

# **HUMAN DETECTION USING RESBPERRY PI**

- Real-time human detection refers to the ability to identify and locate human figures in images or video streams instantly or with minimal delay.
- Haar Cascade classifiers are a type of object detection algorithm used to identify objects in images or video streams. They are particularly known for their application in detecting faces, but they can be trained to detect various types of objects.

## **Requirements of Human Detection:**

### **1. OpenCV Library:**

Installation can be done via package managers like pip for Python

**Sudo apt install opencv-python3**

```
robin@raspberrypi:~ $ sudo apt install python3-opencv
```

### **Install pip for Python 3:**

Python 3 is the recommended version of Python for most modern applications.

**sudo apt install python3-pip**

### **2. Data Requirements:**

- **Training Data:** If you are training a custom model for human detection, you need a substantial dataset of annotated images.

#### **This dataset should include:**

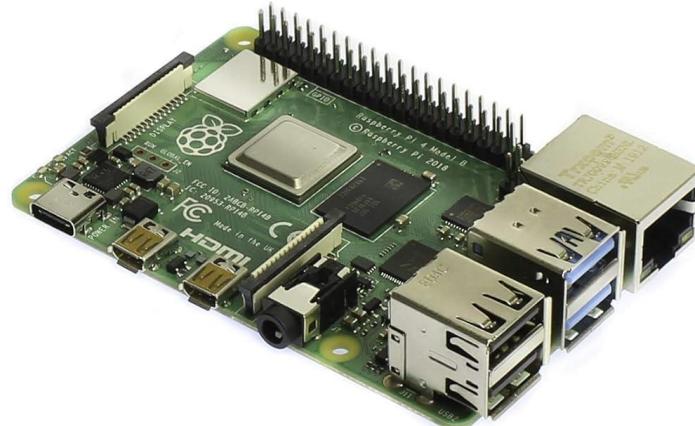
- **Varied Poses:** Images of people in different poses and orientations.
- **Different Lighting Conditions:** Images taken in various lighting conditions to ensure robustness.
- **Different Environments:** Diverse backgrounds and environments to help the model generalize better.
- **Annotations:** Each image should be annotated with bounding boxes or segmentation masks around the humans. Formats like Pascal VOC XML or COCO JSON are commonly used.

#### **Format of Dataset:**

```
dataset/
  images/
    image1.jpg
    image2.jpg
    ...
  annotations/
    image1.xml
    image2.xml
    ...
```

### **3. Hardware Requirements:**

- ❖ **Raspberry Pi Model:** The performance of human detection models will vary based on the Raspberry Pi model:  
**Raspberry Pi 4:** Recommended due to its improved processing power and



memory.

- ❖ **Camera:** A good quality camera module is essential for capturing clear images. The Raspberry Pi Camera Module V2 or a USB webcam with good resolution will suffice.

### **4. OS Installation for Raspberry pi 4:**

- ❖ **Download the OS Image:**

Raspberry Pi OS (formerly Raspbian): The official OS, recommended for most users.

- ❖ **Prepare the SD Card:**

You'll need a microSD card (at least 8 GB recommended) and a card reader. The Raspberry Pi OS image needs to be written to the SD card.

- ❖ **Boot the Raspberry Pi:**

Insert the SD Card: Insert the prepared SD card into the Raspberry Pi's SD card slot. Connect a monitor, keyboard, and mouse (if necessary). Connect the power supply to boot up the Raspberry Pi.



**Pi OS Installation link: <https://www.raspberrypi.com/software/>**

## **5. Connect the Raspberry Pi to Monitor:**

Connecting a Raspberry Pi to a monitor is a straightforward process. The method depends on the type of monitor and the connections available on both the Raspberry Pi and the monitor.

- ❖ **Determine the Type of Monitor and Connections:**

**HDMI Monitor:** If you have a monitor with an HDMI port, you can use a standard HDMI cable.

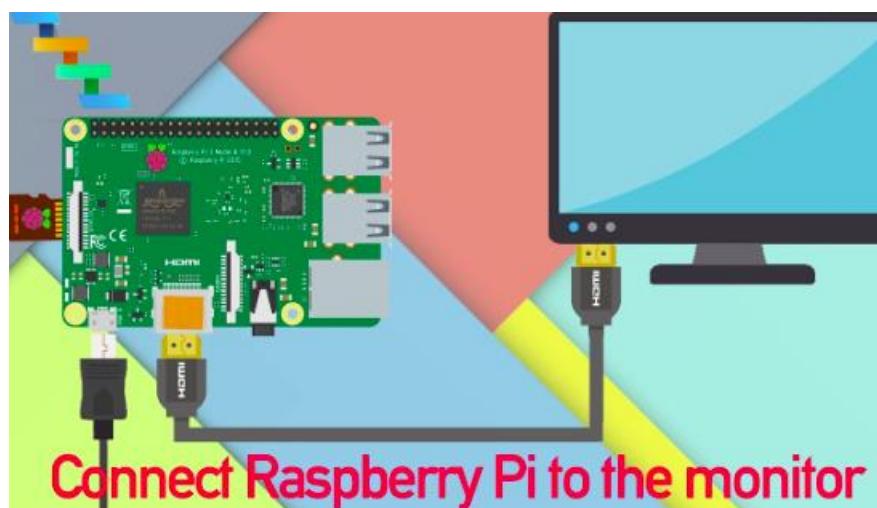
- ❖ **Gather the Necessary Cables and Adapters:**

HDMI Cable: For most modern monitors. Ensure you have a suitable power supply for your Raspberry Pi.

- ❖ **Connect the Monitor to the Raspberry Pi.**

Ensure the Raspberry Pi is powered off before making connections to avoid any potential damage. Simply connect one end of the HDMI cable to the Raspberry Pi's HDMI port and the other end to the monitor's HDMI port.

**“Make sure all connections are secure and firmly in place”**



## Human Detection

### **Aim:**

The aim of this exercise is to implement a real-time Human detection system using Raspberry pi with OpenCV.

### **Hardware and Software Requirements:**

#### **Hardware:**

1. Monitor
2. Raspberry pi
3. Web Camara

#### **Software:**

1. Terminal
2. import Open cv2

### **Code:**

```
Import cv2
import numpy as np
import tflite_runtime.interpreter as tflite

# Load the TensorFlow Lite model
interpreter = tflite.Interpreter(model_path='model.tflite')
interpreter.allocate_tensors()

# Get input and output tensors
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Initialize video capture
cap = cv2.VideoCapture(0)

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

    # Preprocess the image
    input_data = np.expand_dims(frame, axis=0).astype(np.float32)

    # Set input tensor
    interpreter.set_tensor(input_details[0]['index'], input_data)

    # Run inference
    interpreter.invoke()

    # Get output tensor
    boxes = interpreter.get_tensor(output_details[0]['index'])
    classes = interpreter.get_tensor(output_details[1]['index'])
    scores = interpreter.get_tensor(output_details[2]['index'])
```

```

# Post-process and display results
for i in range(len(scores[0])):
    if scores[0][i] > 0.5: # Confidence threshold

    box = boxes[0][i]
    class_id = int(classes[0][i])
    y1, x1, y2, x2 = box
    (h, w, _) = frame.shape
    y1, x1, y2, x2 = int(y1 * h), int(x1 * w), int(y2 * h), int(x2 * w)
    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.putText(frame, f'Class {class_id} ({scores[0][i]:.2f})', (x1, y1 - 10),
               cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

# Display the resulting frame
cv2.imshow('Human Detection', frame)

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

cap.release()

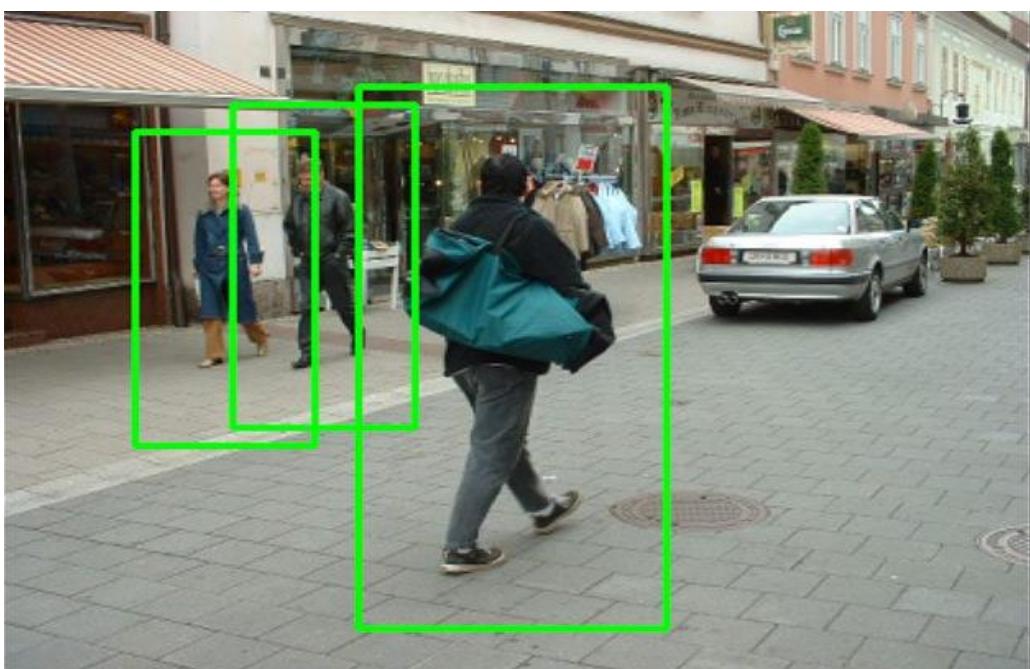
cv2.destroyAllWindows()

```

### Steps:

1. Import cv2
2. Load the Classifier Dataset
3. Initialize Video Capture
4. Define the Detection Function
5. Read Frames and Detect Faces
6. Break the loop and close the window when the 'q' key is pressed.

### Output:



**Reference link:**

**Pi OS Installation:** <https://www.raspberrypi.com/software/>

**Dataset:** <https://www.kaggle.com/datasets/constantinwerner/human-detection-dataset>

**You Tube Reference link:**

1. <https://youtu.be/kX6zWqMP9U4?si=YLCEhT0QhapYTUur>
2. <https://youtu.be/iOTWZI4RHA8?si=d5THT5e5qxvy2aOb>
3. <https://youtu.be/HnjGunbK4u8?si=i3XrWsomv5CrhBjr>

----- X -----

