

Libraries

OpenCV

Mediapipe

Creating 2 Python Files for each project

HandTrackingModule.py	FingerCounting.py
HandTrackingModule.py	AVirtualPainting.py
HandTrackingModule.py	HandVirtualMouse.py
HandTrackingModule.py	HandVolumeControl.py

For each project, it is important to create a separate python file that comprises Class and the functions associated with that respective project.

Defining Hand Class and it's Methods

```
import cv2
import mediapipe as mp
import time
import math
import numpy as np

class handDetector():
    def __init__(self,
mode=False, maxHands=2,
detectionCon=0, trackCon=0):
        self.mode =
mode
        self.maxHands
= maxHands
        self.detectionCon = detectionCon
        self.trackCon
= trackCon
        self.mpHands =
mp.solutions.hands
```

Defining Hand Class and it's Methods (cont)

```
> self.hands = self.mpHands.Hands-
(self.mode, self.maxHands, self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

def findHands(self, img, draw=True):
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    self.results = self.hands.process(imgRGB)
    # print(results.multi_hand_landmarks)

    if self.results.multi_hand_landmarks:
        for handLms in self.results.multi_hand_landmarks:
            if draw:
                self.mpDraw.draw_landmarks(img, handLms, self.mpHands.HAND_CONNECTIONS)

            return img

def findPosition(self, img, handNo=0, draw=True):
    xList = []
    yList = []
    bbox = []
    self.lmList = []
    if self.results.multi_hand_landmarks:
        myHand = self.results.multi_hand_landmarks[handNo]
        for id, lm in enumerate(myHand.landmark):
            # print(id, lm)
            h, w, c = img.shape
            cx, cy = int(lm.x*w), int(lm.y*h)
            xList.append(cx)
            yList.append(cy)
            # print(id, cx, cy)
            self.lmList.append([id, cx, cy])
            if draw:
```

Defining Hand Class and it's Methods (cont)

```
> cv2.circle(img, (cx, cy), 5,
(255, 0, 255), cv2.FILLED)

        xmin, xmax = min(xList), max(xList)
        ymin, ymax = min(yList), max(yList)
        bbox = xmin, ymin, xmax, ymax

        if draw:
            cv2.rectangle(img, (xmin - 20,
ymin - 20), (xmax + 20, ymax + 20), (0, 255, 0), 2)

            return self.lmList, bbox

def fingersUp(self):
    fingers = []

    # Thumb
    if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
        fingers.append(1)
    else:
        fingers.append(0)

    # Fingers
    for id in range(1, 5):
        if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:
            fingers.append(1)
        else:
            fingers.append(0)

    # totalFingers = fingers.count(1)

    return fingers

def findDistance(self, p1, p2, img, draw=True, r=15, t=3):
    x1, y1 = self.lmList[p1][1:]
    x2, y2 = self.lmList[p2][1:]
```



By meghalagarwal

Not published yet.

Last updated 31st May, 2022.

Page 1 of 4.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Defining Hand Class and it's Methods (cont)

```
> cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

if draw:
    cv2.line(img, (x1, y1), (x2, y2), (255,
0, 255), t)
    cv2.circle(img, (x1, y1), r, (255, 0,
255), cv2.FILLED)
    cv2.circle(img, (x2, y2), r, (255, 0,
255), cv2.FILLED)
    cv2.circle(img, (cx, cy), r, (0, 0,
255), cv2.FILLED)
    length = math.hypot(x2 - x1, y2 - y1)

return length, img, [x1, y1, x2, y2, cx,
cy]
```

This Class represents the Hand and the methods represent the various actions associated with those hands and finger moments in order to capture the actions.

HandVirtualPaint File

```
import cv2 as cv
import numpy as np
import time
import os
import HandTrackingModule as htm
brushThickness = 25
eraserThickness = 100
folderPath = "/home/meghal/Personal/Kernelvegerge -
_AI/Tracking/AdvanceComputerVisions/AIVirtualPainting/Head er"
myList = os.listdir(folderPath)
myList.sort()
print(myList)
```

HandVirtualPaint File (cont)

```
> overlayList = []
for imagePath in myList:
    image = cv.imread(f"{folderPath}/{imagePath}")
    overlayList.append(image)
#overlayList.sort()
print(len(overlayList))
header = overlayList[0]
drawColor = (255, 0, 255)
cap = cv.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
detector = htm.HandDetector(detectionCon=0,maxHands=1)
xp, yp = 0, 0
vidCanvas = np.zeros((480, 640, 3), np.uint8)
while True:
    # 1. Import image
    success, vid = cap.read()
    vid = cv.flip(vid, 1)
    #print(vid.shape)
    # 2. Find Hand Landmarks
    vid = detector.findHands(vid)
    lmList = detector.findPosition(vid, draw=False)
    if len(lmList) != 0:
        print(lmList)
        # tip of index and middle fingers
        x1, y1 = lmList[8][1:]
        x2, y2 = lmList[12][1:]
        # 3. Check which fingers are up
        fingers = detector.fingersUp()
```

HandVirtualPaint File (cont)

```
> # print(fingers)
# 4. If Selection Mode – Two finger are up
if fingers[1] and fingers[2]:
    # xp, yp = 0, 0
    #cv.rectangle(vid, (x1, y1-25), (x2,
y2+25), (255, 0, 255), cv.FILLED)
    print("Selection Mode")
    # Checking for the click
    if y1 < 100:
        if 50 < x1 < 150:
            header = overlayList[0]
            drawColor = (255, 0, 255)
        elif 150 < x1 < 200:
            header = overlayList[1]
            drawColor = (255, 0, 0)
        elif 240 < x1 < 390:
            header = overlayList[2]
            drawColor = (0, 255, 0)
        elif 490 < x1 < 640:
            header = overlayList[3]
            drawColor = (0, 0, 0)
            cv.rectangle(vid, (x1, y1 - 25),
(x2, y2 + 25), drawColor, cv.FILLED)
    # 5. If Drawing Mode – Index finger is up
    if fingers[1] and fingers[2] == False:
        cv.circle(vid, (x1, y1), 15, drawColor,
cv.FILLED)
        print("Drawing Mode")
        if xp == 0 and yp == 0:
            xp, yp = x1, y1
```



By meghalagarwal

cheatography.com/meghalagarwal/

Not published yet.

Last updated 31st May, 2022.

Page 2 of 4.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

HandVirtualPaint File (cont)

```
> #cv.line(vid, (xp, yp), (x1, y1),
drawColor, brushThickness)
    if drawColor == (0, 0, 0):
        cv.line(vid, (xp, yp), (x1, y1),
drawColor, eraserThickness)
        cv.line(vidCanvas, (xp, yp), (x1,
y1), drawColor, eraserThickness)
    else:
        cv.line(vid, (xp, yp), (x1, y1),
drawColor, brushThickness)
        cv.line(vidCanvas, (xp, yp), (x1,
y1), drawColor, brushThickness)
    xp, yp = x1, y1
    ## Clear Canvas when all fingers
are up
    # if all (x >= 1 for x in fingers):
    # vidCanvas = np.zeros((720, 1280,
3), np.uint8)
    vidGray = cv.cvtColor(vidCanvas,
cv.COLOR_BGR2GRAY)
    _, vidInv = cv.threshold(vidGray, 50, 255,
cv.THRESH_BINARY_INV)
    vidInv = cv.cvtColor(vidInv, cv.COLOR-
_GRAY2BGR)
    vid = cv.bitwise_and(vid, vidInv)
    vid = cv.bitwise_or(vid, vidCanvas)
    # Setting the header image
    vid[0:100, 0:640] = header
    #vid = cv.addWeighted(vid, 0.5, vidCan-
vas, 0.5, 0)
    cv.imshow("Video", vid)
    #cv.imshow("Video Canvas", vidCanvas)
    #print(vidCanvas.shape)
    # cv.imshow("Video Inv", vidInv)
```

HandVirtualPaint File (cont)

```
> # print(vidInv.shape)
cv.waitKey(1)
```

This Code represents the logic of how to detect hand moments i.e. Selection / Drawing.

AI Virtual Mouse

```
import cv2
import numpy as np
import HandTrackingModule as
htm
import time
import autopy
wCam, hCam = 640, 480
frameR = 100 # Frame Reduction
smoothing = 7
pTime = 0
plocX, plocY = 0, 0
clocX, clocY = 0, 0
cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)
detector = htm.HandDetector(
rmaxHands=1)
wScr, hScr = autopy.screen.size()
# print(wScr, hScr)
while True:
    # 1. Find hand Landmarks
    success, img =
cap.read()
    img = detector.find-
dHands(img)
    lmList, bbox = detect-
or.findPosition(img)
    # 2. Get the tip of the
index and middle fingers
    if len(lmList) != 0:
        x1, y1 = lmList -
[8][1:]
```

AI Virtual Mouse (cont)

```
> x2, y2 = lmList[12][1:]
    # print(x1, y1, x2, y2)
    # 3. Check which fingers are up
    fingers = detector.fingersUp()
    # print(fingers)
    cv2.rectangle(img, (frameR, frameR),
(wCam - frameR, hCam - frameR), (255, 0,
255), 2)
    # 4. Only Index Finger : Moving Mode
    if fingers[1] == 1 and fingers[2] == 0:
    # 5. Convert Coordinates
        x3 = np.interp(x1, (frameR, wCam -
frameR), (0, wScr))
        y3 = np.interp(y1, (frameR, hCam -
frameR), (0, hScr))
        print(x3)
    # 6. Smoothen Values
        clocX = plocX + (x3 - plocX) / smooth-
ening
        clocY = plocY + (y3 - plocY) / smooth-
ening
    # 7. Move Mouse
        autopy.mouse.move(wScr - clocX,
clocY)
        cv2.circle(img, (x1, y1), 15, (255, 0,
255), cv2.FILLED)
        plocX, plocY = clocX, clocY
    # 8. Both Index and middle fingers are up
: Clicking Mode
    if fingers[1] == 1 and fingers[2] == 1:
    # 9. Find distance between fingers
        length, img, lineInfo = detector.findDi-
stance(8, 12, img)
        print(length)
```



By meghalagarwal

Not published yet.

Last updated 31st May, 2022.

Page 3 of 4.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

AI Virtual Mouse (cont)

```
> # 10. Click mouse if distance short
if length < 40:
    cv2.circle(img, (lineInfo[4], lineInfo[5]),
               15, (0, 255, 0), cv2.FILLED)
    autopy.mouse.click()

# 11. Frame Rate
cTime = time.time()
fps = 1 / (cTime - pTime)
pTime = cTime
cv2.putText(img, str(int(fps)), (20, 50),
cv2.FONT_HERSHEY_PLAIN, 3, (255, 0, 0), 3)

# 12. Display
cv2.imshow("Image", img)
cv2.waitKey(1)
```

It is the same as Hand Virtual Painting Tool.

Virtual Volume Control

```
import cv2
import time
import numpy as np
import HandTrackingModule as htm
import math
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume

# Defining the Width and Height of the Video to be displayed
wCam, hCam = 640, 480
cap = cv2.VideoCapture(0)
cap.set(3, wCam)
```

Virtual Volume Control (cont)

```
> cap.set(4, hCam)
pTime = 0
detector = htm.handDetector(detectionCon=0.7)
devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
# volume.GetMute()
# volume.GetMasterVolumeLevel()
volRange = volume.GetVolumeRange()
minVol = volRange[0]
maxVol = volRange[1]
vol = 0
volBar = 400
volPer = 0
while True:
    success, img = cap.read()
    img = detector.findHands(img)
    lmList = detector.findPosition(img, draw=False)
    if len(lmList) != 0:
        # print(lmList[4], lmList[8])
        x1, y1 = lmList[4][1], lmList[4][2]
        x2, y2 = lmList[8][1], lmList[8][2]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2
        cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
        cv2.circle(img, (x2, y2), 15, (255, 0, 255), cv2.FILLED)
```

Virtual Volume Control (cont)

```
> cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), 3)
    cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)

    length = math.hypot(x2 - x1, y2 - y1)
    # Hand range 50 - 300
    # Volume Range -65 - 0
    vol = np.interp(length, [50, 300], [minVol, maxVol])
    volBar = np.interp(length, [50, 300], [400, 150])
    volPer = np.interp(length, [50, 300], [0, 100])
    print(int(length), vol)
    volume.SetMasterVolumeLevel(vol, None)
    if length < 50:
        cv2.circle(img, (cx, cy), 15, (0, 255, 0), cv2.FILLED)
        cv2.rectangle(img, (50, 150), (85, 400), (255, 0, 0), 3)
        cv2.rectangle(img, (50, int(volBar)), (85, 400), (255, 0, 0), cv2.FILLED)
        cv2.putText(img, f'{int(volPer)} %', (40, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0), 3)
        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        cv2.putText(img, f'FPS: {int(fps)}', (40, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0), 3)
        cv2.imshow("Img", img)
        cv2.waitKey(1)
```

It captures the tips and the moments of 2 finger's tips to control the volume.



By meghalagarwal

cheatography.com/meghalagarwal/

Not published yet.

Last updated 31st May, 2022.

Page 4 of 4.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>