Lab 3: Sound Analytics with Raspberry Pi 4/3 using Microphone

1. What are some advantages of performing image processing on edge devices like the Raspberry Pi?

Answer:

- Enables **real-time or near-real-time** insights and reactions.
- Enhances **privacy and security** by keeping data local.
- Reduces dependency on cloud or high-bandwidth network connections.

---

2. What commands are used to update and upgrade your Raspberry Pi system?

Answer:

- sudo apt update
- sudo apt upgrade

---

3. Why is a virtual environment named image created in this lab?

Answer:
To isolate and manage Python packages used in the lab (like opencv, mediapipe, etc.), ensuring compatibility and preventing conflicts with system-wide libraries.

---

4. What are the commands to create and activate a virtual environment called image?

Answer:

bash

CopyEdit

sudo apt install python3-venv

python3 -m venv image

source image/bin/activate

---

5. What library is used to access webcam frames in Python?

Answer:
OpenCV (cv2) is used to capture video frames using cv2.VideoCapture().

---

6. How does color segmentation work in OpenCV?

Answer:
By defining **RGB (or BGR)** value ranges for each color and using cv2.inRange() to isolate pixels within that range. A mask is applied to extract only the desired color regions from the image.

---

7. What function in OpenCV is used to apply a color mask to an image?

Answer:
cv2.bitwise_and() is used to apply the mask and extract the desired colored regions.

---

8. Why is the function normalizeImg() used in the color segmentation code?

Answer:
To scale pixel values of each segmented image to 0–255 range, ensuring consistent display quality regardless of the original intensity.

---

9. What is the purpose of cv2.hconcat() in the color segmentation code?

Answer:
It horizontally combines the original frame and segmented images (e.g., red, green, blue) into a single output for visual comparison.

---

10. What library is used in this lab to extract Histogram of Oriented Gradients (HoG) features?

Answer:
scikit-image is used for HoG feature extraction and better visualization.

---

11. What is the purpose of converting images to grayscale before extracting HoG features?

Answer:
HoG focuses on gradient (edge) information, which is more prominent and efficient to compute in **grayscale** images.

---

12. Why is resizing the image important for real-time performance on Raspberry Pi?

Answer:
Downsizing images **reduces computational load**, allowing faster processing and higher frame rates—essential for edge devices with limited resources.

---

13. What does changing the pixels_per_cell or patch size in HoG affect?

Answer:
It affects the **granularity** and **density** of the features extracted. Smaller patches capture fine details; larger patches emphasize broader structures.

---

14. What does OpenCV's cv2.HOGDescriptor() do in the sample code?

Answer:
It initializes a default **pre-trained people detector** using Histogram of Oriented Gradients (HoG) and Support Vector Machine (SVM).

---

15. What does hog.detectMultiScale() return?

Answer:
It returns **bounding boxes** and **confidence scores (weights)** for detected people in the image.

---

16. How does the sample code decide which detected person is closest to the center?

Answer:
It calculates the horizontal distance of each detected person from the frame's center and selects the one with the **smallest distance**.

---

17. What are the outputs printed when a person is detected relative to the center?

Answer:

- "center" if the person is near the center (within tolerance)
- "left" if the person is to the left of the center
- "right" if the person is to the right of the center

---

18. What is MediaPipe, and why is it used in this lab?

Answer:
MediaPipe is an **embedded ML framework** used for efficient and lightweight real-time vision tasks like **face mesh, hand detection**, and **pose estimation**. It runs efficiently on edge devices like the Raspberry Pi.

---

19. How does Mediapipe's FaceMesh differ from OpenCV's traditional face detection?

Answer:
MediaPipe's FaceMesh provides **detailed facial landmark detection** using lightweight ML models, while OpenCV's Haar cascade offers **basic bounding box detection** based on handcrafted features.

---

20. What is the purpose of converting BGR to RGB in the Mediapipe code?

Answer:
MediaPipe expects **RGB images**, while OpenCV captures in **BGR** format. Converting ensures correct color interpretation during processing.

---

21. What function draws the face mesh tesselation and contours in Mediapipe?

Answer:
mp_drawing.draw_landmarks() is used along with predefined styles from mp_drawing_styles.

---

22. Why might one use haarcascade_frontalface_alt2.xml with OpenCV?

Answer:
It is a **Haar Cascade classifier** trained to detect human faces in grayscale images. It provides a simple and fast method for real-time face detection.

---

23. Why is the frame converted to grayscale before Haar-based detection?

Answer:
Haar cascade classifiers are trained on **grayscale images**, and working in grayscale reduces the complexity and speeds up detection.

---

24. What is the effect of resizing frames before detection (e.g., to 256x256)?

Answer:
It **increases processing speed** by reducing the number of pixels to analyze. However, it may also reduce accuracy or miss small faces if downscaled too much.

---

25. What happens in the Haar cascade sample code if no faces are detected?

Answer:
No rectangles are drawn on the frame, and the original video feed is displayed as-is.

---

26. In the color segmentation code, what does the following line do?

python

CopyEdit

mask = cv2.inRange(frame, lower, upper)

Answer:
This line creates a **binary mask** where pixels in the frame that fall within the RGB range specified by lower and upper bounds are turned white (255), and everything else is turned black (0). It is used to isolate specific color regions.

---

27. Why is cv2.bitwise_and() used after applying cv2.inRange()?

Answer:
cv2.bitwise_and() is used to **apply the mask** to the original frame, resulting in an image that shows only the pixels within the specified color range (e.g., red, green, or blue areas).

---

28. In the HoG + SVM people detection code, what does cv2.HOGDescriptor_getDefaultPeopleDetector() do?

Answer:
It returns a **pre-trained Support Vector Machine (SVM)** detector for identifying people using Histogram of Oriented Gradients (HoG) features. This model is commonly used for pedestrian detection.

---

29. Why does the code resize the frame to (256, 256) before running detection?

Answer:
To **reduce computational load** and speed up processing, especially important on resource-constrained devices like the Raspberry Pi.

---

30. What does hog.detectMultiScale() return?

Answer:
It returns two things:

- **boxes** – coordinates of detected regions (bounding boxes).
- **weights** – confidence scores indicating how likely each region contains a person.

---

31. What logic is used to determine whether the person is centered in the frame?

Answer:
The X-coordinate of the center of the bounding box is compared against the center of the frame. If it's within a set **tolerance range** (center_tolerance), the person is considered centered; otherwise, the direction (left or right) is printed.

---

32. In the Mediapipe code, why is the image converted to RGB before processing?

python

CopyEdit

rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

Answer:
Mediapipe models expect **RGB input**, but OpenCV reads frames in **BGR format**. Conversion ensures color channels are in the correct order for accurate inference.

---

33. What are FACEMESH_TESSELATION and FACEMESH_CONTOURS used for in Mediapipe?

Answer:

- FACEMESH_TESSELATION draws a mesh-like grid over the face.
- FACEMESH_CONTOURS highlights key facial features like lips, eyes, and eyebrows with stylized lines.

---

34. In the Haar Cascade face detection code, what does cv2.CascadeClassifier() do?

Answer:
It loads a **pre-trained Haar cascade classifier**, such as haarcascade_frontalface_alt2.xml, which detects objects (faces in this case) in an image using Haar-like features.

---

35. Why is the image converted to grayscale before using detectMultiScale()?

Answer:
Haar cascades are trained on grayscale images, and converting to grayscale simplifies the data, making detection faster and more efficient.

---

36. What is detectMultiScale() and what does it return?

Answer:
It is a function that detects objects (faces here) in an image at multiple scales. It returns a list of **bounding box coordinates** for each detected object.

---

37. What would happen if you skipped cv2.resize() in the OpenCV samples?

Answer:
The system might experience **slower processing and lower frame rates**, especially on Raspberry Pi, due to higher image resolution requiring more computation per frame.

Advanced Feature Detection (Optional Topics)

38. What is SIFT in OpenCV and what is it used for?

Answer:
SIFT (Scale-Invariant Feature Transform) is a **feature detection algorithm** used to

identify and describe keypoints in an image. It's scale and rotation-invariant, making it ideal for tasks like **image matching, recognition, and tracking**.

---

39. What is the difference between SIFT, SURF, and ORB?

Answer:

- **SIFT:** Accurate and robust but slower and patent-encumbered (now public).
- **SURF:** Faster than SIFT, good for real-time, also patented.
- **ORB:** Free, fast, and efficient alternative to SIFT/SURF, suitable for embedded devices.

---

40. What can SIFT, SURF, or ORB be used for in this lab context?

Answer:
They can be used for:

- **Face recognition**
- **Image matching**
- **Tracking moving objects** between frames
- **Augmented reality** applications

---

41. What technique might be used to implement blink detection?

Answer:
You could use **facial landmark detection** (e.g., with Mediapipe) to track **eye aspect ratio (EAR)** — the ratio of eye height to width. A sudden drop in EAR for a few frames usually indicates a blink.

---

42. Why is blink detection useful in real-world applications?

Answer:
Blink detection can be used for:

- **Drowsiness detection** in driver monitoring systems
- **Attention tracking** in education or UX research
- **Human-computer interaction** (e.g., triggering actions through blinks)