# Workshop Day 2: Raspberry Pi with Camera

In this workshop, we will be using Raspberry Pi with camera to perform data collection and processing on the edge and then send the processed image to edge device. OpenCV library will be used in this workshop.

**OpenCV** is an open source computer vision and machine learning software library widely used in vision application. It has over 2,500 optimized algorithms such as facial recognition, object identification, human action classification, camera movement tracking and etc. It was written in C++ and the wrapper for Python, Java and MATLAB are available. It can run in Windows, Linux, Android and Mac OS.

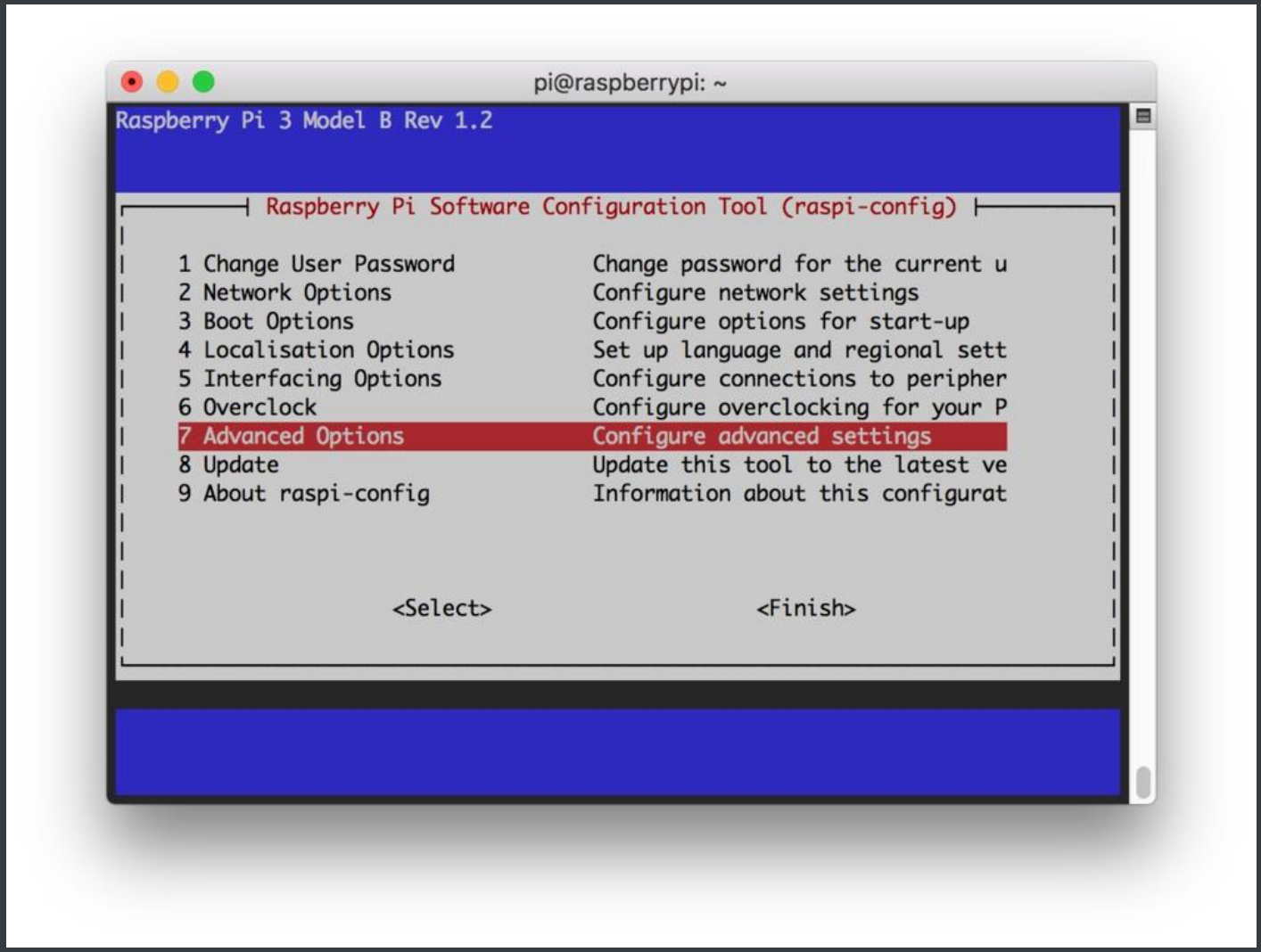
This workshop covers:

- Reading images from camera
- Resizing the images
- Retrieving the required channel
- Filtering and Edge Detection of images
- Region of Interest (ROI) detection
- Sending processed image to edge server

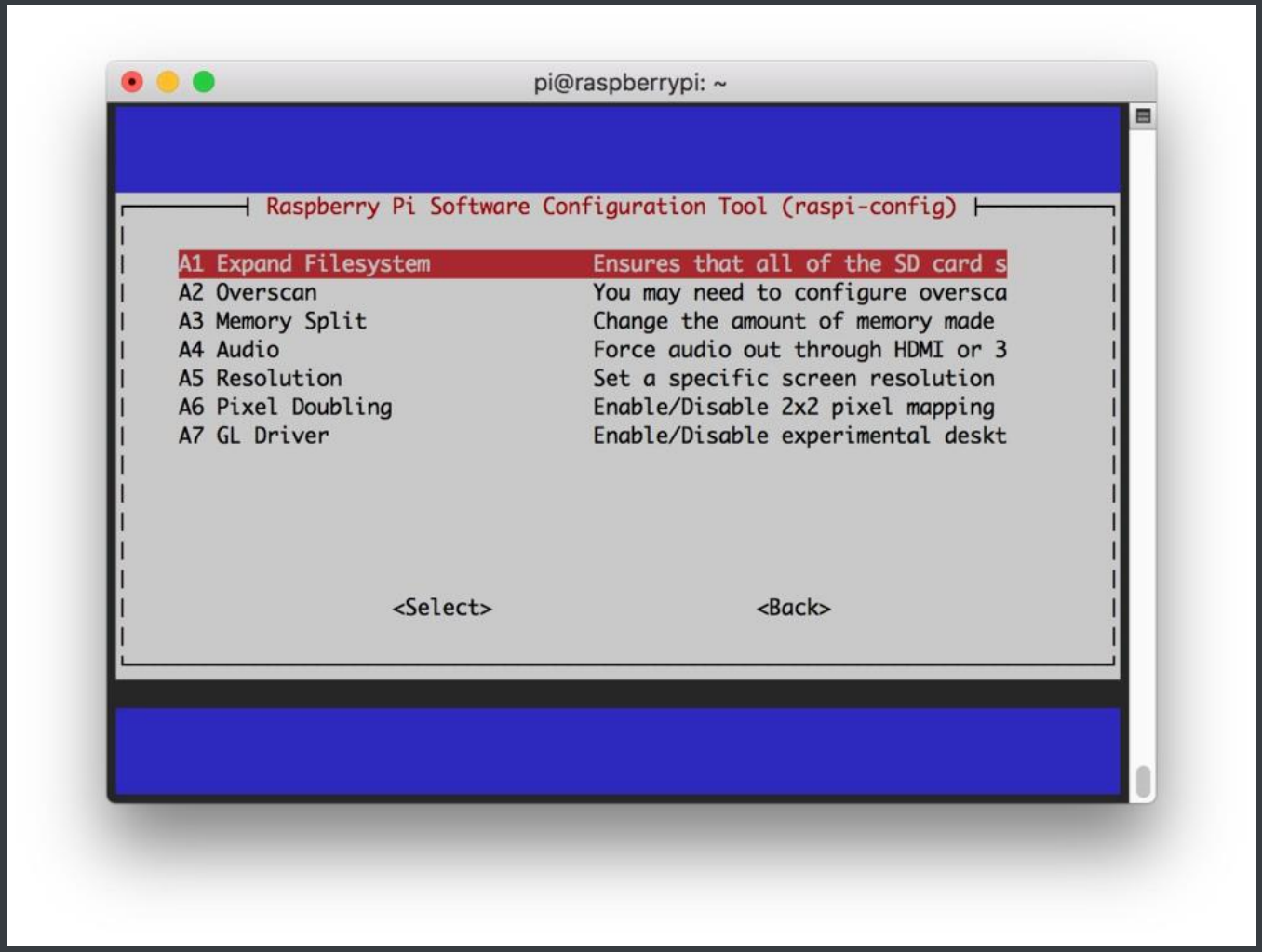
### 1.1 Installation of OpenCV in Rapsberry Pi

1. To make sure the file system on your Raspberry Pi to include all available space on your micro-sd card, you have to enter following command and select the "Advanced Options" menu item:

```
sudo raspi-config
```




2. Followed by choosing the "Expand filesystem"




3. Reboot the system with following command:

```
sudo reboot now
```

4. After rebooting the system, launch the terminal and key in the following command to update the system. It might take some time.

```
sudo apt -y update && sudo apt -y upgrade
```

5. Install the dependencies:

```
sudo apt-get install libhdf5-dev libhdf5-serial-dev libhdf5-103
sudo apt-get install libqtgui4 libqtwebkit4 libqt4-test
sudo apt-get install libatlas-base-dev
sudo apt-get install libjasper-dev
```

6. Install the pip package installer (You can skip this step if you have installed pip package installer previously):

```
wget  https://bootstrap.pypa.io/get-pip.py
sudo python3 get-pip.py
```

7. Install the OpenCV package (It may take 5-15mins):

```
sudo pip3 install opencv-python
```

8. OpenCV library has installed successfully, you may proceed to next step.

### 1.2 Reading images from camera

Following is the code snippet for reading the video by using OpenCV.

```
import cv2

# To capture video.
# Device index is set to 0 as we only have one camera
cap = cv2.VideoCapture(0)

while(True):
  # If frame is read correctly, ret will be True, vice versa
  ret, frame = cap.read()

  # To display the video in new window and named it as "Original"
  cv2.imshow('Original', frame)

  # Press 'q' to quit the program
  if cv2.waitKey(1) & 0xff == ord('q'):
    break

cap.release()
```

```
    cv2.destroyAllWindows()
```

Did you notice that, in vision system, the captured video is processed frame by frame? Can you imagine if all the frames are required to send to cloud services for processing, the bandwidth cost will be tremendous and latency will be high?

### 1.3 Resizing the Image

Following is the code snippet for resizing the image:

```
    import cv2

    # To capture video.
    # Device index is set to 0 as we only have one camera
    cap = cv2.VideoCapture(0)

    while(True):
      # If frame is read correctly, ret will be True, vice versa
      ret, frame = cap.read()

      # To display the video in new window and named it as "Original"
      cv2.imshow('Original', frame)

      # To check the image information (height, width, color channels)
      print(frame.shape)

      # To resize the image (Original image, target size, interpolation
    method)
```

```
    # Interpolation is a type of estimation to construct new data points
  within the range of a discrete set of known data points. Common used
  methods can be found from
  https://docs.opencv.org/4.4.0/da/d54/group_imgproc_transform.html#ga5b
  b5a1fea74ea38e1a5445ca803ff121
    img_resized = cv2.resize(frame, (320, 240), cv2.INTER_LINEAR)

    # To display the resized image
    cv2.imshow('Resized', img_resized)

    # Press 'q' to quit the program
    if cv2.waitKey(1) & 0xff == ord('q'):
      break

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

### 1.4 Retrieving the required channel

Following is the code snippet for converting the colour image to gray scale image:

```
  import cv2

  # To capture video.
  # Device index is set to 0 as we only have one camera
  cap = cv2.VideoCapture(0)

  while(True):
    # If frame is read correctly, ret will be True, vice versa
    ret, frame = cap.read()
```

```
    # To display the video in new window and named it as "Original"
    cv2.imshow('Original', frame)

    # To check the image information (height, width, color channels)
    print(frame.shape)

    # To convert the colour image to gray scale image
    # You can find out more convertion code from
https://docs.opencv.org/master/d8/d01/group__imgproc__color__conversions
.html
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # To display the gray scale image in new window and named it as "Gray"
    cv2.imshow('Gray', frame_gray)

    # Press 'q' to quit the program
    if cv2.waitKey(1) & 0xff == ord('q'):
      break

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

Did you notice that, the original image consists of 3 channels which are the Blue (B), Green (G), Red (R). However, in some applications, we will convert the colour image to gray scale image before further processing.

### 1.5 Filtering and Edge Detection

In image processing, filtering is used to strengthen preferred features or weaken unwanted features. Mean Filtering, Gaussian filtering and Median Filtering are commonly used for filtering. Filtering can also smoothen or sharpen an image as well as to enhance edges in image. For edge detection, Canny Edges is used to find the boundaries of objects in image.

Following is the code snippet for filtering and edge detection.

**Please uncomment the commented code to test the filtering and edge detection methods one by one and check their differences.

```
import cv2

# To capture video.
# Device index is set to 0 as we only have one camera
cap = cv2.VideoCapture(0)

while(True):
  # If frame is read correctly, ret will be True, vice versa
  ret, frame = cap.read()

  # To display the video in new window and named it as "Original"
  cv2.imshow('Original', frame)

  # with Mean Filtering (Uncomment the code below to try)
  #frame_mean = cv2.blur(frame,(11,11))
  #cv2.imshow('Mean', frame_mean)

  # with Gaussian Filtering (Uncomment the code below to try)
  #frame_gaussian = cv2.GaussianBlur(frame, (11,11), 0)
  #cv2.imshow('Gaussian', frame_gaussian)

  # with Median Filtering (Uncomment the code below to try)
  #frame_gaussian = cv2.medianBlur(frame,11)
  #cv2.imshow('Median', frame_gaussian)

  # with Canny Edges (frame, minVal, maxVal, Sobel Kernel size)
  # Sobel filtering works by calculating the gradient of image intensity
at each pixel within the image
  #frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
  #frame_canny= cv2.Canny(frame_gray, 10, 30, apertureSize=3)
```

```
    #cv2.imshow('Canny', frame_canny)

    # Press 'q' to quit the program
    if cv2.waitKey(1) & 0xff == ord('q'):
      break

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

### 1.6 Region of Interest (ROI) detection

Cascade Classifier is machine learning based approach to detect objects in images. It is extremely fast computation, good for real-time detection and can be applied to other type of object detections.

Following is the code snippet for face detection. Please download the haarcascade_frontalface_default.xml from Luminus and put it in the same folder with this code. You can find more detection models from https://github.com/opencv/opencv/tree/master/data/haarcascades

Please take note that there are some limitations on using this method:

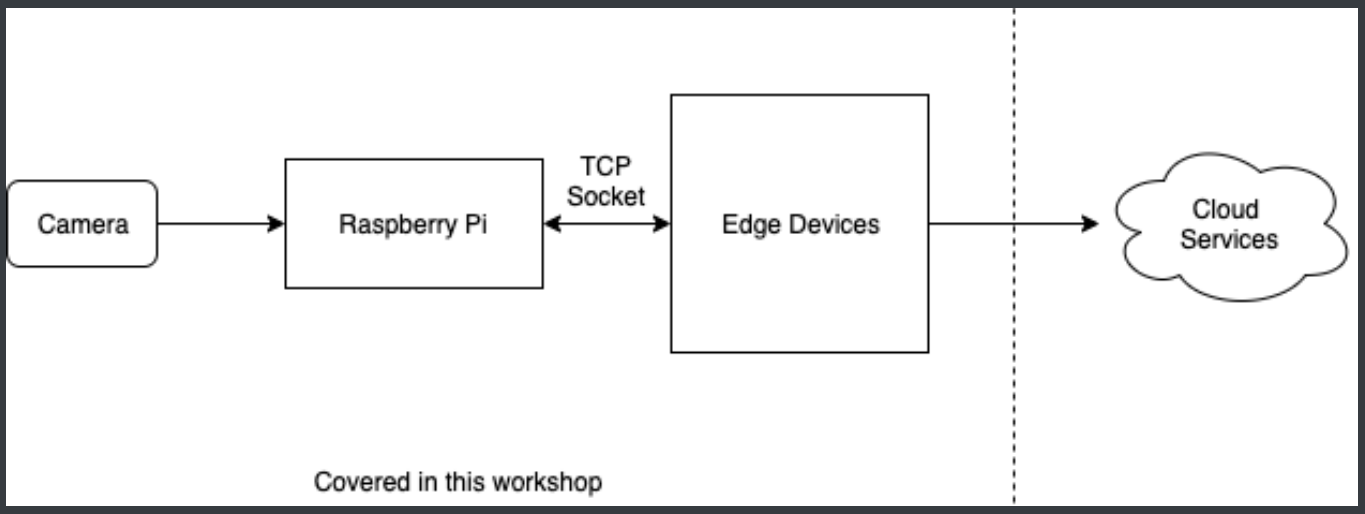
1. Good and effective only on frontal face
2. Not able to cope with faces that rotate around 45 degree
3. Sensitive to lighting conditions

```
import cv2

# To capture video.
# Device index is set to 0 as we only have one camera
cap = cv2.VideoCapture(0)
```

```
while(True):
  # If frame is read correctly, ret will be True, vice versa
  ret, frame = cap.read()

  # Frame should convert into gray scale before perform detection
  frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

  # Load the CascadeClassifier xml file
  face = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

  # Perform detection
  face_dect = face.detectMultiScale(frame_gray, scaleFactor=1.3,
minNeighbors=2)

  # Draw a rectangle box for detected face
  frame_face = frame.copy()
  for (x,y,w,h) in face_dect:
      cv2.rectangle(frame_face, (x,y), (x+w,y+h), (0,255,0), 2)

  # Show the detected face
  cv2.imshow('frame_face',frame_face)

  # Press 'q' to quit the program
  if cv2.waitKey(1) & 0xff == ord('q'):
    break

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

### 1.7 Communication with Edge Server

In this part, we will be sending the image from Raspberry Pi to edge devices via TCP socket as shown below. The image is resized before sending out to minimize the bandwidth usage.




Following is the **client code** to be run in Raspberry Pi. You can treat it as the device (such as CCTV) to capture the image from camera, resizing the image and send it out to edge device:

```
import socket
import cv2
import pickle
import struct

# Replace this with your server IP address
ip = '192.168.XX.XX'
port = 12345

s = socket.socket(socket.AF_INET , socket.SOCK_STREAM)
s.connect((ip, port))
print('Connected to server')
connection = s.makefile('wb')

try:
  # To capture the image
  cap = cv2.VideoCapture(0)
  img_counter = 0

  # Set JPEG Quality; Range from 0 to 100 (100 best)

  encode_param = [int(cv2.IMWRITE_JPEG_QUALITY),90]
```

```
  While True:
        ret, frame = cap.read()
        # Read the image from memory
        # imencode compresses the image and stores it in the memory buffer
        result, frame = cv2.imencode('.jpg', frame, encode_param)

        # Send the image to edge device
        data = pickle.dumps(frame, 0)
        size = len(data)
        s.sendall(struct.pack(">L", size) + data)
        img_counter += 1)

finally:
  print('Sent')
  s.close()
```

Following is the **server code**. You can run this server code on another Raspberry Pi or your computer/laptop. You can treat it as the edge device (such as gateway) to receive the image fromRaspberry Pi and save it:

```
import cv2
import socket
import numpy as np
import pickle
import struct

# Replace this with your host IP address
host = '0.0.0.0'
port = 12345

s = socket.socket(socket.AF_INET , socket.SOCK_STREAM)
s.bind((host, port))
s.listen(5)
```

```
conn,addr = s.accept()
print("Connected to client...")

try:
    name = str('image')+'.jpg'
    data = b""
    payload_size = struct.calcsize(">L")
    print("payload_size: {}".format(payload_size))
    while len(data) < payload_size:
       print("Recv: {}".format(len(data)))
       data += conn.recv(4096)
    print("Done Recv: {}".format(len(data)))
    packed_msg_size = data[:payload_size]
    data = data[payload_size:]
    msg_size = struct.unpack(">L", packed_msg_size)[0]
    print("msg_size: {}".format(msg_size))
    while len(data) < msg_size:
       data += conn.recv(4096)
    frame_data = data[:msg_size]
    data = data[msg_size:]

    frame=pickle.loads(frame_data, fix_imports=True, encoding="bytes")
    frame = cv2.imdecode(frame, cv2.IMREAD_COLOR)
    cv2.imwrite(name,frame)
    cv2.waitKey(1)

finally:
    connection.close()
    s.close()
```

# Project 1: Improving a Smart Store System

After building the solution for autonomous temperature and humidity detection and alert triggering in Day 1 workshop, Musk's boss was very impressed and he tasked Musk again. He requested Musk to capture the image of those people entering the store. The photo should label with timestamp and highlighting the face. The images are then required to store in edge database for recording and further analysis.  Based on what you have learnt so far, can you assist Musk on building the solution?

You may use following code to put text on the image:

```
# Code for putting text (image, text to insert, position, font type, font scale, colour, thickness, anti-aliased line)
cv2.putText(img, 'Minions', (150,410), cv2.FONT_HERSHEY_SIMPLEX, 2.5, (0,0,255), 5, cv2.LINE_AA)
```

You are required to submit the python code(s) to Luminus.