In [ ]:

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


##########################
wCam, hCam = 640, 480
frameR = 100  # Frame Reduction
smoothening = 5
#########################

pTime = 0
plocX, plocY = 0, 0
clocX, clocY = 0, 0




class detector():
    def __init__(self, mode=False, maxHands=2, detect=0.5, track=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detect = detect
        self.track = track

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
                                        self.detect, self.track)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]


    def fhand(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 fPosition(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:
                    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:]
        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]



cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)
detector = detector(maxHands=1)
wScr, hScr = autopy.screen.size()
# print(wScr, hScr)

while True:
    # 1. Find hand Landmarks
    success, img = cap.read()
    img = detector.fhand(img)
    lmList, bbox = detector.fPosition(img)
    # 2. Get the tip of the index and middle fingers
    if len(lmList) != 0:
        x1, y1 = lmList[8][1:]
        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))
            # 6. Smoothen Values
            clocX = plocX + (x3 - plocX) / smoothening
            clocY = plocY + (y3 - plocY) / smoothening

            # 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.findDistance(8, 12, img)
            print(length)
            # 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)
```

In [ ]:

In [ ]: