volume.py

```python
import cv2
import mediapipe as mp
import pyautogui
import math

x1 = y1 = x2 = y2 = 0
webcam = cv2.VideoCapture(0)  # Function to capture the video
my_hands = mp.solutions.hands.Hands()  # To capture our hands
drawing_utils = mp.solutions.drawing_utils  # To draw points in our hands

while True:
    _, image = webcam.read()  # Read the video frame
    image = cv2.flip(image, flipCode: 1)  # Flip the image horizontally
    frame_height, frame_width, _ = image.shape      # Get frame dimensions
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert image to RGB
    output = my_hands.process(rgb_image)  # Process the image for hand landmarks
    hands = output.multi_hand_landmarks  # Get the detected hands

    if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(image, hand)  # Draw hand landmarks
            landmarks = hand.landmark  # Collect finger landmarks
            for id, landmark in enumerate(landmarks):  # Iterate through landmarks
                x = int(landmark.x * frame_width)
                y = int(landmark.y * frame_height)
```

volume.py

```python
import cv2
import mediapipe as mp
import pyautogui
import math

x1 = y1 = x2 = y2 = 0
webcam = cv2.VideoCapture(0)  # Function to capture the video
my_hands = mp.solutions.hands.Hands()  # To capture our hands
drawing_utils = mp.solutions.drawing_utils  # To draw points in our hands

while True:
    _, image = webcam.read()  # Read the video frame
    image = cv2.flip(image, flipCode: 1)  # Flip the image horizontally
    frame_height, frame_width, _ = image.shape      # Get frame dimensions
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert image to RGB
    output = my_hands.process(rgb_image)  # Process the image for hand landmarks
    hands = output.multi_hand_landmarks  # Get the detected hands

    if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(image, hand)  # Draw hand landmarks
            landmarks = hand.landmark  # Collect finger landmarks
            for id, landmark in enumerate(landmarks):  # Iterate through landmarks
                x = int(landmark.x * frame_width)
                y = int(landmark.y * frame_height)
```

volume.py

```python
import cv2
import mediapipe as mp
import pyautogui
import math

x1 = y1 = x2 = y2 = 0
webcam = cv2.VideoCapture(0)  # Function to capture the video
my_hands = mp.solutions.hands.Hands()  # To capture our hands
drawing_utils = mp.solutions.drawing_utils  # To draw points in our hands

while True:
    _, image = webcam.read()  # Read the video frame
    image = cv2.flip(image, flipCode: 1)  # Flip the image horizontally
    frame_height, frame_width, _ = image.shape      # Get frame dimensions
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert image to RGB
    output = my_hands.process(rgb_image)  # Process the image for hand landmarks
    hands = output.multi_hand_landmarks  # Get the detected hands

    if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(image, hand)  # Draw hand landmarks
            landmarks = hand.landmark  # Collect finger landmarks
            for id, landmark in enumerate(landmarks):  # Iterate through landmarks
                x = int(landmark.x * frame_width)
                y = int(landmark.y * frame_height)
```

volume.py

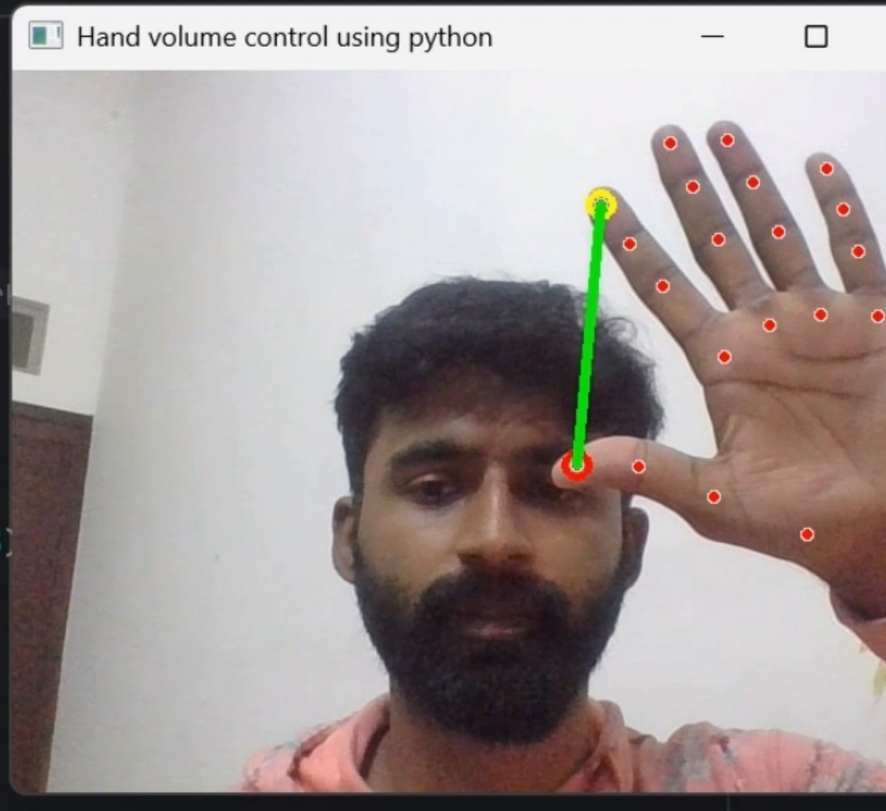
```python
import cv2
import mediapipe as mp
import pyautogui
import math

x1 = y1 = x2 = y2 = 0
webcam = cv2.VideoCapture(0)  # Function to capture the video
my_hands = mp.solutions.hands.Hands()  # To capture our hands
drawing_utils = mp.solutions.drawing_utils  # To draw points in our hands

while True:
    _, image = webcam.read()  # Read the video frame
    image = cv2.flip(image, flipCode: 1)  # Flip the image horizontally
    frame_height, frame_width, _ = image.shape      # Get frame dimensions
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert image to RGB
    output = my_hands.process(rgb_image)  # Process the image for hand landmarks
    hands = output.multi_hand_landmarks  # Get the detected hands

    if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(image, hand)  # Draw hand landmarks
            landmarks = hand.landmark  # Collect finger landmarks
            for id, landmark in enumerate(landmarks):  # Iterate through landmarks
                x = int(landmark.x * frame_width)
                y = int(landmark.y * frame_height)
```

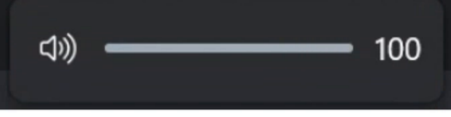
volume.py

```python
import cv2
import mediapipe as mp
import pyautogui
import math

x1 = y1 = x2 = y2 = 0
webcam = cv2.VideoCapture(0)  # Function to capture the video
my_hands = mp.solutions.hands.Hands()  # To capture our hands
drawing_utils = mp.solutions.drawing_utils  # To draw points in our hands

while True:
    _, image = webcam.read()  # Read the video frame
    image = cv2.flip(image, flipCode: 1)  # Flip the image horizontally
    frame_height, frame_width, _ = image.shape      # Get frame dimensions
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert image to RGB
    output = my_hands.process(rgb_image)  # Process the image for hand landmarks
    hands = output.multi_hand_landmarks  # Get the detected hands

    if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(image, hand)  # Draw hand landmarks
            landmarks = hand.landmark  # Collect finger landmarks
            for id, landmark in enumerate(landmarks):  # Iterate through landmarks
                x = int(landmark.x * frame_width)
                y = int(landmark.y * frame_height)
```

volume.py

```
18
19      if hands:
20          for hand in hands:
21              drawing_utils.draw_landmarks(image, hand)  # Draw hand landmarks
22              landmarks = hand.landmark  # Collect finger landmarks
23              for id, landmark in enumerate(landmarks):  # Iterate through landmar
24                  x = int(landmark.x * frame_width)
25                  y = int(landmark.y * frame_height)
26
27                  if id == 8:  # Tip of the index finger
28                      cv2.circle(image, center: (x, y), radius: 8, color: (0, 255, 255)
29                      x1, y1 = x, y
30                  if id == 4:  # Tip of the thumb
31                      cv2.circle(image, center: (x, y), radius: 8, color: (0, 0, 255),
32                      x2, y2 = x, y
33
34          # Calculate Euclidean distance between thumb and index finger
35          dist = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
36
37          # Draw a line between thumb and index finger
38          cv2.line(image, pt1: (x1, y1), pt2: (x2, y2), color: (0, 255, 0), thickness: 5)
39
40          # Volume control based on distance
41          if dist > 50:
42              pyautogui.press("volumeup")
```



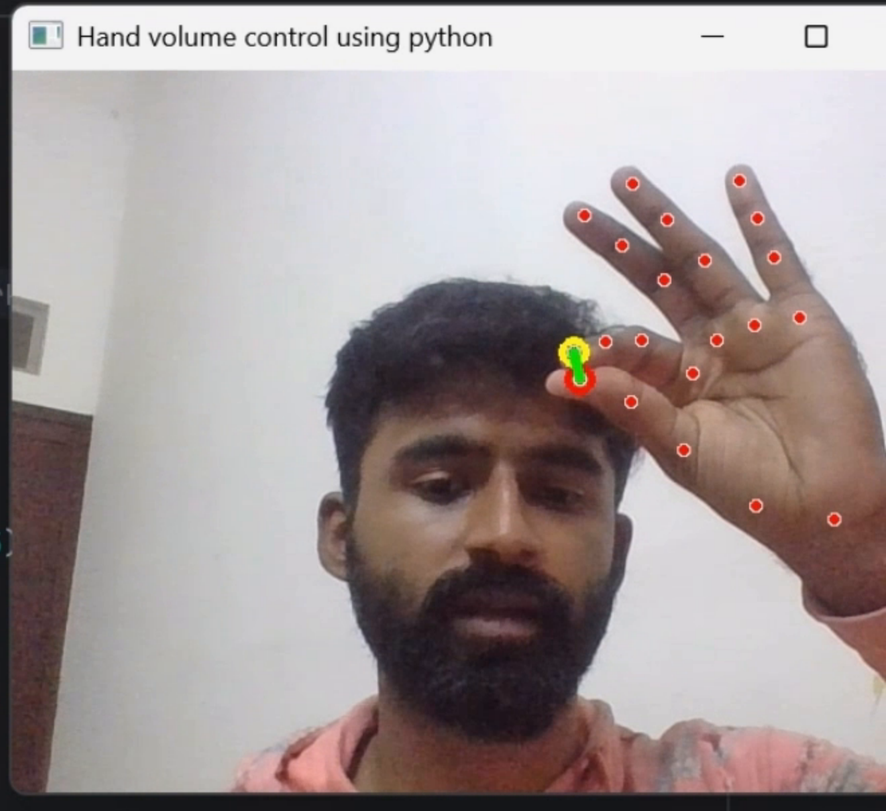


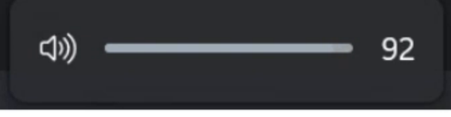
Python > volume.py

23:83 CRLF UTF-8 4 spaces

volume.py

```
18
19      if hands:
20          for hand in hands:
21              drawing_utils.draw_landmarks(image, hand)  # Draw hand landmarks
22              landmarks = hand.landmark  # Collect finger landmarks
23              for id, landmark in enumerate(landmarks):  # Iterate through landmar
24                  x = int(landmark.x * frame_width)
25                  y = int(landmark.y * frame_height)
26
27                  if id == 8:  # Tip of the index finger
28                      cv2.circle(image, center: (x, y), radius: 8, color: (0, 255, 255)
29                      x1, y1 = x, y
30                  if id == 4:  # Tip of the thumb
31                      cv2.circle(image, center: (x, y), radius: 8, color: (0, 0, 255),
32                      x2, y2 = x, y
33
34          # Calculate Euclidean distance between thumb and index finger
35          dist = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
36
37          # Draw a line between thumb and index finger
38          cv2.line(image, pt1: (x1, y1), pt2: (x2, y2), color: (0, 255, 0), thickness: 5)
39
40          # Volume control based on distance
41          if dist > 50:
42              pyautogui.press("volumeup")
```



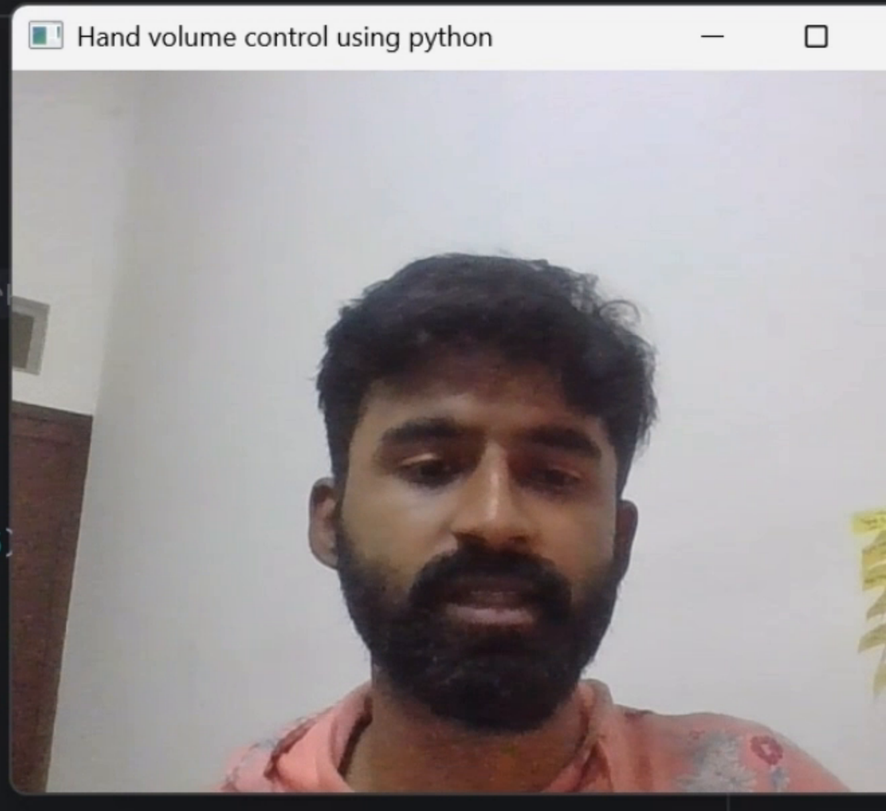


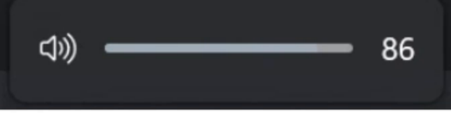
Python > volume.py

23:83 CRLF UTF-8 4 spaces

volume.py

```
18
19      if hands:
20          for hand in hands:
21              drawing_utils.draw_landmarks(image, hand)  # Draw hand landmarks
22              landmarks = hand.landmark  # Collect finger landmarks
23              for id, landmark in enumerate(landmarks):  # Iterate through landmar
24                  x = int(landmark.x * frame_width)
25                  y = int(landmark.y * frame_height)
26
27                  if id == 8:  # Tip of the index finger
28                      cv2.circle(image, center: (x, y), radius: 8, color: (0, 255, 255)
29                      x1, y1 = x, y
30                  if id == 4:  # Tip of the thumb
31                      cv2.circle(image, center: (x, y), radius: 8, color: (0, 0, 255),
32                      x2, y2 = x, y
33
34          # Calculate Euclidean distance between thumb and index finger
35          dist = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
36
37          # Draw a line between thumb and index finger
38          cv2.line(image, pt1: (x1, y1), pt2: (x2, y2), color: (0, 255, 0), thickness: 5)
39
40          # Volume control based on distance
41          if dist > 50:
42              pyautogui.press("volumeup")
```



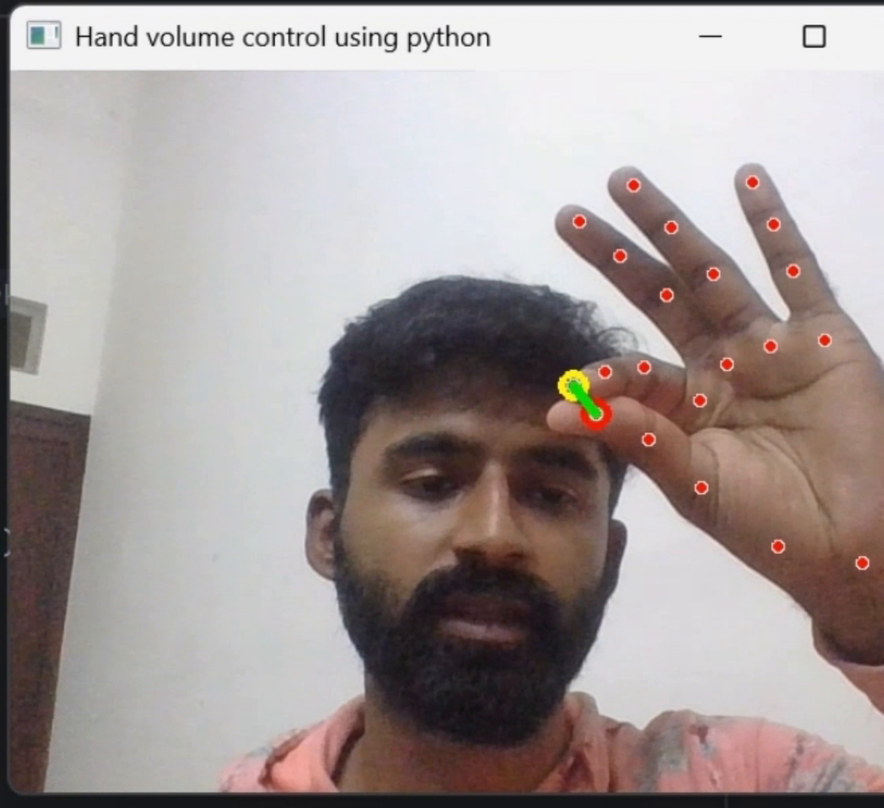


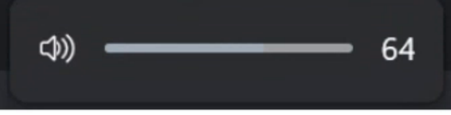
Python > volume.py

23:83 CRLF UTF-8 4 spaces

volume.py

```
18
19      if hands:
20          for hand in hands:
21              drawing_utils.draw_landmarks(image, hand)  # Draw hand landmarks
22              landmarks = hand.landmark  # Collect finger landmarks
23              for id, landmark in enumerate(landmarks):  # Iterate through landmar
24                  x = int(landmark.x * frame_width)
25                  y = int(landmark.y * frame_height)
26
27                  if id == 8:  # Tip of the index finger
28                      cv2.circle(image, center: (x, y), radius: 8, color: (0, 255, 255)
29                      x1, y1 = x, y
30                  if id == 4:  # Tip of the thumb
31                      cv2.circle(image, center: (x, y), radius: 8, color: (0, 0, 255),
32                      x2, y2 = x, y
33
34          # Calculate Euclidean distance between thumb and index finger
35          dist = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
36
37          # Draw a line between thumb and index finger
38          cv2.line(image, pt1: (x1, y1), pt2: (x2, y2), color: (0, 255, 0), thickness: 5)
39
40          # Volume control based on distance
41          if dist > 50:
42              pyautogui.press("volumeup")
```



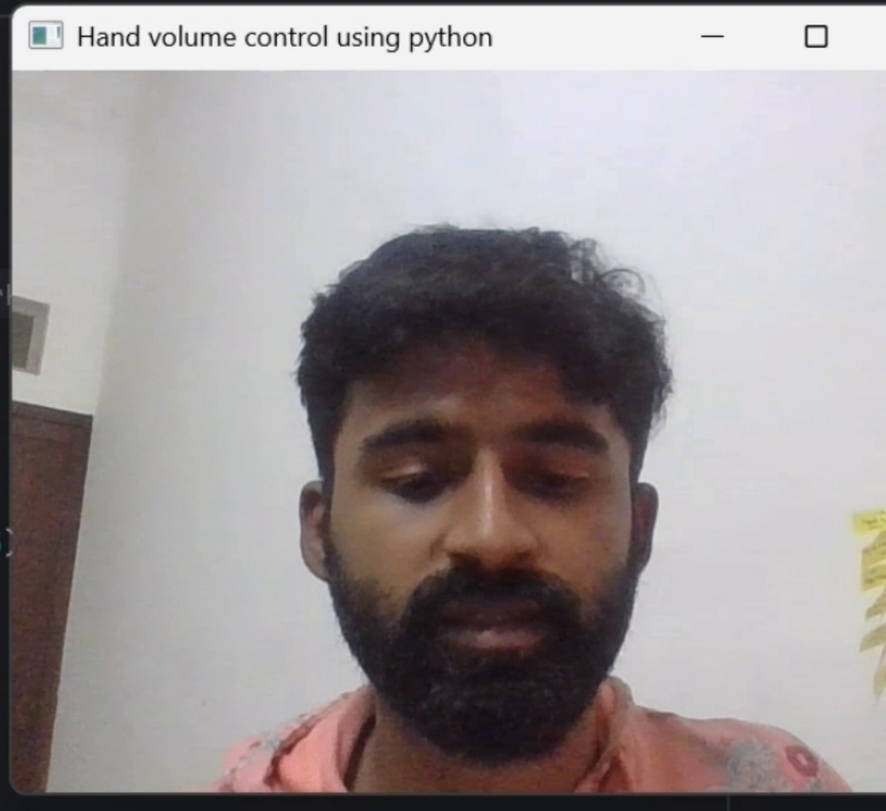


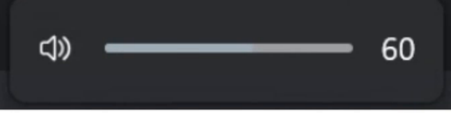
Python > volume.py

23:83 CRLF UTF-8 4 spaces

volume.py

```
18
19      if hands:
20          for hand in hands:
21              drawing_utils.draw_landmarks(image, hand)  # Draw hand landmarks
22              landmarks = hand.landmark  # Collect finger landmarks
23              for id, landmark in enumerate(landmarks):  # Iterate through landmar
24                  x = int(landmark.x * frame_width)
25                  y = int(landmark.y * frame_height)
26
27                  if id == 8:  # Tip of the index finger
28                      cv2.circle(image, center: (x, y), radius: 8, color: (0, 255, 255)
29                      x1, y1 = x, y
30                  if id == 4:  # Tip of the thumb
31                      cv2.circle(image, center: (x, y), radius: 8, color: (0, 0, 255),
32                      x2, y2 = x, y
33
34          # Calculate Euclidean distance between thumb and index finger
35          dist = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
36
37          # Draw a line between thumb and index finger
38          cv2.line(image, pt1: (x1, y1), pt2: (x2, y2), color: (0, 255, 0), thickness: 5)
39
40          # Volume control based on distance
41          if dist > 50:
42              pyautogui.press("volumeup")
```





Python > volume.py

23:83 CRLF UTF-8 4 spaces

youtube.com/watch?v=yJg-Y5byM...

Premium IN

6

Warriyo - Mortals (feat. Laura Brehm) | Future Trap | NCS - Copyright Free Music

NoCopyrightSounds
33.7M subscribers

Subscribe

3.4M | Share | Download | Clip | ...

4

d landmarks

through landmar

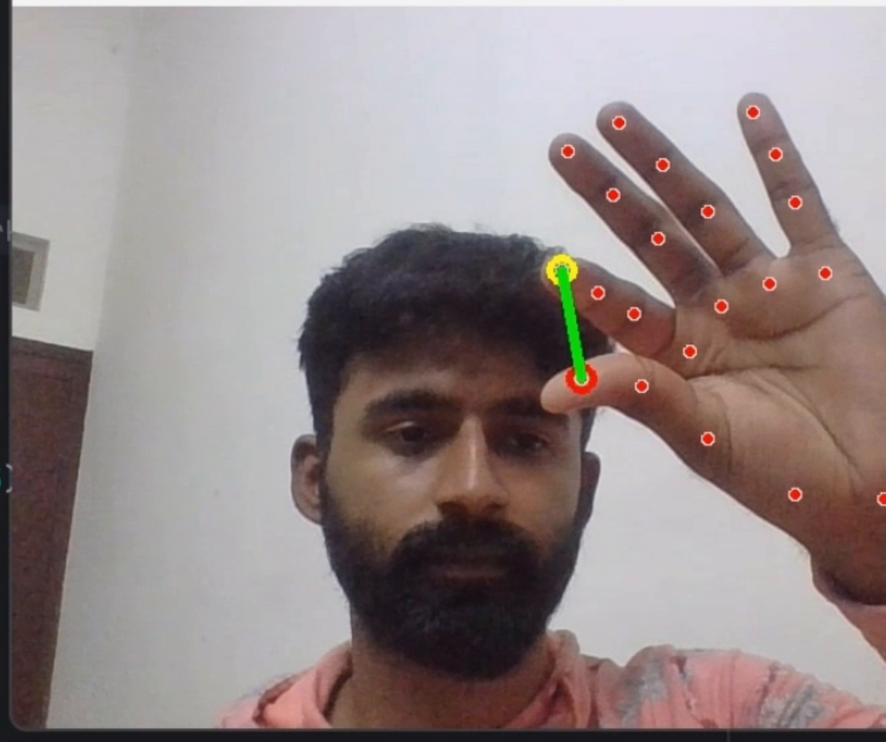
or: (0, 255, 255)

or: (0, 0, 255),

nger

5, 0), thickness: 5)

Hand volume control using python



23:83 CRLF UTF-8 4 spaces

youtube.com/watch?v=yJg-Y5byM...

Premium IN

6

Warriyo - Mortals (feat. Laura Brehm) | Future Trap | NCS - Copyright Free Music

NoCopyrightSounds
33.7M subscribers

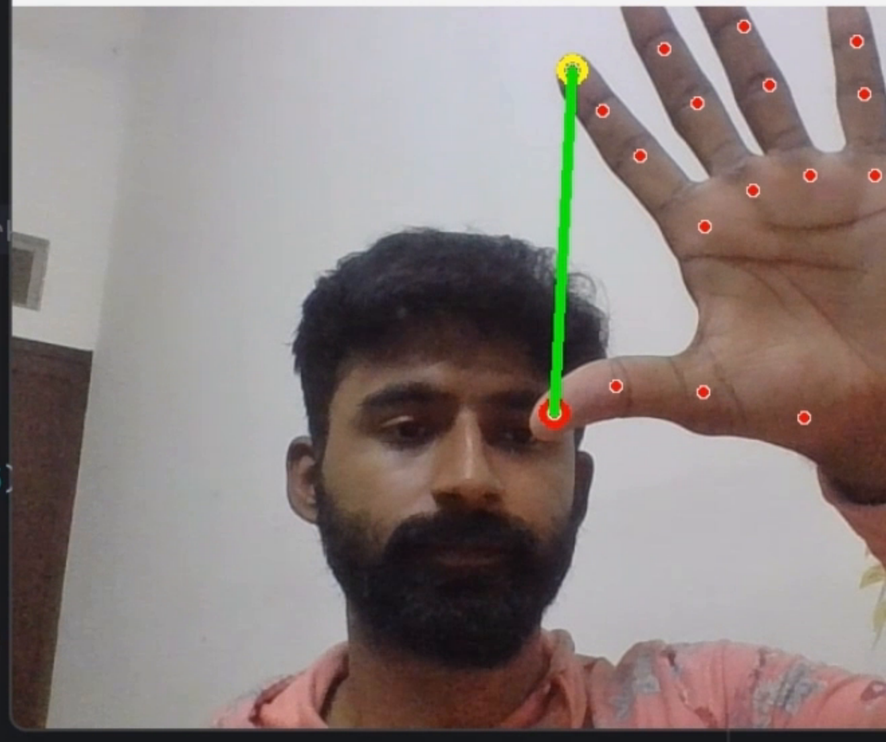
Subscribe

3.4M | Share | Download | Clip | ...

100

```
d landmarks
through landmar
or: (0, 255, 255)
or: (0, 0, 255),
nger
5, 0), thickness: 5)
```

Hand volume control using python



23:83 CRLF UTF-8 4 spaces