Jurnal Ilmiah Teknik Informatika dan Komunikasi
Halaman Jurnal: http://journal.sinov.id/index.php/juitik/index
Halaman UTAMA Jurnal : https://journal.sinov.id/index.php

Mouse Tracking Tangan dengan Klasifikasi Gestur Menggunakan OpenCV dan Mediapipe

Amanda Muchsin Chalik [a], Bilal Abdul Qowy [b], Faiz Hanafi [c], Ahlijati Nuraminah [d]
[a] Program Studi Ilmu Komputer, a.muchsin.chalik@students.esqbs.ac.id, STIMIK ESQ
b Program Studi Ilmu Komputer, b.abdul.q@students.esqbs.ac.id, STIMIK ESQ
[c] Program Studi Ilmu Komputer, f.hanafi@students.esqbs.ac.id, STIMIK ESQ
[d] Program Studi Ilmu Komputer, ahlijati.nuraminah@esqbs.ac.id, STIMIK ESQ

ABSTRAK

Recognition of human hand gestures is a field that has been widely researched at present, because the detection and recognition of hand gestures has great potential to be used as a way of interacting with computers and controlling them in the future. Hand movements are very useful at this time, because the Covid-19 attack that continues to hit the world requires us not to touch a lot of equipment in public places, and requires us not to have direct contact with objects or with the people around us. This research uses a package from the Python programming language, namely OpenCV and Mediapipe. OpenCV is needed to display visuals in Python programs, and mediapipe is a framework for building a multimodal platform such as video, audio or other running data.

Keywords: Python , OpenCV, Hand Gesture, Mediapipe, MouseTracking.

Abstrak

Pengenalan gerakan tangan manusia merupakan bidang yang banyak diteliti saat ini, karena pendeteksian dan pengenalan gerakan tangan memiliki potensi besar untuk digunakan sebagai cara berinteraksi kepada komputer dan mengendalikannya di masa depan. Pergerakan tangan sangatlah berguna untuk saat ini, karena serangan covid – 19 yang terus melanda dunia , mengharuskan kita untuk tidak menyentuh banyak peralatan yang ada di tempat umum , dan mengharuskan kita untuk tidak berkontak langsung dengan benda ataupun dengan manusia di sekitar kita. Penelitian kali ini menggunakan package dari bahasa pemrograman python yaitu OpenCV dan mediapipe. OpenCV diperlukan untuk memunculkan visual pada program Python, dan mediapipe merupakan sebuah framework untuk membangun sebuah multimodal platform seperti video , audio atau data berjalan lainnya.

Kata Kunci: Python , OpenCV, Hand Gesture, Mediapipe, MouseTracking.

1. PENDAHULUAN

Teknologi dapat membawa perubahan yang dapat mempengaruhi banyak hal, baik dalam arti yang baik ataupun dalam hal yang buruk. Semakin banyak inovasi teknologi yang diciptakan manusia sedemikian rupa dengan fungsi-fungsi yang bermacam-macam yang tentunya akan mempermudah, efisien dan efektifitas suatu kegiatan, teknologi gesture tangan ini dapat berguna kedepannya bagi manusia, dengan tanpa menyentuh sebuah perangkat kita sudah bisa mengoperasikannya dengan sangat mudah.

Penelitian ini membahas mengenai mouse tracking menggunakan tangan tanpa menyentuh mouse atau trackpad pada laptop namun menggunakan sebuah kamera pada device, kemudian device kamera tersebut menangkap gambar secara real time dan melacak atau tracking tangan sehingga cursor pada komputer dapat bergerak mengikuti pergerakan tangan manusia. Program dibuat menggunakan bahasa pemrograman python

Received Mei 10, 2021; Revised Juni 2, 2021; Accepted Juli 27, 2021

dengan package OpenCV dan Mediapipe Implementasi program yang dihasilkan dapat sangat beragam namun difokuskan pada cursor mouse untuk mempermudah pengguna dari sisi usability sehingga lebih interaktif dan lebih mudah digunakan.

2. TINJAUAN PUSTAKA

2.1. OpenCV

OpenCV (Open Source Computer Vision Library) adalah sebuah library open source python yang dikembangkan oleh intel yang berfokus untuk menyederhanakan programming terkait citra digital. Awalnya OpenCV merupakan library yang menggunakan bahasa pemrograman C/C++, dan sekarang telah dikembangkan ke dalam bahasa pemrograman python, java, matlab. OpenCV mempunyai banyak fitur, antara lain: pengenalan wajah, pelacakan wajah, deteksi wajah, kalman filtering dan berbagai jenis metode AI (Artificial Intelligence). OpenCV juga menyediakan berbagai algoritma sederhana terkait Computer Vision untuk low level API.

Selain itu, OpenCV juga menggunakan numpy. Numpy merupakan salah satu library python yang digunakan untuk mengimplementasi array dan matriks multidimensi bersama dengan operasi matematika tingkat tinggi. OpenCV terutama digunakan untuk menangkap data dari video langsung karena OpenCV berfokus pada pemrosesan gambar dan video yang ditangkap. Pada penelitian ini OpenCV difokuskan terutama dalam menangkap objek berupa video. OpenCV mencakup struktur data dasar seperti skalar, point dan lain-lain yang digunakan untuk membangun aplikasi OpenCV. Library OpenCV di import ke bahasa pemrograman python menggunakan kode ‘import cv2’.

2.2. Mediapipe

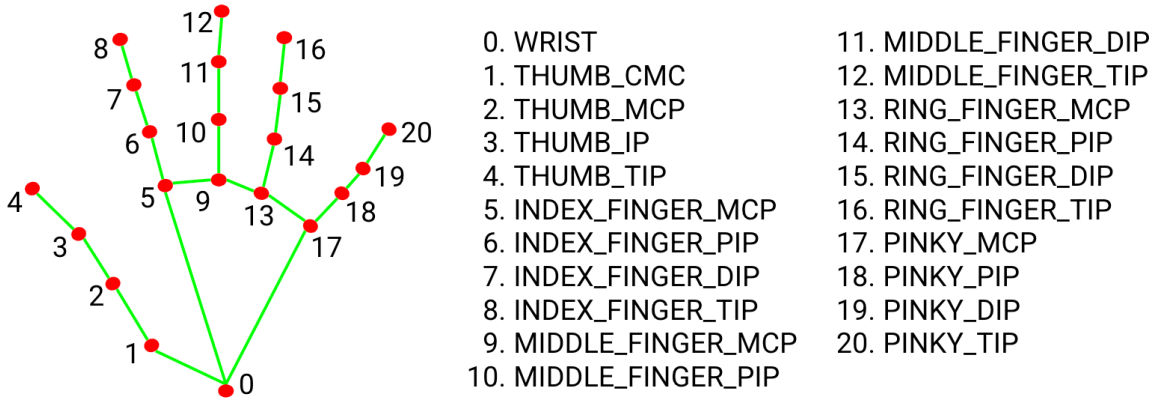
Mediapipe merupakan suatu framework open source untuk membangun sebuah aplikasi multimodal atau berupa video, gambar, audio, dan data lainnya. Framework ini dibuat oleh Google.Inc dengan tujuan memudahkan developer dalam membangun sebuah aplikasi di bidang computer vision seperti kecerdasan buatan dan image processing. Framework ini dapat dijalankan pada platform Android , iOS , JavaScript, dan Python. Selain itu Mediapipe memiliki beragam package atau framework seperti face mesh, object Mediapipe dapat diinstal pada Python melalui syntax pip install mediapipe melalui terminal device yang digunakan, bisa juga install package pada IDE yaitu pycharm secara otomatis.

2.3. Hand Tracking

Mediapipe Hands adalah solusi dari pelacakan tangan dan jari dengan sangat akurat. Fitur ini menggunakan machine learning untuk menyimpulkan 21 titik pada 3D landmark dari sebuah tangan dari satu frame secara real time. metode ini mencapai performa untuk menjalankan tracking secara langsung pada perangkat mobile, dan bahkan bisa mendeteksi lebih dari satu tangan.

Setelah deteksi telapak tangan di seluruh gambar, model telapak tangan kami kemudian dilakukan pelokalan titik kunci yang tepat dari 21 koordinat buku jari 3D di dalam wilayah tangan yang terdeteksi melalui regresi, yaitu prediksi koordinat langsung. Model ini mempelajari representasi pose tangan internal yang konsisten dan kuat bahkan untuk tangan yang terlihat sebagian dan oklusi diri.

Untuk mendapatkan data kebenaran dasar, dilakukan pelabelan keterangan secara manual ~30K gambar dunia nyata dengan 21 koordinat 3D, seperti yang ditunjukkan di bawah ini (diambil nilai Z dari peta kedalaman gambar, jika ada per koordinat yang sesuai). Untuk menutupi kemungkinan pose tangan dengan lebih baik dan memberikan pengawasan tambahan pada sifat geometri tangan, juga dibuat model tangan sintetis berkualitas tinggi di berbagai latar belakang dan memetakannya ke koordinat 3D yang sesuai.



Gambar 1. Hand Landmarks

2.4. Mouse Pointer

Autopy adalah cross platform atau library otomatis pada bahasa pemrograman python secara sederhana. autopy berfungsi menjalankan dan mengontrol system hardware seperti fungsi keyboard, mouse, mencari warna bitmaps pada gambar, dan menampilkan peringatan. pada project ini autopy yang kami fokuskan adalah pada mouse , dimana akan diintegrasikan dengan handtracking mediapipe sehingga mouse pointer dapat bergerak sesuai dengan perintah pada kamera atau tangan yang bergerak.

2.5. Pengolahan Citra Digital

Pengolahan citra digital merupakan disiplin ilmu yang didalamnya mempelajari teknik cara mengolah citra. Citra digital merupakan sebuah pengolahan citra yang dilakukan dengan menggunakan cara digita yang mengubah intensitas cahaya menjadi dua dimensi. proses transformasi 3 dimensi menjadi 2 dimensi agar supaya dapat menghasilkan citra yang nantinya akan dipengaruhi oleh banyak faktor yang mungkin akan mengakibatkan citra penampilan dari sebuah benda tidak akan sama persis dengan bentuk asli fisiknya.

Citra merupakan sebuah matriks yang terdiri dari baris dan kolom yang mana setiap pasangan indeks pada matriks dapat menyatakan satu titik pada citra untuk menyatakan nilai kecerahan pada titik tersebut. Titik-titik tersebut dinamakan sebagai elemen citra atau piksel yang dapat didefinisikan menjadi fungsi dari 2 variabel x dan y.

3. METODOLOGI PENELITIAN

Komputer pada dasarnya dapat dioperasikan karena adanya komunikasi antar manusia dengan komputer. Manusia dapat menciptakan dan memberikan perintah apa yang diinginkan terhadap komputer, kemudian komputer tersebut menjalankankannya sesuai dengan perintah yang diberikan oleh penggunanya. Hand gesture recognition atau disebut dengan pengenalan isyarat tangan merupakan interaksi antar manusia dan komputer (mesin). Hal yang paling mendasar dalam pengenalan isyarat tangan adalah dengan membuat interaksi yang alami (sesuai kebiasaan) antara manusia dan komputer yang mana pengenalan isyarat tersebut dapat digunakan untuk mengontrol mesin atau menyampaikan suatu informasi.

Sistem arsitektur mengenai hand gesture ini adalah penangkapan gambar, pergerakan tangan (hand tracking) dan pengenalan isyarat tangan dan control mouse. Sistem pengenalan isyarat tangan (hand gesture recognition) pada kali ini diperuntukkan untuk mouse tracking terhadap interaksi antara manusia dengan komputer.

4. HASIL DAN PEMBAHASAN

4.1 Instalasi dan Konfigurasi

- Install opencv package
  Silahkan masukkan kode program "*pip install opencv-python*" pada command prompt anaconda kemudian *Enter*. Tunggu hingga proses selesai.
- Install numpy package
  Silahkan masukkan kode program "*pip install numpy*" pada command prompt anaconda kemudian *Enter*. Tunggu hingga proses selesai.
- Install mediapipe package
  Silahkan masukkan kode program "*pip install mediapipe*" pada command prompt anaconda kemudian *Enter*. Tunggu hingga proses selesai.
- Install autopy package
- Silahkan masukkan kode program "*pip install autopy*" pada command prompt anaconda kemudian *Enter*. Tunggu hingga proses selesai.
- Buka package spyder (python 3.8). Buat file baru dengan nama *MouseTrack.py*. Masukkan kode berikut pada code editor

```
import time
import cv2 as cv
import HandTrackkk as ht
import numpy as np
import autopy

wCam, hCam = 640, 480
smoothening = 7

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

cap = cv.VideoCapture(1)
cap.set(3, wCam)
cap.set(4, hCam)
detector = ht.handDetector(maxHands=1)
wScr, hScr = autopy.screen.size()

while True :
    #1. find hand Landmarks
    success, img = cap.read()
    img = detector.findHands(img)
    lmList, bbox = detector.findPosition(img)


    #2. ambil index jari tengah dan telunjuk
    if len(lmList) !=0 :
        x1, y1 = lmList[8][1:]
        x2, y2 = lmList[12][1:]

        #print(x1, y1, x2, y2)
    #3. check jari apa yang naik
    finger = detector.fingersUp()
    print(finger)

    cv.rectangle(img, (frameR, frameR), (wCam - frameR, hCam - frameR),
                 (255, 0, 255), 2


    #4. cmn jari index : gerak mode
    if finger [1] == 1 and finger [2] == 0:
```

```
    #5. merubah kordinat
        x3 = np.interp(x1, (frameR, wCam - frameR), (0, wScr))
        y3 = np.interp(y1, (frameR, hCam - frameR), (0, hScr))
    #6. smooten values
        clocX = plocX + (x3 - plocX) / smoothening
        clocY = plocY + (y3 - plocY) / smoothening
    #7. gerakin mouse
        autopy.mouse.move(wScr - clocX, clocY)
        cv.circle(img, (x1, y1), 15, (255, 0, 255), cv.FILLED)
        plocX, plocY = clocX, clocY
    #8. clik mode
    if finger [1] == 1 and finger [2] == 1:

         #9. cari jarak
        length, img, lineInfo = detector.findDistance(8, 12, img)
        print(length)

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


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

    #12. tampilin
            cv.imshow("cam", img)
            cv.waitKey(1)
```

- Buat kembali file baru di folder yang sama dengan nama “HandTrack.py” Masukkan kode berikut pada kode editor

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


class handDetector():
    def __init__(self, mode=False, maxHands=1, modelComplexity=1,
detectionCon=0.5, trackCon=0.5):
            self.mode = mode
            self.maxHands = maxHands
            self.modelComplex = modelComplexity
            self.detectionCon = detectionCon
            self.trackCon = trackCon

            self.mpHands = mp.solutions.hands
            self.hands = self.mpHands.Hands(self.mode, self.maxHands,
self.modelComplex, 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)

            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):

                    h, w, c = img.shape
                    cx, cy = int(lm.x * w), int(lm.y * h)
                    xList.append(cx)
                    yList.append(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, xmin

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

      def fingersUp(self):
          jari = []
          #thumb
          if self.lmList[self.tipIds[0]][1]> self.lmList[self.tipIds[0]-1][1]:
              jari.append(1)
          else :
              jari.append(0)

          #fingers
          for id in range(1, 5):

           if self.lmList[self.tipIds[id]][2] < self.lmList [self.tipIds[id] -
2][2]:
                  jari.append(1)
              else :
                  jari.append(0)
```

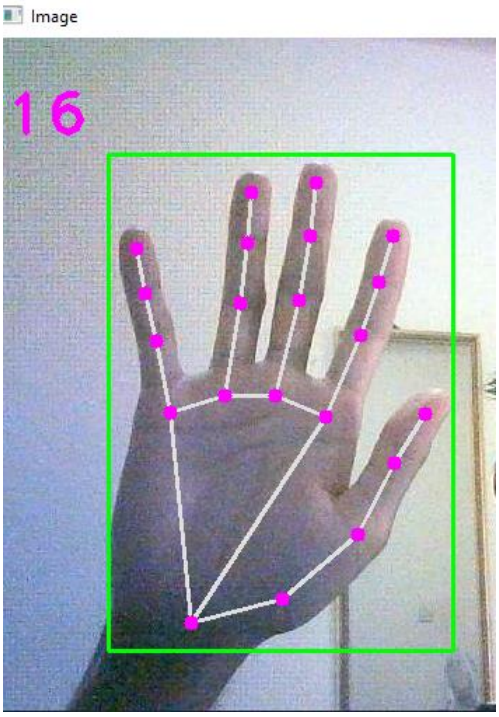
```
        return jari
    def findDistance(self, p1, p2, img, draw=True,r=15, t=3):
        x1, y1 = self.lmList[p1][1:]
        x2, y2 = self.lmList[p2][1:]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

        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]

    def main():
        pTime = 0
        cTime = 0
        cap = cv2.VideoCapture(1)
        detector = handDetector()
        while True:
            success, img = cap.read()
            img = detector.findHands(img)
            lmList, bbox = detector.findPosition(img)
            if len(lmList) != 0:
                print(lmList[4])

            cTime = time.time()
            fps = 1 / (cTime - pTime)
            pTime = cTime

            cv2.putText(img, str(int(fps)), (10, 70),
    cv2.FONT_HERSHEY_PLAIN, 3,
                        (255, 0, 255), 3)

            cv2.imshow("Image", img)
            cv2.waitKey(1)


    if __name__ == "__main__":
        main()
```

- Run kode program
- Lakukan testing pada objek

4.2 Pengujian

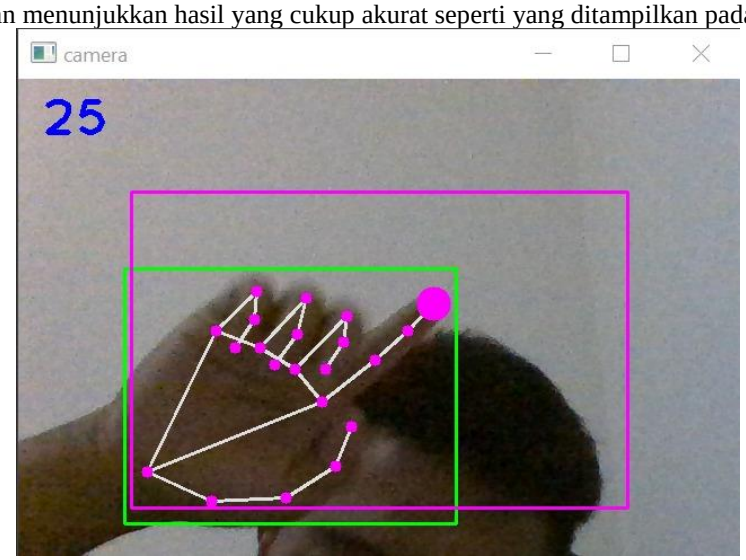
Testing pada Hand

Testing pada tangan menunjukkan hasil yang cukup akurat seperti yang ditampilkan pada Gambar 2



Gambar 2. Pengujian pada tangan

Testing pada Mouse Pointer
Testing pada tangan menunjukkan hasil yang cukup akurat seperti yang ditampilkan pada Gambar 3



Gambar 3. Pengujian pada Mouse Pointer

5. KESIMPULAN DAN SARAN

Dalam analisis yang telah dilakukan, dapat disimpulkan bahwa program dapat berjalan dengan baik, namun terdapat sedikit issue pada mouse pointer, terdapat baris kode yang tidak bisa dijalankan namun masih memungkinkan untuk seluruh program dapat berjalan. Berdasarkan hasil tersebut dapat disimpulkan bahwa program yang dihasilkan dapat implementasikan dalam project khusus lainnya yang membutuhkan mouse pointer dengan gesture controlling.

DAFTAR PUSTAKA

[1] Fadli,Feri dan Munawir. 2019. Kontrol Mouse Menggunakan Webcam Berdasarkan Deteksi Warna. Aceh

[2] Sunyoto, Andi dan Agus Harjoko. 2004.Review Teknik, Teknologi, Metodologi dan Implementasi Pengenalan Gestur Tangan Berbasis Visi. Yogyakarta.

[3] Makahaube, Stella Stelani, Alwin Melkie Sambul dan Sherwin RU Sompie. (2021).Implementation of Gesture Recognition Technology for Automated Education Service Kiosk.Manado.

[4] Varun, Kollipara Sai, I. Puneeth, dan Prem Jacob. 2011. Virtual Mouse Implementation using Open CV. India.

[5] Yunita,Helda dan Endang Setyati. 2019. Hand Gesture Recognition Sebagai Pengganti Mouse Komputer Menggunakan Kamera. Banjarmasin.