**What is Image Processing?**

It is important to know what exactly image processing is and what is its role in the bigger picture before diving into its how's. Image Processing is most commonly termed as 'Digital Image Processing' and the domain in which it is frequently used is 'Computer Vision'. Don't be confused - we are going to talk about both of these terms and how they connect. Both Image Processing algorithms and Computer Vision (CV) algorithms take an image as input; however, in image processing, the output is also an *image*, whereas in computer vision the output can be some *features/information about* the image.

**Why do we need it?**

The data that we collect or generate is mostly raw data, i.e. it is not fit to be used in applications directly due to a number of possible reasons. Therefore, we need to analyze it first, perform the necessary pre-processing, and then use it.

For instance, let's assume that we were trying to build a cat classifier. Our program would take an image as input and then tell us whether the image contains a cat or not. The first step for building this classifier would be to collect hundreds of cat pictures. One common issue is that all the pictures we have scraped would not be of the same size/dimensions, so before feeding them to the model for training, we would need to resize/pre-process them all to a standard size.

This is just one of many reasons why image processing is essential to any computer vision application.

**Prerequisites**

Before going any further, let's discuss what you need to know in order to follow this tutorial with ease. Firstly, you should have some basic programming knowledge in any language. Secondly, you should know what machine learning is and the basics of how it works, as we will be using some machine learning algorithms for image processing in this article. As a bonus, it would help if you have had any exposure to, or basic knowledge of, Open CV before going on with this tutorial. But this is not required.

One thing you should definitely know in order to follow this tutorial is how exactly an image is represented in memory. Each image is represented by a set of pixels i.e. a matrix of pixel values. For a grayscale image, the pixel values range from 0 to 255 and they represent the intensity of that pixel. For instance, if you have an image of 20 x 20 dimensions, it would be represented by a matrix of 20x20 (a total of 400-pixel values).

If you are dealing with a colored image, you should know that it would have three channels - Red, Green, and Blue (RGB). Therefore, there would be three such matrices for a single image.

**Installation**

Note: Since we are going to use OpenCV via Python, it is an implicit requirement that you already have Python (version 3) already installed on your workstation.

$ pip install opencv-python

To check if your installation was successful or not, run the following command in either a Python shell or your command prompt:

import cv2

**Some Basics You Should Know**

Before we move on to using Image Processing in an application, it is important to get an idea of what kind of operations fall into this category, and how to do those operations. These operations, along with others, would be used later on in our applications. So, let's get to it.

For this article we'll be using the following image:



**Note**: The image has been scaled for the sake of displaying it in this article, but the original size we are using is about 1180x786.

You probably noticed that the image is currently colored, which means it is represented by three color channels i.e. Red, Green, and Blue. We will be converting the image to grayscale, as well as splitting the image into its individual channels using the code below.

**Finding Image Details**

After loading the image with the imread() function, we can then retrieve some simple properties about it, like the number of pixels and dimensions:

**Problem Statement:**

Confidential Data Security System that detects anysuspicious activity and sends an email and sms alert of the data theft to the admin.

**Hardware Components :**

Raspberry Pi model 4 **P**IR sensor

Pi Camera

Breadboard

LED and resistors connecting wires

**Software Components**:

Raspbian OS



Python libraries

Internet connection

**Theory:**

**1.Introduction**

In surveillance, CCTV camera is costly because of the use of computer. It reserves too much space for continues recording and also require manpower to detect the unauthorized Activity. But compared to the existing system Raspberry pi system is much cheaper with better resolution and low power consumption feature. Here infrared (IR) sensors are used as a simple but powerful people presence triggers. This system is suitable for small confidential data area surveillance.

i.e. personal office cabin, bank locker room, home. Whenever the motion is detected through PIR sensor inside the room the image is captured through camera and temporarily stored in the raspberry pi module. Internet of things based application can be used remotely to view the activity and get notifications when motion is detected. System works standalone without the PC once programmed.

**1.1 Motivation**

The use of M2M (machine to machine) communication is an advantage over the traditional Data Acquisition System (DAS)as the monitoring and controlling can be done without human intervention . As the system becomes fully automatic so the amount of error decreases and the efficiency of the system increases drastically.

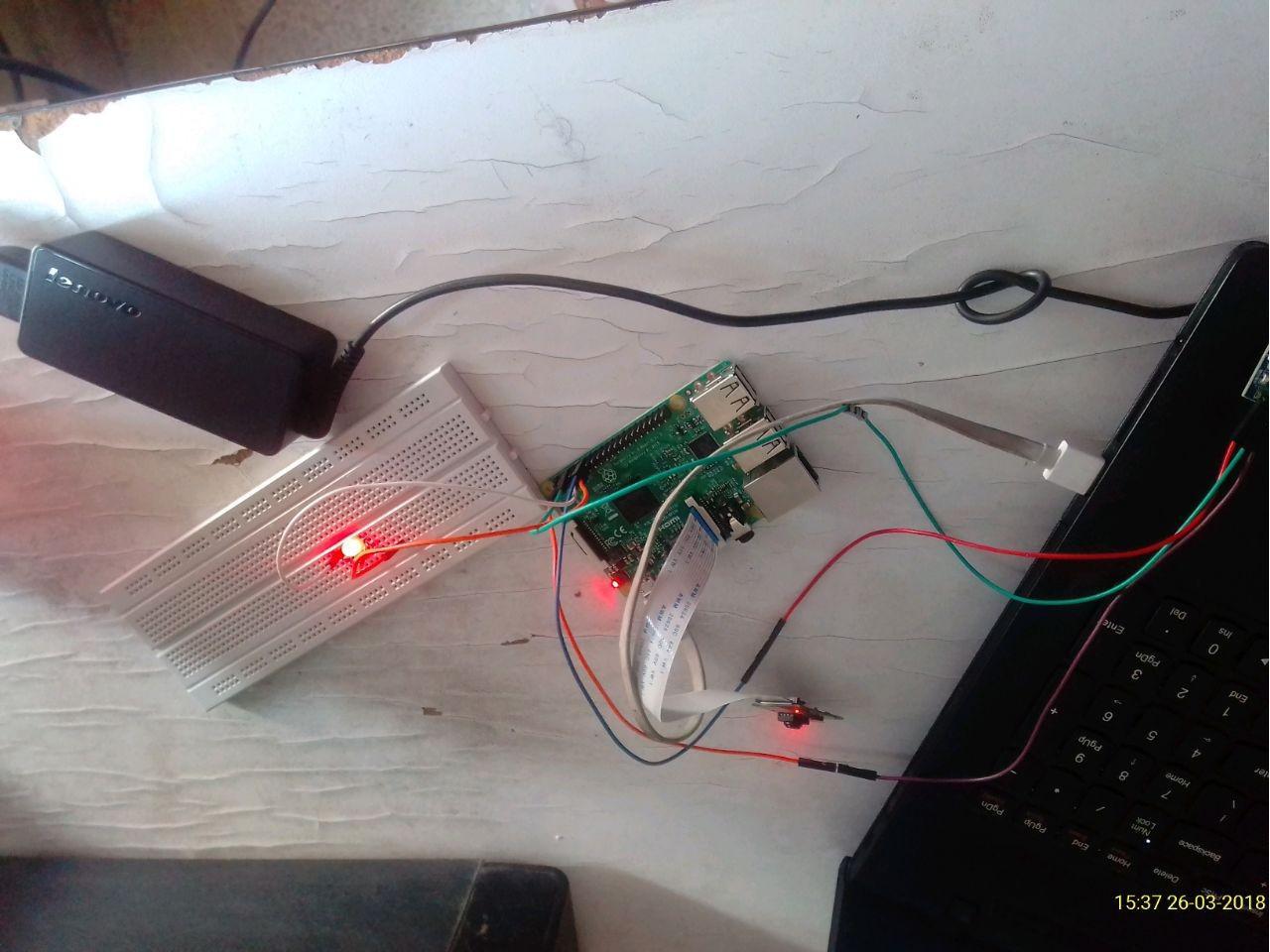
**1.2 Advantages**

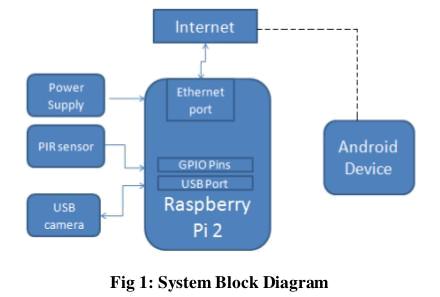
It offers privacy on both sides since it is being viewed by only one person. It is a simple circuit. The operating system used here is Raspbian OS. Just because all of those weak points of the surveillance system, an energy efficient portable system is proposed, that can take pictures when the burglary happens and send out an alert signal at the same time is much better than the current in use surveillance systems. It is simple to implement, small size portable stand-alone device with its own power source, energy capable with instantaneous alert, truly cheap for residential and personal use.

**2. PROPOSED WORK**

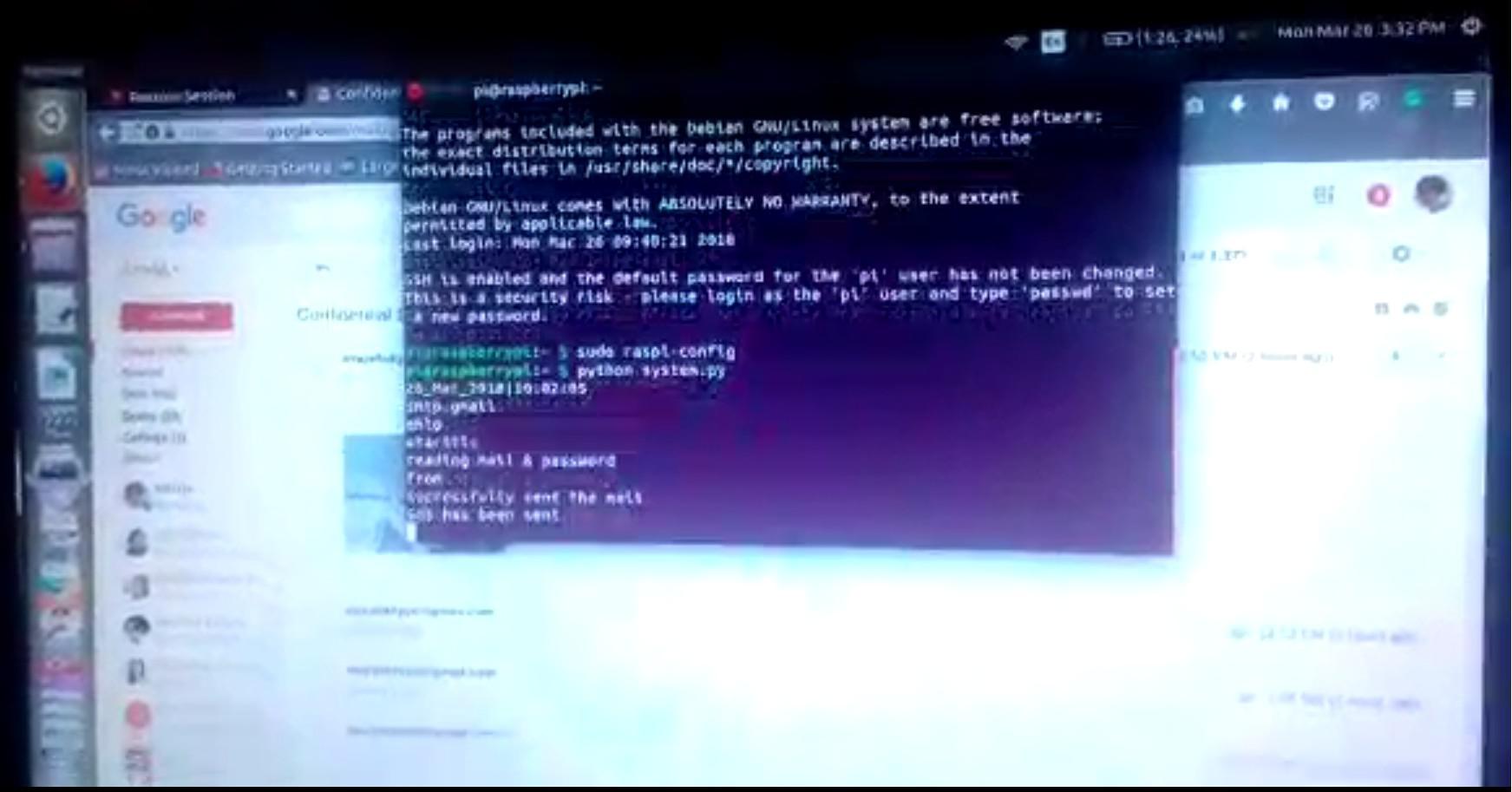
The aim is to make a smart data security system which can be monitored by owner remotely. As it is connected with the system with IOT, system will send the email notification and sms to android device when an data theft is detected inside the room. It is required to develop and implement and affordable low cost web-camera based surveillance system for remote security monitoring.Authorized user can access to their monitoring system remotely via internet . This entire work is done on raspberry pi with Raspbian operating system ported on it.

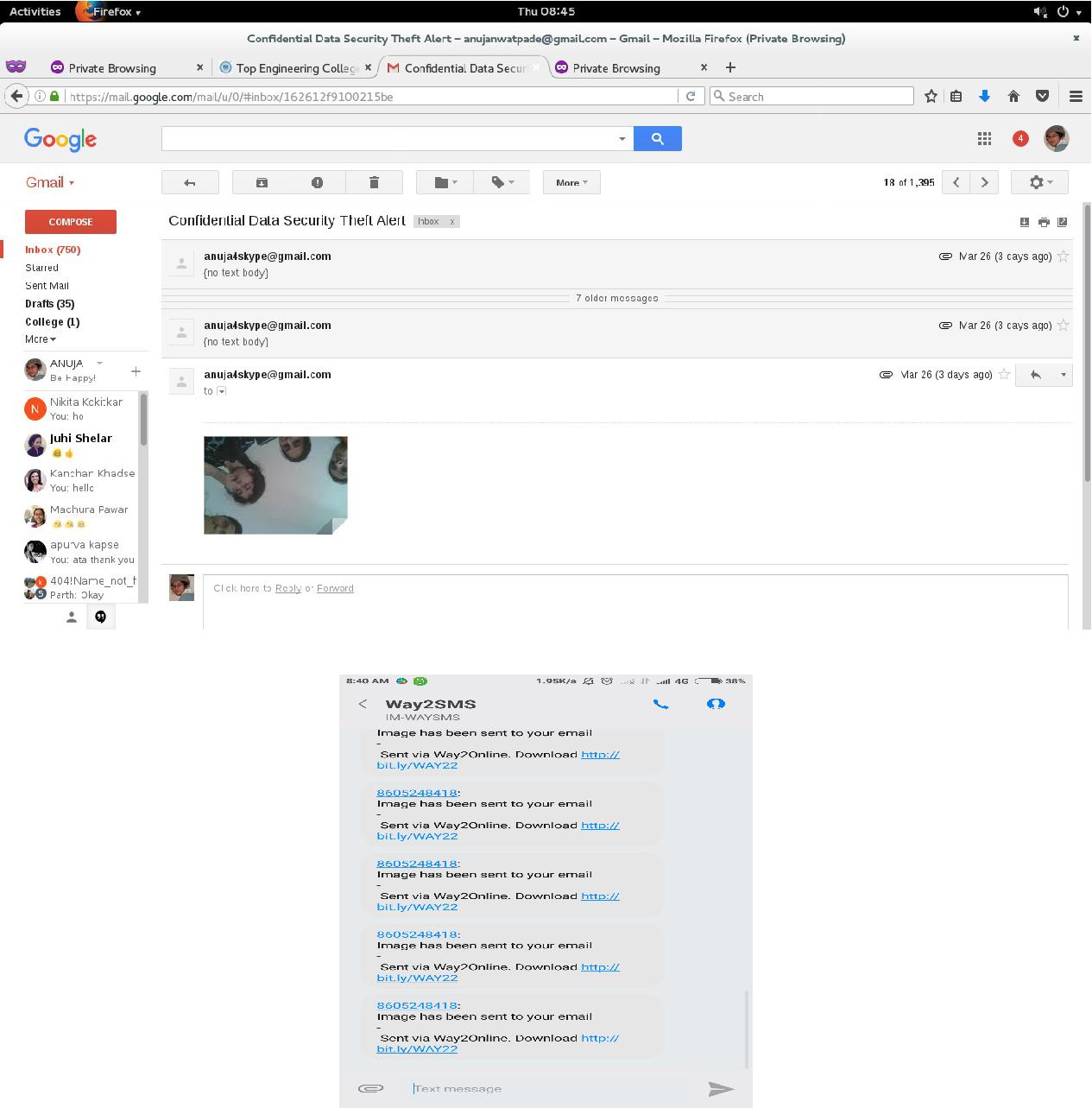
**2.1 Connection**





**2.2 OUTPUT**





**CONCLUSION:**

Thus we have successfully implemented the smart surveillance system in such a way that it can fulfill the needs of the user for particular surveillance area. It has countless applications and can be used in different environments and scenarios. User can view captured image remotely. On Future expansion live video streaming can be added as per the user requirement.

**Teaching The Camera**

First create an ML model having datasets of normal activities such as walking, talking, reading, sitting etc. Then feed the datasets of Suspicious Activity such as fighting, boxing, pointing guns or any other violent movement deemed suspicious into the ML model.

Also, perform such activities in front of the smart camera so that various movements are captured. This will be useful for training the ML model and deploying it on Raspberry Pi to make a smart AI camera.

To create and train an ML model, there are several flexible options such as TensorFlow, Google Teachable, Edge Impulse, Lobe etc. You can choose any and work on the above idea. Here, I am demonstrating with the help of Google Teachable.

In Google Teachable, select the PoseNet option for tracking the various body movements and actions. Earlier, we performed different actions like walking, talking, eating, standing etc. By correctly labelling them, feed these datasets into the ML model. Similarly, feed the datasets of activities like pointing guns, firing guns, fighting, beating etc.