IoT enabled Video Surveillance System using Raspberry Pi

Rahul Muppanagouda Patil[1], Ram Srinivas[2], Rohith Y[3], N R Vinay[4] and Pratiba D[5]
Department of Computer Science and Engineering
R.V. College of Engineering, Bangalore
Karnataka, India
Email: {[1]rahulpatil61197, [2]ramsrinivas57, [3]rohithgoud10, [4]nrvinay08}@gmail.com, [5]pratibad@rvce.edu.in

***Abstract*—The Internet of Things (IoT) is the communication of various devices or "things" with each other by the sharing of usable and relevant data. In today's age of digitization and "smart" devices, one of the foremost requirements is for new, innovative technology integral to the concepts of Internet of Things in everyday life. The need for surveillance and monitoring has drastically increased over the course of the past few decades. Both private areas and public areas alike require these types of surveillance systems to ensure security, and are generally very costly. In this paper, is proposed, an innovative surveillance system which is powered by the Raspberry Pi, Amazon Web Services and Google Drive that provided results with minimal latencies.**

Keywords— Motion Detection, OpenCV, AWS, Raspberry Pi, Video Surveillance, Internet of Things, Google Drive

I. INTRODUCTION

The Internet of things (IoT) is the network of physical devices such as automobiles, home appliances, and other items embedded with electronics, software, sensors, actuators, and network connectivity which enable these objects to connect and exchange data. Each thing is uniquely identifiable through its embedded computing system but is able to inter-operate within the existing Internet infrastructure. The concept of IoT is currently being used in several fields such as agricultural industry, automotive industry, medical industry and of course, in security and surveillance. IoT expects use of low cost consuming and low power consuming devices with controlled adverse effects on the environment.

Video Surveillance systems have been an extremely essential part of security systems installed in all types of offices, schools, hotels, businesses and even people"s homes. The traditional methods of having around-the-clock surveillance using CCTV systems bore great burdens in the form of costs and maintenance to the consumer. They also lacked communication with other devices and were restricted to the sphere in which the system existed. The need of a connected surveillance system is higher than ever today, as there is a constant need by users to monitor people and entities that are dear to them and not everybody is capable of meeting the high prices and costs of these systems. Therefore, with the introduction of small and smart standalone computers such as the Raspberry Pi, any user can develop their own surveillance systems that are not just isolated systems, but ones that also interact with various other devices and infrastructures. Businesses can also provide such products at much lower costs now, very much in contrast to the systems provided in the past which were both costly and also required a lot of manual monitoring.

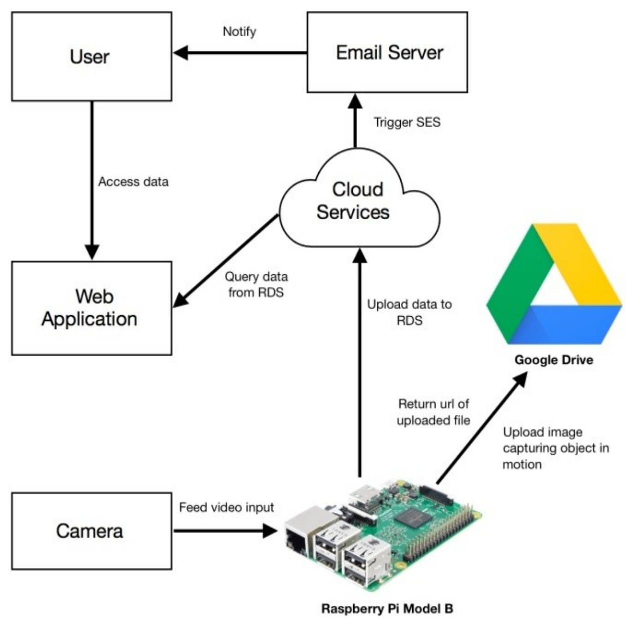

Fig. 1. Surveillance System Architecture

Applications such as Video Surveillance have become palpably better by their incorporation in the Internet of Things system, which is merely just connecting several devices and transferring relevant data from one to another, preferably in real-time.

In this paper, is further elaborated, a cost-effective surveillance system that has been developed using the popular background subtraction method to detect motion. The system also incorporates various other features using services provided by Amazon Web Services, such as SES for notifying the user via e-mail and AWS RDS for storing relevant data. Google drive has been used to store images on cloud to make it easily accessible to the user. A client-side GUI for users has also been developed, to view data in a user-friendly, interactive format and also to gain relevant insights from them. Hence, an IoT based system has been successfully implemented using the Raspberry Pi, AWS as well as Google Drive.

II. Related Work

Extensive research in the field of computer vision has paved way for many successful systems that utilize image processing techniques. The most common techniques include background subtraction and differential frame analysis. [1] proposes a three layer architecture. The first layer is the motion detection layer, which is the outcome of a Python script and the second layer triggers the actions based on a configuration file. The third layer triggers actions like sending an email, storing the images/videos to an FTP server. Live video streaming system based on Raspberry Pi connected to a 5MP Camera Module. The sensitivity of their system [2] revolves around the chosen value for a threshold which is a measure of the rate of change of a pixel location. [3] delivers a system that compares two successive images and determines the motion and takes the necessary steps following the detection. The method proposed in [4] uses a combination of temporal frame differences and background subtraction which can successfully detect slow moving targets as well. To improve the accuracy, a cross-correlation method is adopted for better results regarding slow and stationary objects and background updating is employed. A three-tier based approach was proposed in [5], one for efficiently determining the background (BM Module) using rapid matching followed by accurate matching, second being a novel block based entropy evaluation method for identification of block candidates of moving targets, and third being the object extraction (OE Module) which examined every block to identify a moving object.

The system proposed in this paper focus primarily on the intercommunication of devices and services to build a successful working surveillance system. It is highly capable and will work with any of the latest motion detection algorithms that have been proposed up till today.

III. Motivation

Existing systems that are commonly in use monitor areas, leaving their footage to be reviewed at a later stage. Alternatively, there exist many common security systems that serve as alert systems upon the intrusion of an infiltrator via one or many entry points, mostly to serve as a security system for residents residing within an apartment or building at the time. However, existing systems are not always area or room-specific, and do not alert the user about an infiltration, irrespective of the user‟s location. In addition, none are implemented using systems akin to the Raspberry Pi.

IV. Implementation

The system uses three-tier architecture, where the first tier is the background subtraction algorithm running to detect motion that keeps constantly running in a loop triggering certain functions when deemed necessary. The second tier is the database and cloud storage which stores important data in cloud servers. It also includes the e-mail servers that aid in notifying the user in real-time. The third and final tier is the part of the system that experiences user interaction. This includes a web application that has been developed so that the user can observe information related to their system.

The system is essentially divided into various steps or processes as listed below:

1. Video feed is continuously transferred from USB Camera to Raspberry Pi.
2. Motion detection script constantly running the background uses these input frames to make a decision based on certain parameters.
3. If motion is detected, and is long enough to be regarded as a significant motion, the frame in which motion is detected is immediately uploaded to the drive in cloud.
4. User is notified with relevant information such as motion start time along with an image URL that links to the image uploaded to the drive in previous step.
5. Motion time is uploaded into the RDS in cloud for future analysis and visualization.
6. User can log into a web application using given credentials to view important information regarding their systems and also visualize the data uploaded to database.

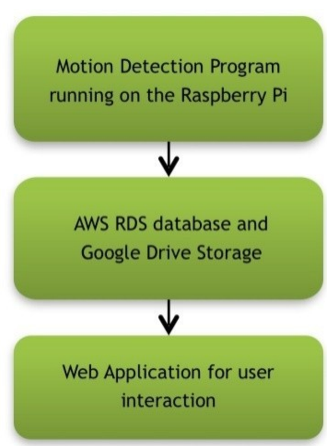

Fig. 2. Three tiers of the System

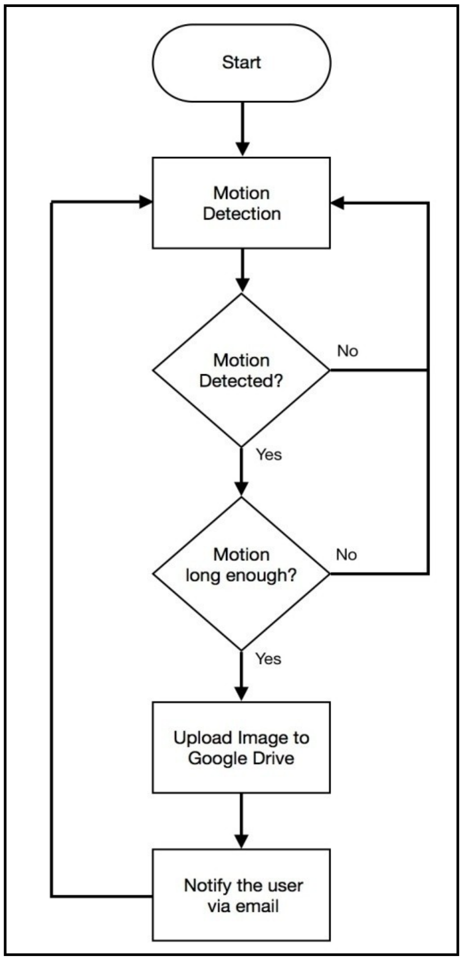

Fig. 3. System Flowchart

A. Motion Detection using Background Subtraction

The primary function of the system is to detect motion that occurs in a video stream,which is being captured and fed as input continuously. The method used must be able to make the decision in real-time and must therefore be quick and accurate. Motion detection using background subtraction is essentially performed by just comparing successive frames, and if significant change is noticed then motion is said to be occurred.

Also, the duration of the detected motions is considered by counting the number of frames in which motion is detected. Only if the number of these frames is greater than a minimum threshold, a response will be triggered.

The system also provides a provision where in motions that occur with minimal time gap be considered as one single motion. The time elapsed between successive motions is calculated for every motion detected and is considered different from the preceding detected motion only if the actual time elapsed is greater than or equal to some minimum time gap as specified by either the user or developer. This time gap can vary according to what type of environment the surveillance system is used for.

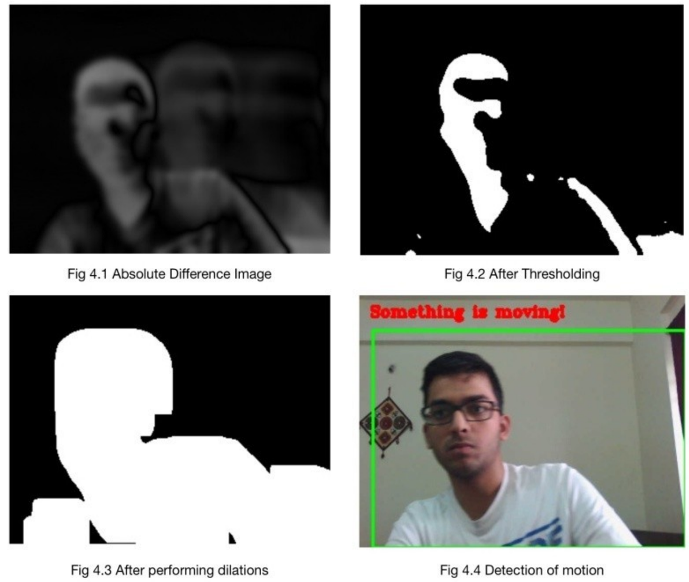

Fig. 4. Background Subtraction

The algorithm has been implemented in Python using the OpenCV module. The methods used on successive frames to detect motion are as described below:

1) First each frame is captured by the camera and transferred to the Raspberry Pi. If the frame is valid, then it is used for upcoming operations.

2) The frame is resized to half the size to reduce computations and is then converted to a single channel image. This makes finding the absolute difference between frames easy and less expensive.

3) The image is blurred using Gaussian Blur to de-noise the image and is then fed to the accumulateWeighted function to model the background dynamically.

4) The absolute difference is calculated between the current frame and the dynamic background to obtain Fig. 4.1 in which motion can be observed.

5) This image is then applied to a threshold so that only significant changes are considered, and is dilated to form proper solid entities that depict entity that caused the motion.

6) Finally contours are found in this dilated image, and only if contours are larger than a certain pre-defined threshold then a motion is said to be detected.

7) Finally a bounding rectangle is drawn over the entity that caused the motion by obtaining the coordinates of the contour and passing them as parameters to draw the rectangle.

This algorithm will tend to give inaccurate results in low light conditions. It detects changes in lighting as motion and does not perform to its optimum when there are shadows in the video. It also detects non-threatening motion such as displacement of household objects, which are nothing but false results.

B. *Storing image and relevant data on Cloud server*

In the Internet of Things era there is a need for this type of surveillance system to have the obtained data to be accessible and available to other authorized systems and applications, and not just be stored locally in the Raspberry Pi. The system satisfies this need by using a popular AWS service called RDS (Relational Database System) which is hosted on a virtual instance in the cloud. This remote storage makes it readily accessible to other applications that can give really good deductions from the data. It also helps the user to view information obtained from their systems in real-time from any authorized device that has access to the network.

The system uses the PyMySql library to establish connection to the MySQL server in the cloud and enables execution of SQL queries to insert or update data in the table.

To store images of the instances when motions are detected, popular cloud storage service „Google Drive" is used. This is done using the PyDrive API provided by Google to authenticate the user, upload image to drive and to extract a shareable link to the image uploaded directly from our main program.

MySQL [HOME_SURVEILLANCE]> select * from Motion_Detection;

motionid	entry_time	exit_time	image_link	user_id
0	2017-10-28 02:52:22	2017-10-28 02:52:23	https://docs.google.com/open?id=0B4fXzgZJLu5iMGNzamFfbkVYakE	1
1	2017-10-28 02:46:42	2017-10-28 02:46:43	https://docs.google.com/open?id=0B4fXzgZJLu5iQTZDV250UGloZlk	1
2	2017-10-28 02:46:49	2017-10-28 02:46:49	https://docs.google.com/open?id=0B4fXzgZJLu5iUHJqZE02NTgyY28	1
3	2017-10-28 02:46:55	2017-10-28 02:46:57	https://docs.google.com/open?id=0B4fXzgZJLu5iVS1fUHhXTWtLcFk	1
4	2017-10-28 02:47:11	2017-10-28 02:47:11	https://docs.google.com/open?id=0B4fXzgZJLu5iRG0xZnAxbkNxZzA	1
5	2017-10-28 02:47:20	2017-10-28 02:47:26	https://docs.google.com/open?id=0B4fXzgZJLu5iQkxaZVc0cDZjaXM	1
6	2017-10-28 02:47:38	2017-10-28 02:47:39	https://docs.google.com/open?id=0B4fXzgZJLu5iNVZFRE5mOXNFY2M	1
7	2017-10-28 02:47:45	2017-10-28 02:47:47	https://docs.google.com/open?id=0B4fXzgZJLu5ibEI3Q2hWcmRhMFk	1
8	2017-10-28 02:47:52	2017-10-28 02:47:53	https://docs.google.com/open?id=0B4fXzgZJLu5iTTBFVzRjVVVKYms	1
9	2017-10-28 02:55:11	2017-10-28 02:55:22	https://docs.google.com/open?id=0B4fXzgZJLu5ibXpZTThkMGhvb1k	1
10	2017-10-28 02:55:28	2017-10-28 02:55:36	https://docs.google.com/open?id=0B4fXzgZJLu5iazBmV1AyaFRZWHc	1
11	2017-10-28 02:55:46	2017-10-28 02:55:47	https://docs.google.com/open?id=0B4fXzgZJLu5iUThiRk4yNDlKNU0	1
12	2017-10-28 02:55:55	2017-10-28 02:55:56	https://docs.google.com/open?id=0B4fXzgZJLu5iYUM5bGpkVE1Jejg	1
13	2017-10-28 02:56:07	2017-10-28 02:56:07	https://docs.google.com/open?id=0B4fXzgZJLu5iZ0FKYnJpOXFPdEU	1
14	2017-10-28 10:18:23	2017-10-29 05:32:34	https://docs.google.com/open?id=0B4fXzgZJLu5iR1psZ1pJcVk5UVE	1
15	2017-10-28 10:18:40	2017-10-28 10:18:50	https://docs.google.com/open?id=0B4fXzgZJLu5iTF9LbnZvdHk1YzA	1
16	2017-10-28 10:19:03	2017-10-28 10:19:04	https://docs.google.com/open?id=0B4fXzgZJLu5iMUo2NlZXeXBGM0U	1
17	2017-10-28 10:50:21	2017-10-28 10:50:24	https://docs.google.com/open?id=0B4fXzgZJLu5iVE4yd2l6MzR3YUk	1
18	2017-10-28 10:50:41	2017-10-28 10:50:43	https://docs.google.com/open?id=0B4fXzgZJLu5ieE5GYW5nbmI4U0U	1
19	2017-10-28 10:52:39	2017-10-28 10:52:40	https://docs.google.com/open?id=0B4fXzgZJLu5iendxYnZhZGR1SmM	1

Fig. 5. Motion time and image URL entries in AWS RDS

C. *Notifying the User*

Simple Email Service is a free service provided by Amazon Web Service. This is part of our second tier which alerts the user if any motion were to occur. AWS has provided a Python module called boto3 which enables any AWS account holder to access all the services directly from their Python scripts.

On a Linux System, boto3 can be installed by using the pip package in Python: **sudo pip install boto3** and the credentials can be setup by specifying certain parameters such as the access key id, secret access key and the region in which the instance is hosted.

This feature is used to access the cloud services such as Amazon SES that notifies the user via e-mail, where the email contains important information such as the time motion was detected. It also contains a link to the image uploaded into the Google drive for that corresponding motion.

D. *Data visualization and User interaction*

For the application to be wholesome, it is imperative that the user functionality extend beyond merely that of receiving e-mail alerts and viewing pictures. There is a need for the user to be able to view all the data over a certain time-frame in some format, from a remote location at his/her leisure. Hence, a web portal was developed for the same. The user is required to merely login to the portal using his/her assigned credentials, and can subsequently view all the stored data regarding foreign movements detected in the form of various interactive graphs. The user could even choose to view the stored data in raw text form.

Python"s Flask module was used to develop the web portal, along with HTML and CSS for the UI, while another couple of Python libraries, Bokeh and Pandas are used to plot and render the interactive graphs from which the user can infer a lot of information such as rate of occurrences of motion, average duration of motions, etc.

V. Experiment Results and Analysis

The system is developed to detect motion in real-time and respond to significant motions appropriately by notifying the user immediately about this anomaly and logging important information such as start and end of detected motion times. It also uploads and stores the image frame in which motion was detected.

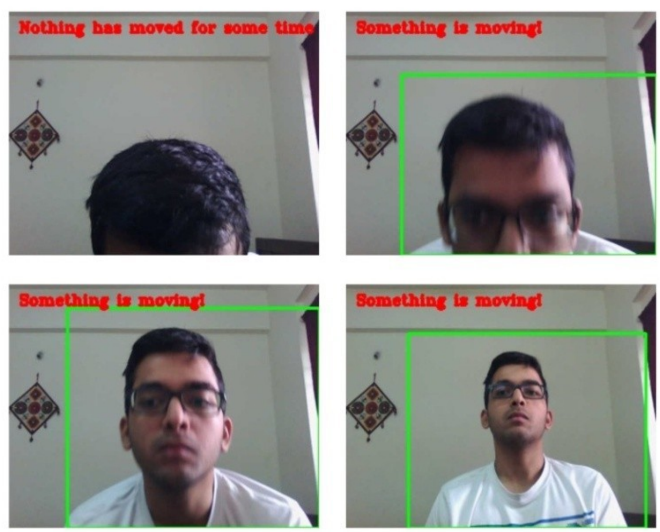

Fig 6. Motion with single entity

Results of earlier experiments with movement caused by a single entity are as shown in the collection of figures in Fig. 6 which depicts the entry of an entity from the bottom of the frame and gradually moving upwards. The region of detection of the motion has been highlighted by a bounded green rectangle.

On detection of the motion, the image is immediately uploaded to Google Drive. After successful upload of the file, the image URL or link to the image corresponding to the motion is extracted.

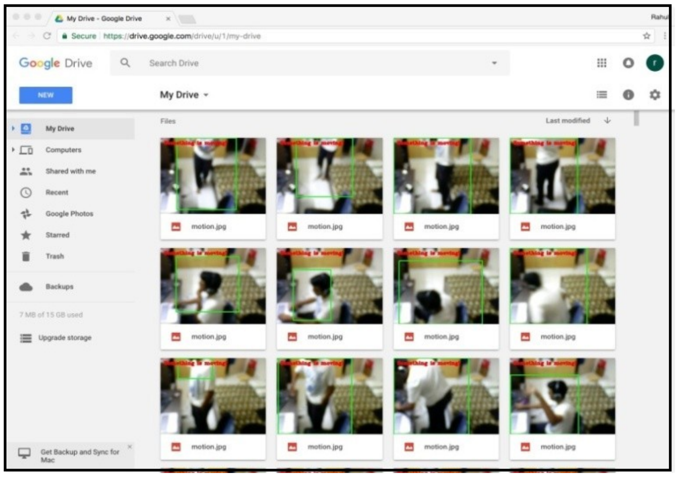

Fig. 7. Google drive with uploaded images

The URL of the image in the drive for the current motion is extracted and is sent along with start time of the motion in the body of the e-mail. The user receives an email notification on any device which has the user"s email account as a connected account.

The e-mail received by the user contains time of start of motion as well as image captured by camera as shown in Fig. 8.

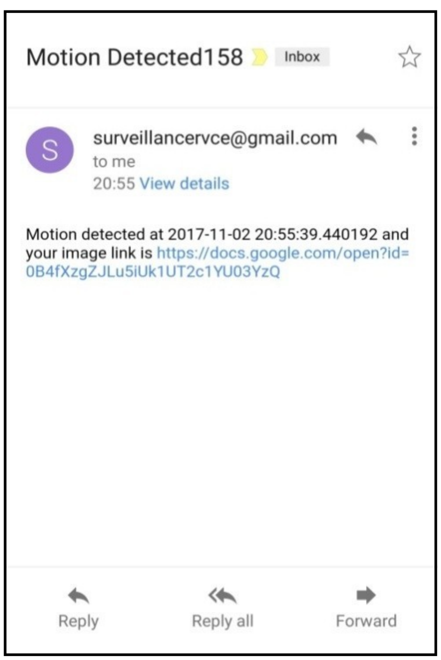

Fig. 8. E-mail Notification received by User

At the end of the motion, the end time of the motion is also logged into the RDS as it helps in identifying various important factors in the future.

To make the system wholesome, it was essential to have further interaction between the user and the system. For the same, a web application was developed that allows authorized users to view information limited to just their surveillance system.

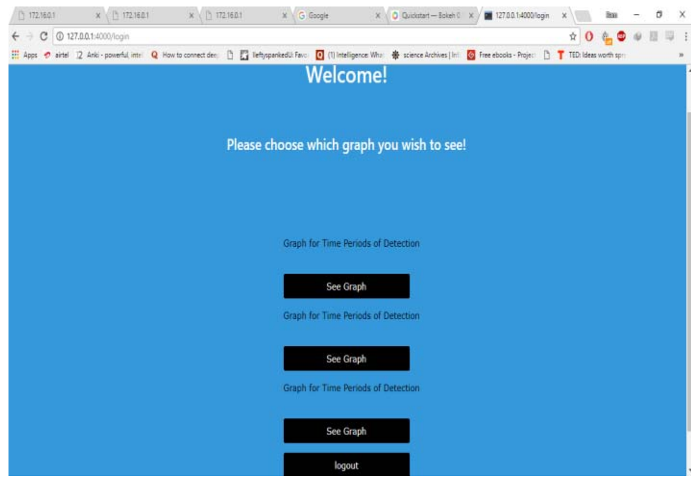

Fig. 9. Application home page

From the home page the user has options to view various graphs that have been plotted using data extracted from RDS. The graphs and data visualized here are unique to each user and their system.

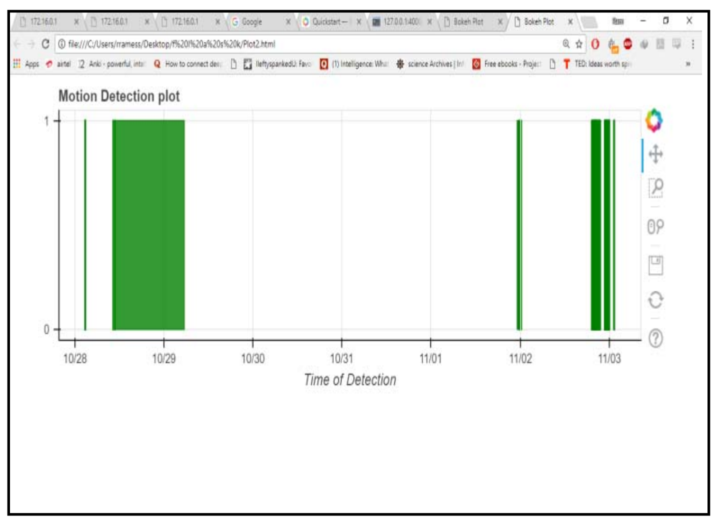

Fig. 10. Time Duration graph – 1

On clicking any of the "See Graph" buttons, the application will render different html pages which contain different types of graphs. It comes with several interactive operations such as pan, zoom, save, etc. which can be triggered by choosing one of the many options listed as icons on the edges of the plot.

Fig. 11. Time Duration graph – 2

On choosing which graph the user want to see, Flask renders the html template on which Bokeh produces the output file as a graph. These graphs are interactive and aid the user in visualizing and analyzing the data.

The system was later tested in a different environment where multiple entities were causing motion simultaneously. The system continued to perform accurately by detecting the motion as well as the causes and responding to this change of state, instantly and accurately as shown in Fig. 12 below.

Fig. 12. System Detects Multiple Motions and entities

Internet of Things expects the processes within such systems to occur in real time. Hence, the system was tested on nearly 5 hours of surveillance video directly from the USB camera. 500 entries of times taken for various sub-processes were logged into a CSV file as the script was running. The durations for execution of each of these processes were averaged to give us an understanding of latency between various processes that can be expected in such a system.

I. SUB-PROCESS DURATION ANALYSIS

	Write Image on R Pi	Upload image to Drive	Upload time to RDS	Trigger sending email
Average Duration (sec)	0.0123	4.0641	0.5177	1.7907
Standar d Deviation (sec)	0.0023	1.6671	0.0561	0.3986

Table I: Values are rounded off to 4 decimal places

II. SYSTEM LATENCY ANALYSIS

	Time it takes for user to get notified of detected motion	Total Process
Average Duration (sec)	5.8731	6.8818
Standar d Deviation (sec)	1.6672	1.5193

Table II: Values are rounded off to 4 decimal places

The Total process is the duration to execute all the tasks, namely writing the image to memory, uploading image to Google Drive, uploading information such as entry time and image URL to RDS, triggering sending of e-mail and finally uploading end time of the motion into RDS.

The average latency between detection of motion and notification to the user has been calculated as approximately 5.9 seconds. This value can be easily reduced to a much smaller value of 1.8 seconds by to not uploading the captured image to Google Drive before sending the e-mail. Although, the system was made with the aim that the user should receive the notification for a certain motion in a single e-mail rather than sending multiple notifications, which would be impractical. This decision was made because a near 4.1 second delay would not induce any severe consequences for a video surveillance application, plus the user gets to view the captured image by clicking the image URL added in the e-mail.

Standard deviation of durations of each of these tasks has been calculated as it is a reliable way to measure how the response times and latencies of the system vary.

978-1-5386-2044-1/17/$31.00 ©2017 IEEE

VI. CONCLUSION

Any room or section of one"s home can easily be put under effective surveillance using the developed system. Upon the detection of foreign motion, the user is instantly notified along with the required visuals he/she requires to ascertain whether the motion is that of an infiltrator or harmless, leaving the due course of action to the user"s discretion. By virtue of the Internet of Things concepts implemented while developing the system, the user can receive minimal-latency updates in any location as long as there exists an Internet connection. Similarly, the user is also able to access the stored data from any remote location on a computer or mobile device, provided the correct login credentials are entered.

VII. FUTURE WORK

Wide applications of Motion Detection Systems have surfaced itself in the last few years from tracking to object classification. A system that uses a deep learning model to classify the objects into simple things like "person", "car", and "door" has been successfully tested on our computers that possess much better performance specifications than a Raspberry Pi. This method had consumed a lot of processing time leading to a lot of delay, mainly because of lower RAM and CPU processing capability of the Raspberry Pi (1GB).

In future, the endeavor is to write a more optimized and less memory hungry algorithm for a smarter detection and could also possibly waver off some training datasets that compromises little on performance.

VIII. REFERENCES

[1] Aamir Nizam Ansari \ Mohamed Sedkyl, Neelam Sharma, Anurag Tyagil, „An Internet of Things Approach for Motion Detection using Raspberry Pi" in 2015 International Conference on Intelligent Computing and Internet of Things, pp. 131-134

[2] Huu-Quoc Nguyen, Ton Thi Kim Loan, Bui Dinh Mao and Eui-Nam Huh, „Low Cost Real-Time System Monitoring Using Raspberry Pi" in ICUFN 2015, pp. 857-859

[3] Virginia Menezes, Vamsikrishna Patchava, M. Surya Deekshith Gupta, „Surveillance and Monitoring System Using Raspberry Pi and SimpleCV" during 2015 International Conference on Green Computing and Internet of Things, pp. 1276-1278

[4] M. Chen, X.L. Xu, „An Improved Method for Motion Detection by Image Difference" in Information Science and Control Engineering 2012, 7-9 Dec

[5] Shih-Chia Huang, „An Advanced Motion Detection Algorithm with Video Quality Analysis for Video Surveillance Systems" in IEEE Transactions on Circuits and Systems for Video Technology, 18 Oct, 2010.