**Nama : Asri Indah Pertiwi (2006111)**

**Sarah Khoerunisa (2006050)**

**Kelas : Pengolahan Citra Digital B**

***Image Processing***

**Laporan Analisis, Hasil Percobaan dan Pengembangan**

Program dalam melakukan image processing dan segmentasi dengan menggunakan Bahasa Python dengan aplikasi Visual Studio Code menggunakan library os, mediapipe, open cv, time dan *file* *hand traking module*.

Citra jarimatika yang dibuat ini menggunakan dataset yang dicari sendiri melalui beberapa sumber. Pada jurnal rujukan tahapan yang dilakukan dalam pengolah citra ini adalah *enhancement* dengan menggunakan filter lolos rendah dengan fungsi *Gaussian Blur*, *GrayScale*, deteksi tepi dengan Canny, deteksi *countour*, dilatasi untuk menebalkan tepian *countour* dan terakhir dilanjutkan dengan melakukan proses ekstraksi ciri dengan Moment Hu. Hasil dari program pada jurnal rujukan tersebut adalah dapat mendeteksi tepian bentuk jari dengan metode segmentasi dengan Canny menujukkan kejelasan tegas dari tepian obyek, dengan metode deteksi *contou*r menunjukan bahwa citra terlihat jelas bentuk dari obyek jarinya masih ada beberapa kelas sulit teridentifikasi, dan Momen Hu memberikan hasil informasi ciri dari *contour* jari yang dapat membantu untuk pembelajaran bentuk jarimatika. Namun penulis melakukan pengembangan yaitu menjadi deteksi jari dengan opencv, dimana tidak hanya dapat mendeteksi tepian dan mengekstraksi ciri pada program yang dibuat ini dapat mengenali bentuk dan berapa nilai dari jari yang dideteksi.

Pada langkah awal dilakukan penginstalan python versi 3.8, kemudian dilanjut dengan instal pip dan mediapipe, setelah itu baru lah bisa dilanjutkan dengan membuat kode program. Untuk program ini hanya bisa dilakukan dengan menggunakan python versi 3.8, jika menggunakan versi lain mediapipe tidak akan bisa terinstal dan program tidak akan berhasil dijalankan.

Berikut merupakan sourcode dari proram yang dibuat:

Sourcode finger *countour*

|  |
| --- |
| import cv2  import time  import os  import HandTrackingModule as htm  wCam, hCam = 640, 480  cap = cv2.VideoCapture(1)  cap.open(0,cv2.CAP\_DSHOW)  cap.set(3, wCam)  cap.set(4, hCam)  folderPath = "foto"  myList = os.listdir(folderPath)  print(myList)  overlayList = []  for imPath in myList:      image = cv2.imread(f'{folderPath}/{imPath}')      # print(f'{folderPath}/{imPath}')      overlayList.append(image)  print(len(overlayList))  pTime = 0  detector = htm.handDetector(detectionCon=0.75)  tipIds = [4, 8, 12, 16, 20]  while True:      success, img = cap.read()      img = detector.findHands(img)      lmList = detector.findPosition(img, draw=False)      # print(lmList)      if len(lmList) != 0:          fingers = []          # Thumb          if lmList[tipIds[0]][1] > lmList[tipIds[0] - 1][1]:              fingers.append(1)          else:              fingers.append(0)          # 4 Fingers          for id in range(1, 5):              if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:                  fingers.append(1)              else:                  fingers.append(0)          # print(fingers)          totalFingers = fingers.count(1)          print(totalFingers)          h, w, c = overlayList[totalFingers - 1].shape          img[0:h, 0:w] = overlayList[totalFingers - 1]          cv2.rectangle(img, (20, 225), (170, 425), (0, 255, 0), cv2.FILLED)          cv2.putText(img, str(totalFingers), (45, 375), cv2.FONT\_HERSHEY\_PLAIN,                      10, (255, 0, 0), 25)      cTime = time.time()      fps = 1 / (cTime - pTime)      pTime = cTime      cv2.putText(img, f'FPS: {int(fps)}', (400, 70), cv2.FONT\_HERSHEY\_PLAIN,                  3, (255, 0, 0), 3)      cv2.imshow("Image", img)      cv2.waitKey(1) |

Sourcode *hand tracking module*

|  |
| --- |
| import cv2  import mediapipe as mp  import time      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        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):            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)                  # print(id, cx, cy)                  lmList.append([id, cx, cy])                  if draw:                      cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)            return lmList      def main():      pTime = 0      cTime = 0      cap = cv2.VideoCapture(0)      detector = handDetector()      while True:          success, img = cap.read()          img = detector.findHands(img)          lmList = 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() |













