

# **Estimating Crowd Numbers with OpenCV and Raspberry Pi**

## **Abstract**

In the era of advanced technologies, digital image processing has rapidly become a crucial component of various electronic devices, including smartphones, security cameras, and laptops.

This project aims to develop a crowd counting system using OpenCV on a Raspberry Pi integrated with ThingSpeak for remote monitoring. The system employs a Pi Camera Module V2 to continuously capture frames, which are then processed using the Histogram of Oriented Gradients (HOG) descriptor for object detection. These processed frames are compared against OpenCV's pre-trained models to detect and count people within the image. The resulting count is transmitted to a ThingSpeak channel, enabling real-time monitoring from any location globally.

## **Components Needed**

### **Hardware:**

- Raspberry Pi 4 Model B (or any compatible version)
- Raspberry Pi Camera Module V2

### **Software and Online Services:**

- ThingSpeak IoT Platform
- Python 3.0
- OpenCV 3.0 Library

## **Installing OpenCV on Raspberry Pi**

For this project, we will utilize the OpenCV library to perform crowd detection. To install OpenCV on your Raspberry Pi, please follow these steps:

### **1. Update the Package List**

Begin by updating your Raspberry Pi's package list to ensure all packages are up to date:

```
sudo apt-get update
```

### **2. Install Required Dependencies**

Install the necessary dependencies that OpenCV relies on:

```
sudo apt-get install libhdf5-dev -y
```

```
sudo apt-get install libhdf5-serial-dev -y
```

```
sudo apt-get install libatlas-base-dev -y
```

```
sudo apt-get install libjasper-dev -y
```

```
sudo apt-get install libqtgui4 -y
```

```
sudo apt-get install libqt4-test -y
```

### **3. Install OpenCV**

With the dependencies installed, you can now install OpenCV using the following command:

```
pip3 install opencv-contrib-python
```

## **Installing Additional Required Packages**

Before proceeding to program the Raspberry Pi for crowd counting, we need to install some additional packages that are crucial for our project's functionality.

## 1. Installing imutils

The imutils library offers a set of convenience functions that simplify basic image processing tasks when working with OpenCV. These tasks include translation, rotation, resizing, skeletonization, and more. It also makes it easier to display images using Matplotlib.

To install imutils, execute the following command:

```
pip3 install imutils
```

## 2. Installing Matplotlib

The matplotlib library is a comprehensive tool for creating static, animated, and interactive visualizations in Python. It will be instrumental in visualizing data and results during the development and testing phases of the project.

Install matplotlib by running:

```
pip3 install matplotlib
```

## Setting Up ThingSpeak for Crowd Counting

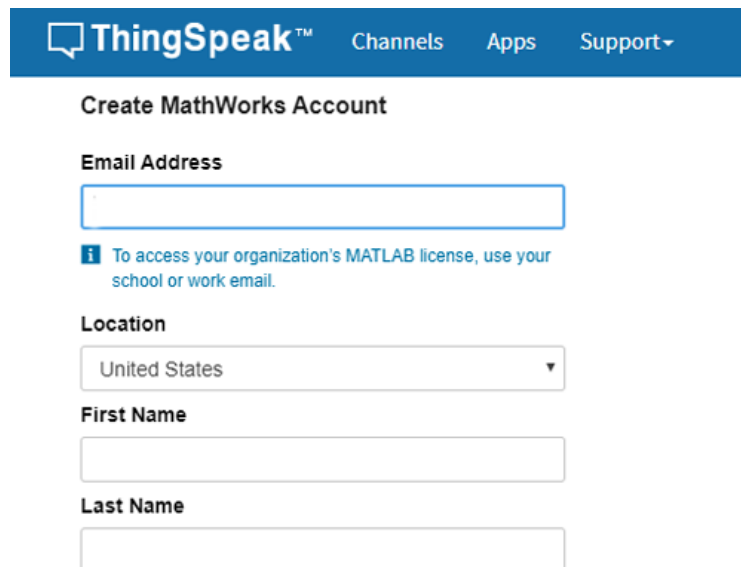
ThingSpeak is a popular Internet of Things (IoT) platform that enables remote monitoring and control of data over the internet. By leveraging ThingSpeak, we can access our data from anywhere in the world and manage systems using the channels and web interfaces it provides.



To set up ThingSpeak for people counting, follow these steps:

## 1. Create an Account or Sign In

- Visit the [ThingSpeak](https://thingspeak.com) website and click on Sign Up to create a new account.
- Fill in your personal details as prompted.
- After registration, verify your email address by clicking the link sent to your inbox.
- If you already have an account, simply sign in with your existing credentials.



The screenshot shows the 'Create MathWorks Account' page on the ThingSpeak website. The page has a blue header with the ThingSpeak logo and navigation links for 'Channels', 'Apps', and 'Support'. The main content area is white and contains the following fields and instructions:

- Create MathWorks Account**
- Email Address**: A text input field.
- Location**: A dropdown menu currently showing 'United States'.
- First Name**: A text input field.
- Last Name**: A text input field.

Below the email address field, there is a blue information icon and the text: 'To access your organization's MATLAB license, use your school or work email.'

## 2. Create a New Channel

- Once logged in, navigate to your account dashboard.
- Click on the **"New Channel"** button to create a new data channel.
- In the channel setup page, provide a **Name** and **Description** for your data stream. For instance, you might name it **"People Counting Data"**.
- Under the **Fields** section, create a field named **"People"** to store the count data. You can add multiple fields if needed for additional parameters.

## New Channel

**Name**

**Description**

**Field 1**  ☒

**Field 2**  ☐

### 3. Save the Channel

- After entering all the necessary information, click on the "**Save Channel**" button to save your new channel.

### 4. Retrieve API Key and Channel ID

- To send data from your Python script to ThingSpeak, you'll need the channel's **API Key** and **Channel ID**.
- Navigate to the **API Keys** tab within your channel settings.
- Copy the **Write API Key** and **Channel ID**; you'll need to insert these into your Python script to authenticate and direct data to the correct channel.

## Crowd Estimation

Channel ID: 812060  
Author: choudharys04  
Access: Private

[Private View](#) [Public View](#) [Channel Settings](#) [Sharing](#) [API Keys](#)

## Write API Key

Key

[Generate New Write API Key](#)

## Hardware Setup

Here we only require Raspberry Pi and Pi Camera V2 for this **OpenCV human counting project** and you just need to attach the camera ribbon connector in the camera slot given in the Raspberry pi



The Pi 4 Camera board is a purpose-built expansion board for the Raspberry Pi computer. The Raspberry Pi hardware is connected via a specialized CSI interface. In its native still-capture mode, the sensor's resolution is 5 megapixels. Capturing at up to 1080p and 30 frames/second in video mode is possible. Because of its portability and compact size, this camera module is fantastic for handheld applications.

## Python Program Explanation for People Counting

The complete Python code for our crowd counting project using OpenCV is provided at the end of this document. In this section, we will delve into the key components of the code to enhance understanding.

### Importing Required Libraries

At the beginning of the script, we import all the necessary libraries that facilitate image processing, object detection, and data transmission:

```
import cv2

import imutils

from imutils.object_detection import non_max_suppression

import numpy as np

import requests

import time

import base64

from matplotlib import pyplot as plt

from urllib.request import urlopen
```

### Explanation:

- **Imutils:**

For use with OpenCV and either version of Python, this package provides a set of helper functions for everyday image processing tasks such as scaling,

cropping, skeletonizing, showing Matplotlib pictures, grouping contours, identifying edges, and more.

- **Numpy:**

You can manipulate arrays in Python with the help of the NumPy library. Matrix operations, the Fourier transform, and linear algebra are all within their purview. Because it is freely available to the public, anyone can use it. That's why it's called "Numerical Python," or "NumPy" for short.

Python's list data structure can replace arrays, but it could be faster. NumPy's intended benefit is an array object up to 50 times quicker than standard Python lists. To make working with NumPy's array object, ndarray, as simple as possible, the library provides several helpful utilities. Data science makes heavy use of arrays because of the importance placed on speed and efficiency.

- **Requests:**

You should use the requests package if you need to send an HTTP request from Python. It hides the difficulties of requests making behind a lovely, straightforward API, freeing you to focus on the application's interactions with services and data consumption.

- **Time:**

In Python, the time module has a built-in method called local time that may be used to determine the current time in a given location depending on the time in seconds that have passed since the epoch (). tm.isdst will range from 0 to 1 to indicate whether or not daylight saving time applies to the current time in the region.



- **Base64:**

If you need to store or transmit binary data over a medium better suited for text, you should look into using a Base64 encoding technique. There is less risk of data corruption or loss thanks to this encoding method. Base64 is widely used for many purposes, such as MIME-enabled email storing complicated data in XML and JSON.

- **Matplotlib:**

When it comes to Python visualizations, Matplotlib is your one-stop shop for everything from static to animated to interactive. Matplotlib facilitates both straightforward and challenging tasks. Design graphs worthy of publication. Create movable, updatable, and zoomable figures.

- **urllib.request:**

If you need to make HTTP requests with Python, you may be directed to the brilliant requests library. Though it's a great library, you may have noticed that it needs to be a built-in part of Python. If you prefer, for whatever reason, to limit your dependencies and stick to standard-library Python, then you can reach for urllib.request!

## **Configuring ThingSpeak Channel**

After importing the required libraries, we need to set up the ThingSpeak channel by entering the channel ID and the Write API key that you obtained earlier. These

credentials are essential for authenticating and sending data to your ThingSpeak channel:

```
channel_id = 812060 # PUT CHANNEL ID HERE
```

```
WRITE_API = 'X5AQ3EGIKMBYW31H' # PUT YOUR WRITE KEY HERE
```

```
BASE_URL = "https://api.thingspeak.com/update?api_key= {}".format(WRITE_API)
```

## **Initialize the HOG**

Now, initialize the HOG (Histogram Oriented Object descriptor). HOG is one of the most popular techniques for object detection and has been used in several applications. *cv2.HOGDescriptor\_getDefaultPeopleDetector()* used to call a pre-trained model of OpenCV for people detection

```
hog = cv2.HOGDescriptor()
```

```
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

## **Defining the Detector Function**

Inside the *detector()*, Pi receives an RGB image split into three colour channels. After it, it resizes the image using *imutils*. Then it calls the *detectMultiScale()* method to analyze the image to know if a person exists using the classification result from the SVM model.

```
def detector(image):
```

```
    image = imutils.resize(image, width=min(400, image.shape[1]))
```

```
    clone = image.copy()
```

```
    rects, weights = hog.detectMultiScale(image, winStride=(4, 4), padding=(8, 8), scale=1.05)
```

## Applying Non-Maximum Suppression

If you're getting false positive results or detection failures due to capture-box overlap, try running the below code, which uses non-max suppressing capability from imutils to activate overlapping regions.

```
for (x, y, w, h) in rects:

    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)

rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])

result = non_max_suppression(rects, probs=None, overlapThresh=0.7)

return result
```

## Capturing and Processing Video Frames

With the help of OpenCV's VideoCapture() method, the image is retrieved from the Pi camera within the record() function, where it is resized with the imutils before being sent to ThingSpeak.

```
def record(sample_time=5):

    camera = cv2.VideoCapture(0) #if it doesn't work, try 'camera = cv2.VideoCapture(0, cv2.CAP_V4L2)'

    frame = imutils.resize(frame, width=min(400, frame.shape[1]))

    result = detector(frame.copy())

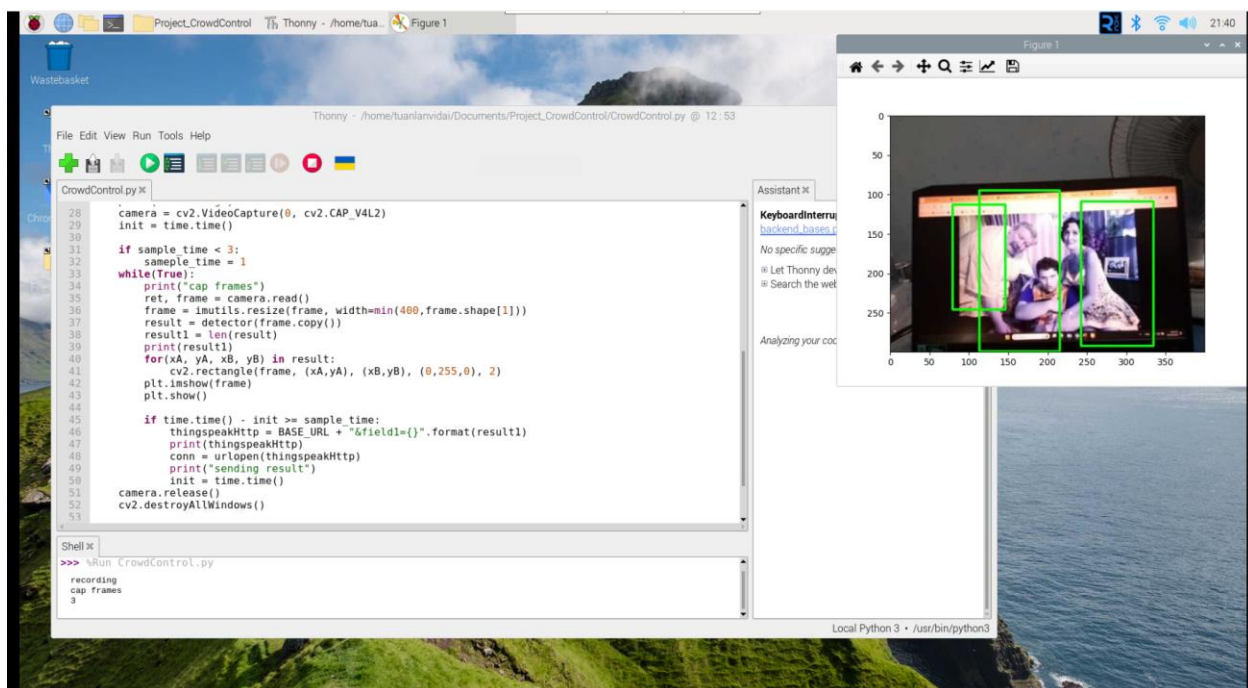
    thingspeakHttp = BASE_URL + "&field1={}".format(result1)
```

## Testing the OpenCV People Counter

Launch the program by extracting it to a new folder. You'll need to give Python a few seconds to load all the necessary modules. Start the program. A new window will pop up, showing the camera's output after a few seconds. Make sure your Raspberry Pi camera is operational before running the python script. The following command is used to activate the python script after a review of the camera has been completed:

```
pi@raspberrypi:~ $ python3 thing.py
recording
cap frames
3
https://api.thingspeak.com/update?api_key=X5AQ3EGIKMBYW31H&field1=3
sending result
```

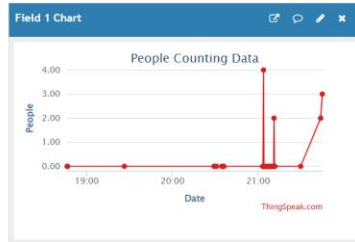
Then you will find a window popping up with your video feed in it. Pi will take the first frame and process it using the OpenCV to detect the number of peoples. If it detects the people, you will find a box around it like this:



Now check your ThingSpeak channel, where you can monitor the crowd size from anywhere in the world.

#### Channel Stats

Created: about 4 hours ago  
Last entry: about a minute ago  
Entries: 40



Code:

```
import cv2

import imutils

from imutils.object_detection import non_max_suppression

import numpy as np

import requests

import time

import base64

from matplotlib import pyplot as plt

from urllib.request import urlopen

channel_id = 2672477 #change to your id

WRITE_API = 'DICFWQVLEBQHZZTK' #Change to your api

BASE_URL = "https://api.thingspeak.com/update?api_key={}".format(WRITE_API)

hog=cv2.HOGDescriptor()

hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

**def detector(image):**

**image = imutils.resize(image, width=min(400, image.shape[1]))**

**clone = image.copy()**

**rects, weights = hog.detectMultiScale(image, winStride=(4,4), padding=(8,8), scale=1.05)**

**for (x,y,w,h) in rects:**

**cv2.rectangle(image, (x,y), (x+w, y+h), (0, 0, 255), 2)**

**rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])**

**result = non\_max\_suppression(rects, probs=None, overlapThresh = 0.7)**

**return result**

**def record(sample\_time=5):**

**print("recording")**

**camera = cv2.VideoCapture(0, cv2.CAP\_V4L2)**

**init = time.time()**

**if sample\_time < 3:**

**sameple\_time = 1**

**while(True):**

**print("cap frames")**

**ret, frame = camera.read()**

**frame = imutils.resize(frame, width=min(400,frame.shape[1]))**

**result = detector(frame.copy())**

```
result1 = len(result)

print(result1)

for(xA, yA, xB, yB) in result:

    cv2.rectangle(frame, (xA,yA), (xB,yB), (0,255,0), 2)

plt.imshow(frame)

plt.show()
```

```
if time.time() - init >= sample_time:

    thingspeakHttp = BASE_URL + "&field1={}".format(result1)

    print(thingspeakHttp)

    conn = urlopen(thingspeakHttp)

    print("sending result")

    init = time.time()

camera.release()

cv2.destroyAllWindows()
```

```
def main():

    record()

if __name__ == '__main__':

    main()
```

## References

- [1] OpenCV Team, "OpenCV-Python Tutorials," [Online]. Available: [https://docs.opencv.org/4.x/d6/d00/tutorial\\_py\\_root.html](https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html).
- [2] Rosebrock, A, " imutils: Image Processing Made Convenient. GitHub repository.," [Online]. Available: <https://github.com/jrosebr1/imutils>.
- [3] MathWorks, "ThingSpeak Documentation," [Online]. Available: <https://www.mathworks.com/help/thingspeak/>.
- [4] R. P. Foundation, "Camera Module," [Online]. Available: <https://www.raspberrypi.com/documentation/accessories/camera.html>.
- [5] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, 2007.
- [6] K. Reitz and O. Chavent, "Requests: HTTP for Humans," [Online]. Available: <https://docs.python-requests.org/en/master/>.