Volume Control With Hand Detection

TEAM MEMBERS :

RAKESH NARAPAREDDY - RA1911033010079
LVR DHEERAJ-RA1911033010092

ABSTRACT:

The Volume Control With Hand Detection Open CV Pyhton was developed using Pyhton Open CV, In this Python Open CV Project we are going Build a Volume Controller with Open CV. To change the volume of a computer.

Gesture recognition helps computers to understand human body language. This helps to build a more potent link between humans and machines, rather than just the basic text user interfaces or graphical user interfaces (GUIs). In this project for gesture recognition, the human body's motions are read by computer camera.

OBJECTIVE :

The objective of this project is to develop an interface which will capture human hand gesture dynamically and will control the volume level. For this, Deep Learning techniques such as Yolo model, Inception Net model+ LSTM, 3-D CNN+LSTM and Time Distributed CNN+LSTM have been studied to compare the results of hand detection.

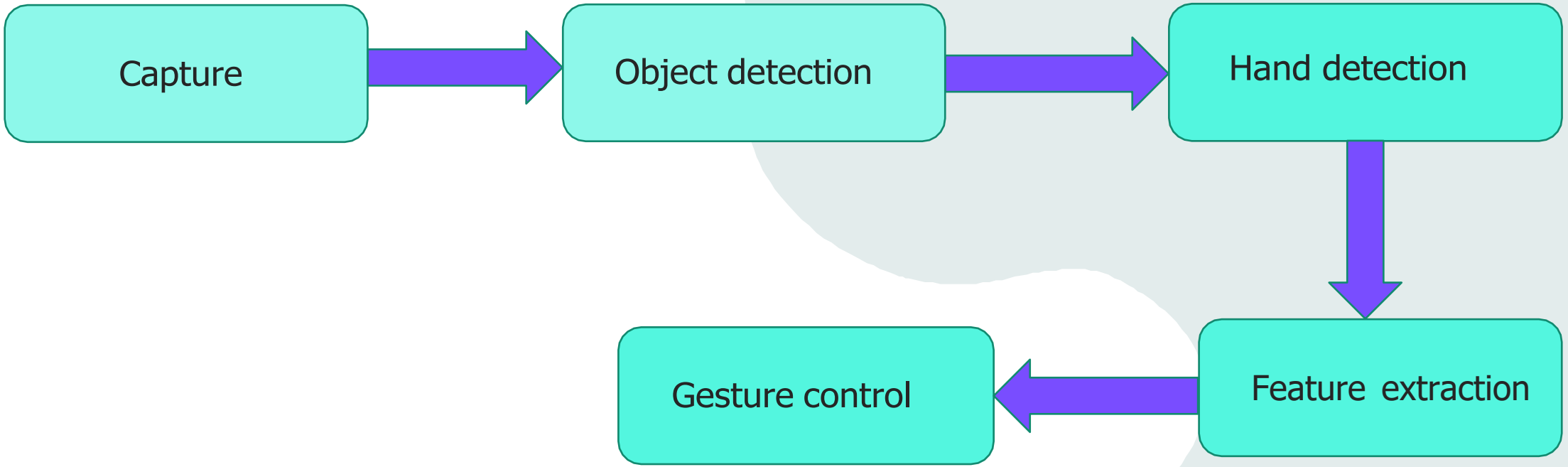
ISSUES IN EXISTING PROJECT :

- Can't be used for long distance
- Sometimes not accurate
- Requires a decent camera
- May be confused by two palms

PROPOSED METHODOLOGY :

- Detect hand landmarks
- Calculate the distance between thumb tip and index finger tip.
- Map the distance of thumb tip and index finger tip with volume range. For my case, distance between thumb tip and index finger tip was within the range of 30 – 350 and the volume range was from –63.5 – 0.0.
- In order to exit press ‘Spacebar'.

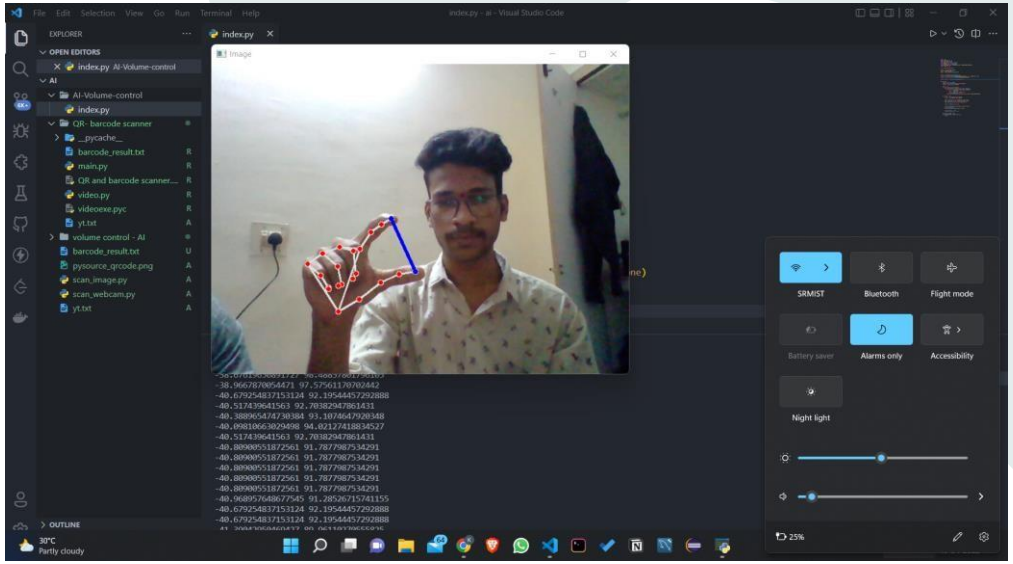
System Flow Diagram :

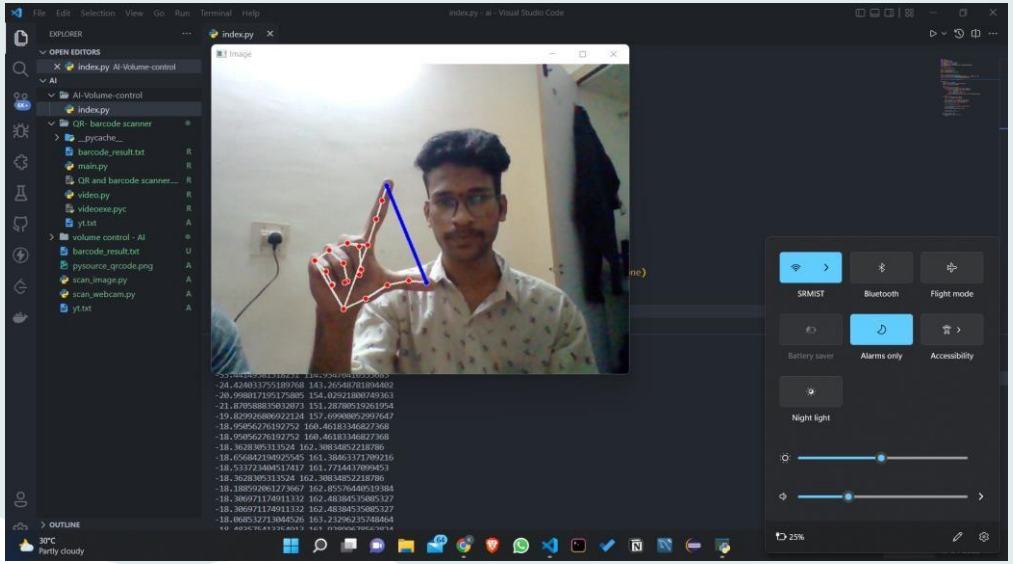


WORKING PRINCIPLE :

The camera in our device is used for this project. It detects our hand with points in it so as it can see the distance between our thumb fingertip and index finger tip. The distance between the points 4 and 8 is directly proportional to the volume of device.

SCREENSHOTS :





```
import cv2
import mediapipe as mp
from math import hypot
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import numpy as np
cap = cv2.VideoCapture(0)
mpHands = mp.solutions.hands
hands = mpHands.Hands()
mpDraw = mp.solutions.drawing_utils
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
volMin,volMax = volume.GetVolumeRange()[:2]
while True:
success,img = cap.read()
imgRGB = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
results = hands.process(imgRGB)
lmList = []
if results.multi_hand_landmarks:
for handlandmark in results.multi_hand_landmarks:
```

```
for id,lm in enumerate(handlandmark.landmark):
h,w,_ = img.shape
cx,cy = int(lm.x*w),int(lm.y*h)
lmList.append([id,cx,cy])
mpDraw.draw_landmarks(img,handlandmark,mpHands.HAND_CONNECTIONS)
if lmList != []:
x1,y1 = lmList[4][1],lmList[4][2]
x2,y2 = lmList[8][1],lmList[8][2]
cv2.circle(img,(x1,y1),4,(255,0,0),cv2.FILLED)
cv2.circle(img,(x2,y2),4,(255,0,0),cv2.FILLED)
cv2.line(img,(x1,y1),(x2,y2),(255,0,0),3)
length = hypot(x2-x1,y2-y1)
vol = np.interp(length,[15,220],[volMin,volMax])
print(vol,length)
volume.SetMasterVolumeLevel(vol, None)
# Hand range 15 - 220
# Volume range -63.5 - 0.0
cv2.imshow('Image',img)
if cv2.waitKey(1) & 0xff==ord('q'):
break
```