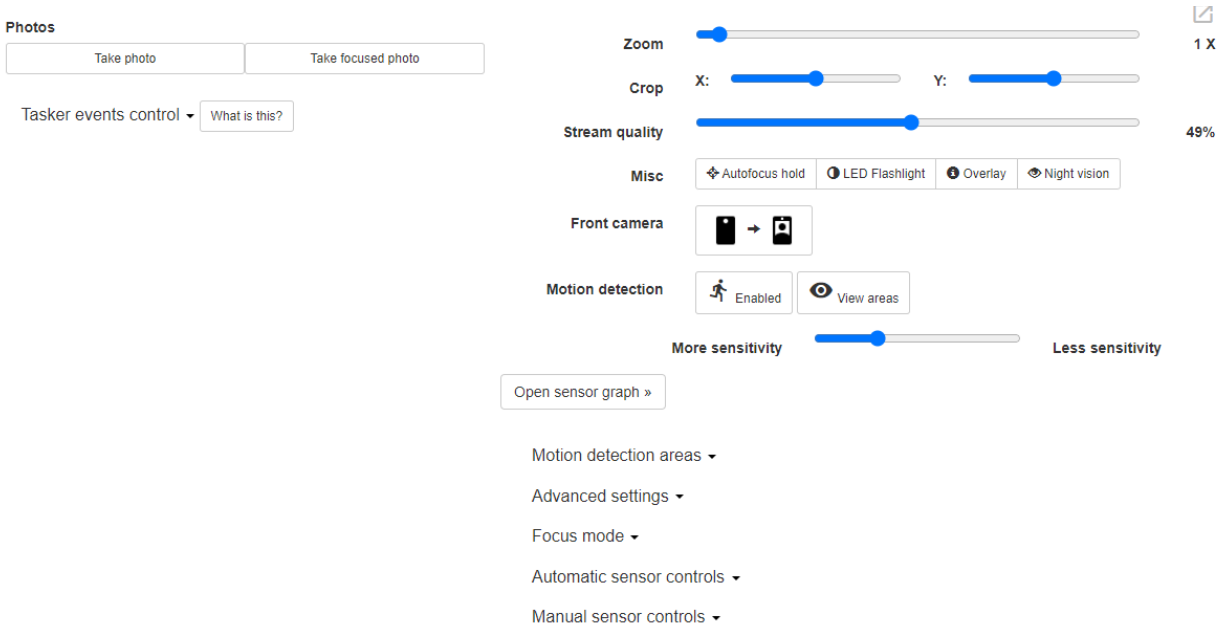
Starting an IP server



Footage accessible over the network

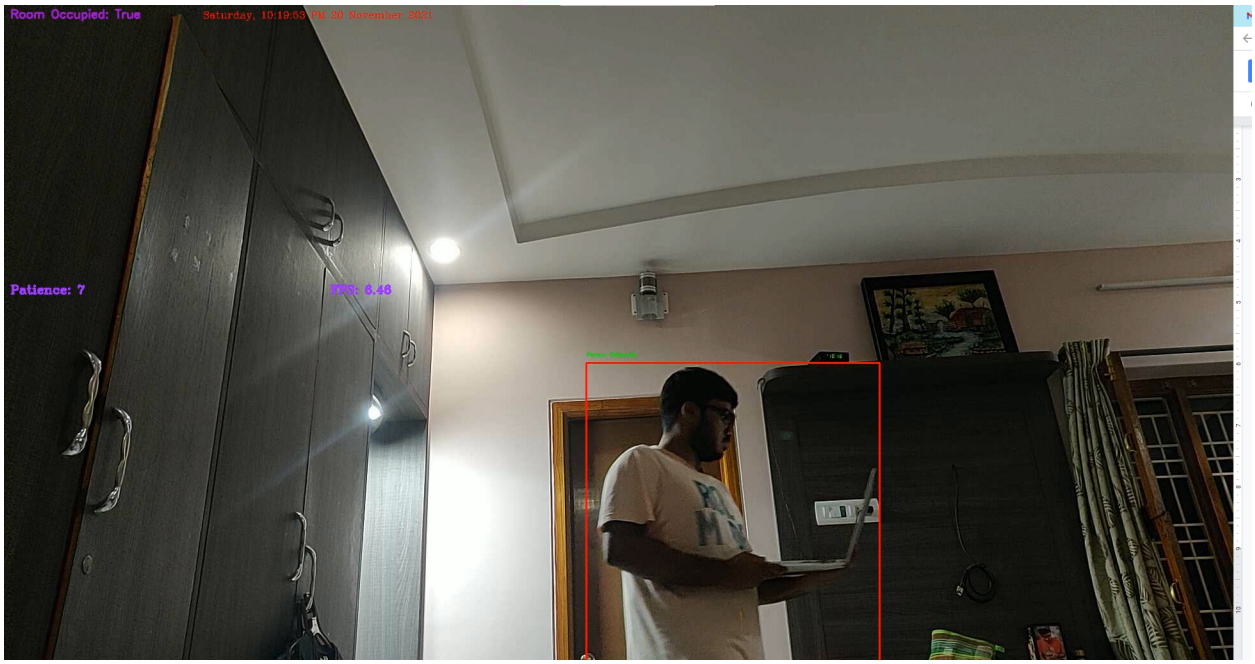


Adjustable features

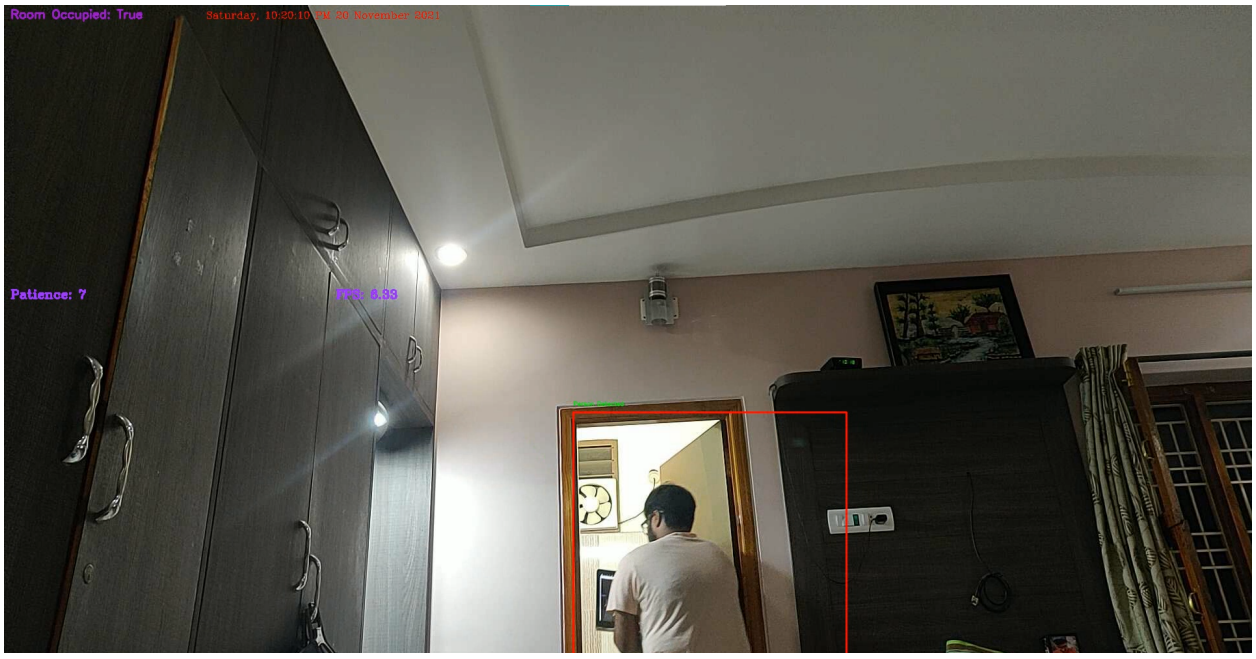


Start of the simulation

Start of the recording when person is detected



Before the end of the clip, when room occupied turns false and patience to 0



Drive Link to implementation videos:

https://drive.google.com/drive/folders/1NGHZ06qc8TQxmHfvPV80gnO6dXFEs0X-?usp=sharing

Conclusion & Future Work

Complete simulation of an easy simulation system has been achieved using appropriate techniques by decreasing additional effort for securing a place has been demonstrated. This project's application is a proof of concept for creating an application that can detect human motion in a room while simultaneously saving video clips to a local PC. To achieve the best operation, the built application is suited to a smartphone with a 5-megapixel camera. An internet connection is also required, as well as an IP camera application from the Google Play Store. The constructed application produces a decent detection result with improved accuracy when labelling the captured video clips with a correct file name (Date and Time Stamp), i.e., identifying the image whether or not it contains humans. The IP Camera has a dynamic IP and needs to update it every time.

The system consists of both hardware and software components that work together to offer a reliable motion detection system. The background deduction approach was used in OpenCV execution, and the results of the directed studies suggested that an interloper-based movement recognition system should be more precise, eliminating false alarms. The recommended system is partially self-contained and wireless, resulting in a dependable, robust, easy-to-use, and low-cost security system. For communication, WIFI has been utilised to connect to the internet. This work can be extended as:

- The IP Camera has a dynamic IP and needs to update it every time.
- Accessing IP Camera Outside of the local WIFI Network by determining external IP and configuring it.
- Sending notifications to registered mobile when some moment is captured.

References

[1] AIP Conference Proceedings 2173, 020013 (2019); https://doi.org/10.1063/1.5133928 Published Online: 11 November 2019

[2] P. Viola and M. J. Jones, "Robust real time face detection," International Journal Of Computer Vision vol.57,no.2,pp. 137-154, 2004.

[3] GV Balakrishna , B. Santhosh Kumar, ”Smart intruder detection using video surveillance,” International Journal Of Science Technology and Management, vol no.5,Issue no.2,February 2016.

[4] P. Viola and M. Jones, ”Rapid object detection using a boosted classifier of simple features,” IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 511–518,2001.

[5] K. Shrivastava, A. Verma, and S. P. Singh,” Distance Measurement of an Object or Obstacle by Ultrasound Sensors using P89C51RD2,” International Journal of Computer Theory and Engineering, vol. 2, No.1 February, 2010,1793-8201.

[6] K.S.Shilpashree, Lokesha.H and Hadimani Shivkumar,” Implementation of Image Processing on Raspberry Pi,” International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, Issue 5, May 2015.

[7] Havaei1, Jodoin1, Efficient interactive brain tumour segmentation as within-brain kNN classification. University Sherbrooke, Canada. 2014

[8] Mobileye, Mobileye - Our Vision. 2016 [updated 04/25- 2016, cited 25/04-2016]. Available at: http://www.mobileye.com/

[9] Vargas, M., F. Barrero and J.M. Milla, 2008. An Enhanced Background Estimation Algorithm for Vehicle Detection in Urban Traffic Video Proceedings of the 11 International IEEE Conference on Intelligent Transportation Systems, Beijing, China, October 12-15, 2008.

[10] Shanker, Lai, Android porting concepts, Electronics Computer Technology (ICECT), 2011 3rd International Conference on, Kanyakumari, 2011, pp. 129-133

[11] M. Piccardi (October 2004). "Background subtraction techniques: a review". IEEE International Conference on Systems, Man and Cybernetics 4. pp. 3099–3104.doi:10.1109/icsmc.2004.1400815

[12] Yoshida, T., 2004. Background differencing technique for image segmentation based on the status of reference pixels.” International Conference on Image Processing ICIP’04, pages 3487–3490, Singapore, October 24-27, 2004

[13] Elhabian, S., K. El-Sayed and S. Ahmed, 2008 Moving object detection in spatial domain using background removal techniques - State-of-Art. Recent Pat on Computer Sci., 1(1): 32-54

[14] Papert, The summer vision project, Massachusetts Institute of technology, 1996

[15] Roqueiro, Petrushin, Counting people using video cameras, Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA, 2006

[16] Shaily Pandey and Sandeep Sharma, ”An Optimistic Approach for Implementing Viola Jones Face Detection Algorithm in Database System and in Real Time,” International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 vol. 4 Issue 07, July-2015.