Aplikasi *Image Processing* Untuk Menentukan Volume Gelas Secara Otomatis Pada Mesin Kopi

Dibuat oleh:

Muhammad Naufal Fadhilah

10220059

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

JURUSAN FISIKA

TAHUN AJARAN 2023/2024

I. Pedahuluan

Aplikasi ini merupakan solusi inovatif untuk memastikan volume yang akurat dalam penggunaan mesin kopi. Dengan memanfaatkan teknologi Image Processing, sistem ini mampu secara otomatis mengukur dan menentukan volume gelas tanpa intervensi manusia. Prosesnya dimulai saat gelas ditempatkan di bawah mesin kopi, dan kamera terpasang akan mengambil gambar gelas.

Melalui algoritma Image Processing yang canggih, aplikasi ini dapat mengenali tepi dan dimensi gelas. Informasi volume yang diperoleh dari citra digunakan untuk mengatur dosis kopi yang disajikan, memastikan bahwa setiap minuman yang dihasilkan sesuai dengan ukuran yang diinginkan.

I.1. Latar Belakang

Industri kopi berkembang pesat dan memiliki tantangan tersendiri dalam menjaga konsistensi dan kualitas penyajian minuman kepada pelanggan. Meskipun mesin kopi semakin otomatis, masalah konsistensi volume gelas masih menjadi isu penting. Setiap gelas yang dihidangkan harus memiliki volume yang tepat untuk memastikan bahwa setiap tegukan menyajikan pengalaman kopi yang konsisten.

Dalam banyak kasus, penyajian kopi masih melibatkan pengukuran manual volume gelas, yang rentan terhadap variasi dan kesalahan manusia. Pengukuran yang tidak konsisten dapat memengaruhi rasa kopi dan pengalaman pelanggan secara keseluruhan. Oleh karena itu, diperlukan solusi otomatis yang dapat memberikan pengukuran volume gelas yang akurat dan konsisten.

I.2. Tujuan

Tujuan dari penelitian ini adalah untuk membuat mesin kopi cerdas yang dapat mengisi gelas secara otomatis dengan takaran dan volume yang tepat berdasarkan size atau ukuran dari gelas yang dipakai.

I.3. Manfaat

Manfaat dari penelitian ini, yaitu:

1. Meningkatkan kualitas pelayanan produk dengan menyajikan kopi dengan takaran yang konsisten
2. Meningkatkan efisinesi operasional dengan mengurangi keterlibatan manual dalam pengukuran

II. Teori

II.1. Canny Edge Detection

Titik-titik pada gambar yang mengalami perubahan kecerahan secara tiba-tiba disebut tepi. Titik tepi biasanya mencakup batas objek dan jenis perubahan kecerahan lainnya, serta tepi kebisingan. Deteksi tepi adalah teknik image processing yang digunakan untuk menidentifikasi titik pada gambar digital melalui dikontinuitas intensitas kecerahan gambar. Tepinya juga direpresentasikan sebagai kumpulan titik-titik yang saling berhubungan yang terletak di perbatasan antara dua area. Tepinya menggambarkan batas objek, dan oleh karena itu fitur ini dapat digunakan untuk mengelompokkan gambar ke area atau objek utamanya.

Canny edge detection adalah teknik mengekstrasi informasi struktur penting pada suatu objek gambar dan mengurangi banyak data yang diproses pada gambar. Proses algoritma deteksi tepi Canny terdapat lima tahapan, yaitu menghaluskan gambar menggunakan Gaussian filter, mencari intesitas gradien pada gambar, lakukan double threshold untuk menentukan potensi tepi, dan melakukan hysteresis.

II.1.1. Gaussian Filter

Semua hasil deteksi tepi biasanya rentan terpengaruh dengan noise pada gambar sehingga diperlukan pemfilteran semua noise tersebut yang dapat menimbulkan deteksi tepi yang tidak diperlukan. Persamaan dari kernel Gaussian filter untuk ukuran (2k+1)x(2k+1) sebagai berikut

$$\text{new\_image} = \frac{1}{159}\begin{bmatrix} 2 & \cdots & 2 \\ 5 & \ddots & 5 \\ 2 & \cdots & 2 \end{bmatrix} * \text{old\_image} \qquad \text{II.1}$$

Hal penting yang perlu dimengeri saat menentukan size dari Gaussian kernel akan mempengaruhi performa dari detector. Semakin besar ukuran kernel, semakin rendah sensitivitas detektor terhadap noise, tetapi kesalahan lokalisasi untuk mendeteksi tepi akan sedikit meningkat. filter Gaussian 5×5 adalah ukuran yang paling tepat untuk sebagian besar kasus, tetapi ukurannya juga bervariasi tergantung pada situasi tertentu.

II.1.2. Besar Intensitas dan Arah Gradien Pada Gambar

Tepi pada gambar merupakan kumpulan titik dengan variasi arah, jadi algoritma Canny menggunkana empat filter untuk medeteksi tepi horizontal, vertical, dan diagonal pada gambar. Operator deteksi tepi mengembalikan nilai turunan pertama dari arah horizontal (G_x) dan arah vertikal (G_y).

$$G_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \text{kernel pixel}$$

$$G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \text{kernel pixel} \qquad \text{II.2}$$

Besar (G) dan arah (Θ) gradien tipe dapat ditentukan dengan

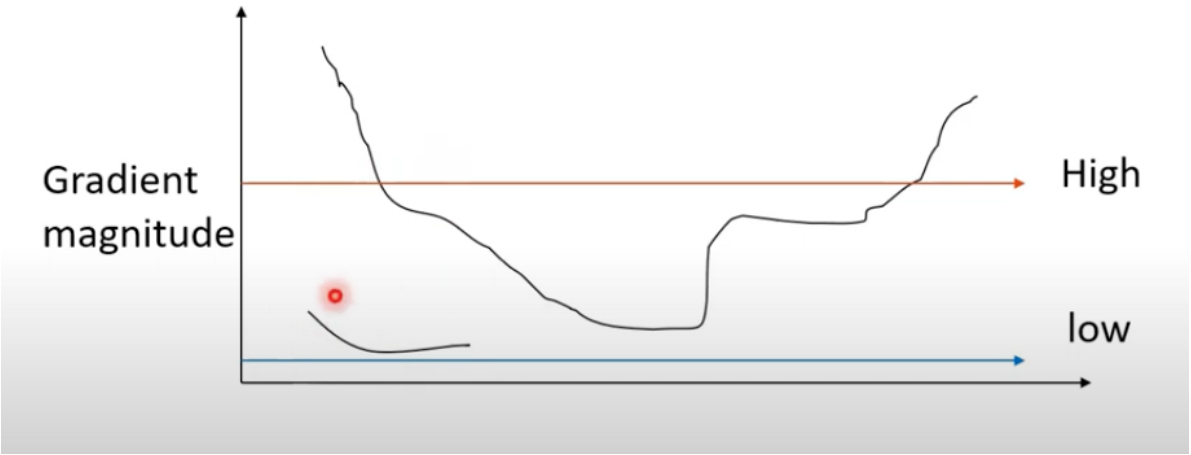
$$G = \sqrt{G_x^2 + G_y^2}$$
$$\Theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \quad \text{II.1}$$

II.1.3. Penekanan atau Threshold dan Hysteresis

Threshold pada canny edge detection merupakan salah satu fitur untuk menentukan seberapa banyak tepi yang akan dihasilkan pada suatu gambar. Hal ini cukup penting karena terkait dengan tujuan yang ingin dicapai oleh pengguna edge detection. Sebagai contoh apabila kita melakukan face recognition pada wajah, maka perlu kita mengambil semua tepi pada bagian pentingnya, seperti mata, hidung, mulut, telingan dan lain sebagainya agar kita bisa membedakan wajah satu orang dengan yang lainnya. Namun, apabila kita ingin mengenali struktur atau bentuk dari buah, maka kita gak perlu sampai sedetail wajah hingga mencari sampai ke pori-pori dari buah cukup dari bentuk luarnya, maka kita sudah bisa membuat prediksi yang akurat. Threshold dibagi menjadi dua, yaitu upper threshold dan lower threshold, fungsi dua jenis ini adalah sebagai berikut

1. Jika intesitas gradient pixel lebih dari upper threshold, maka pixel tersebut dapat diterima sebagai tepi
2. Jika intesitas gradient pixel kurang dari lower threshold, maka pixel tersebut dapat ditolak sebagai tepi
3. Jika intesitas gradient pixel berada di antara upper threshold dan lower threshold, maka pixel tersebut hanya akan dapat diterima apabila dia tersambung dengan pixel yang melebih batas upper threshold

Seperti yang kita lihat pada **gambar I.1**, apabila nilai dari gradient magnitude lebih dari Hight atau upper threshold, maka nilai pixel tersebut akan mejadi edge atau tepi, sedangkan untuk nilai gradient di bawah low atau lower threshold, maka nilai tersebut otomatis akan disingkirkan. Lalu, bagaimana dengan nilai di antara upper threshold dan lower threshold? Disinilah kita menemukan istilah hysteresis, dimana nilai dari gradient akan tetap dipertahankan pada area ini selama gradient tersebut bersambung dengan pixel yang memiliki gradient di atas upper threshold.

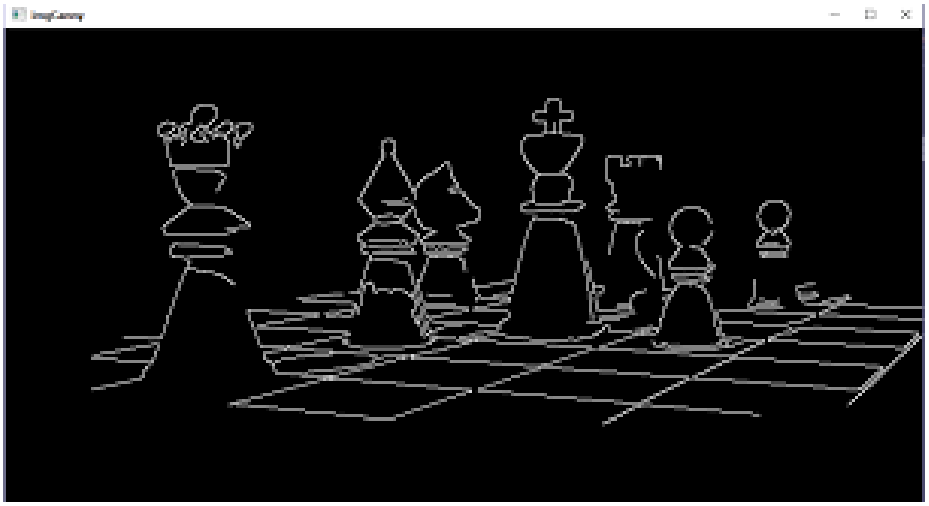


Gambar I.1. Hyteresis Thresholding

III. Metode

III.1. Menentukan Kontur Pada Objek Untuk Dideteksi

Kontur dapat kita temukan menggunakan modul yang sudah tersedia pada python, tetapi sebelum itu, kita perlu melakukan image processing pada gambar yang ingin kita carikan kontur hingga gambar yang kita miliki hanya tersisa bilangan binary (image canny) pada bagian tepi gambar menggunakan *canny edge detector*, seperti yang sudah dijelaskan pada dasar teori **bab II.1.1**. *Canny edge* untuk berbagai tepi pada gambar, seperti yang dapat lihat pada **gambar III.1**. Hal ini diperlukan sebagai argument pada fungsi **findContour()** pada OpenCV.

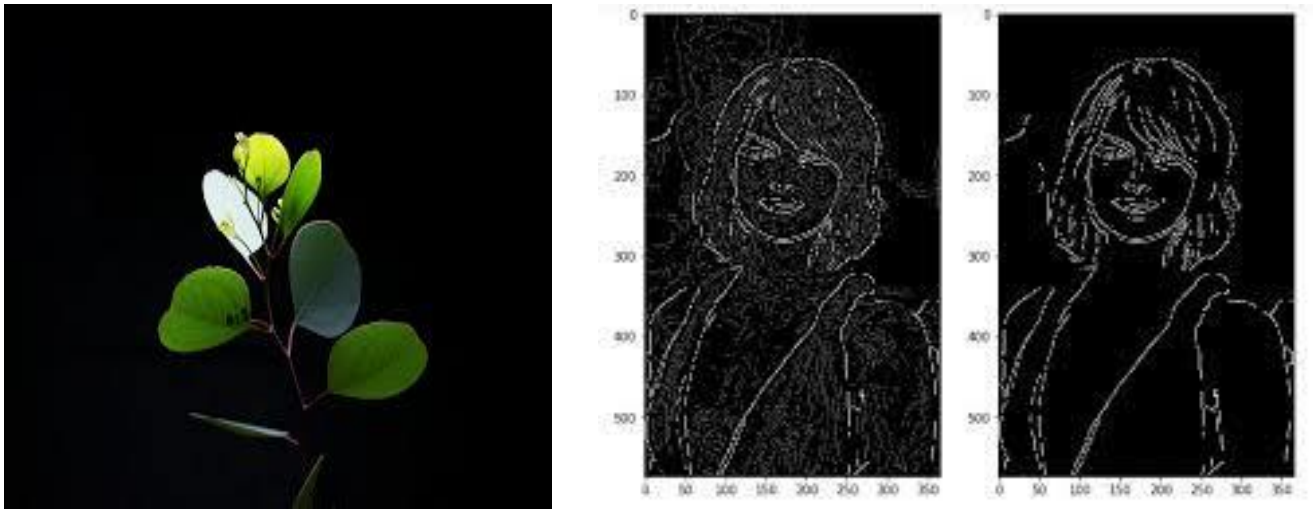


Gambar III.1 image canny

III.2. Memfilter Kontur

Dalam mendeteksi kontur, kita memerlukan beberapa kondisi khusus untuk mencegah pendeteksian kontur secara berlebihan dan hanya fokus pada benda yang ingin kita deteksi. Untuk meminimalisir ini, kita dapat melakukannya dengan

membuat background yang homogen dan menentukan threshold gradient pixel. Background yang homogen dapat memudahkan kita untuk menentukan kontur pada objek karena prinsip dari *canny edge detection* adalah dengan melihat selisih intesitas gradient pixel. Setelah tepi dibuat, maka kita perlu memfilter lagi karena banyaknya tepi yang dibuat pada image akan sangat sulit untuk diidentifikasi begitu juga apabila tepi terlalu sedikit dan tidak berbentuk. Hal ini perlu dilakuakn dengan menerapkan threshold dan hysteresis.



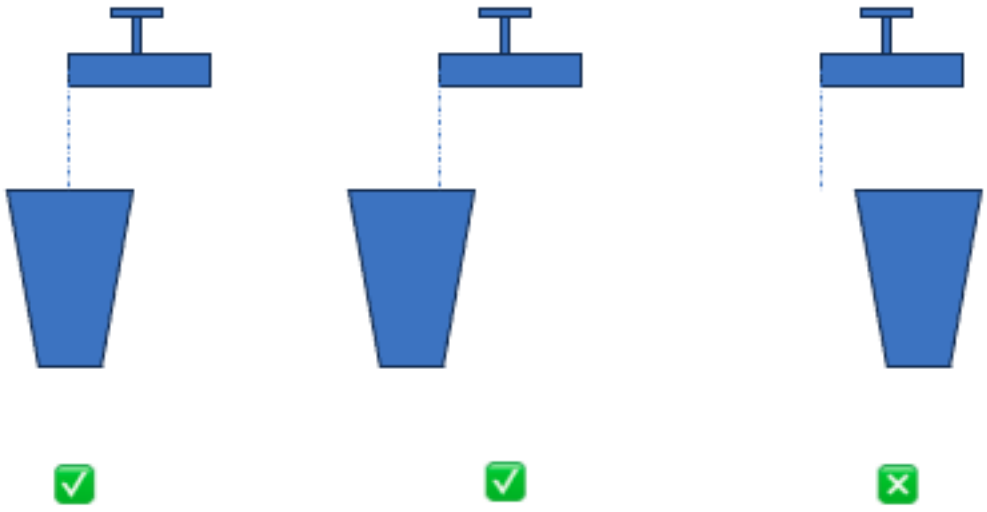
(a) (b)

Gambar III.2 (a) contoh gambar dengan background homogen dan (b) image canny dengan upper threshold yang rendah (kiri) dan upper threshold yang tinggi (kanan)

III.3. Mendeteksi posisi gelas hingga tepat berada di bawah keran

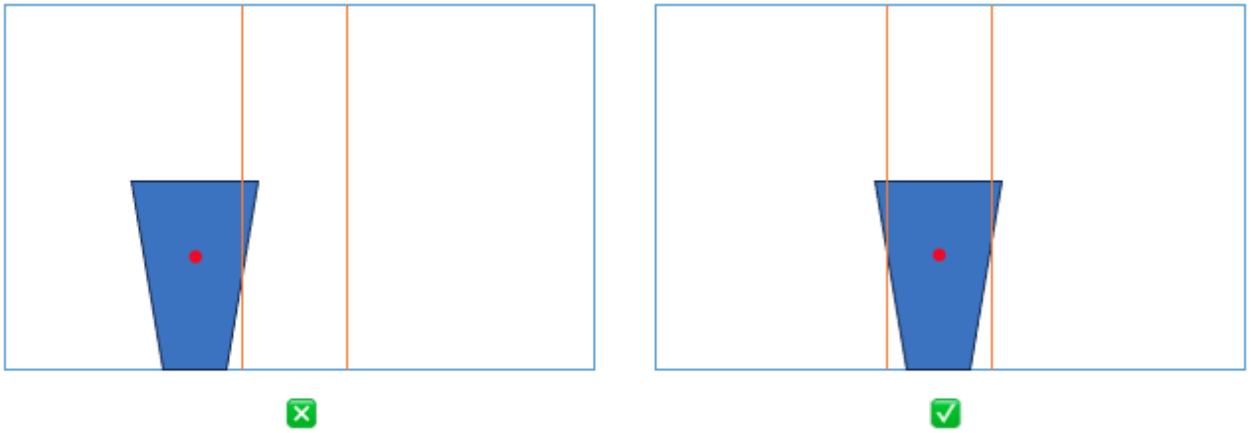
II.3.1. Menempatkan gelas hingga tepat berada di tengah kamera

Posisi keran disesuaikan tepat berada di tengah kamera sehingga posisi gelas harus menyesuaikan. Karena gelas memiliki lebar, maka posisi titik tengah dari gelas tidak harus tepat dengan titik tengah kamera. Dengan ini kita bisa membuat rentang tertentu untuk posisi gelas selama dia tidak keluar dari jangkauan lebar bagian atas gelas, tempat dimana air akan masuk terhadap titik tengah kamera, seperti yang terlihat pada **gambar III.3**.



Gambar III.3 skema rentang posisi gelas yang diperkenankan dan yang tidak diperkenankan

Hal ini bisa dilakukan dengan menerapkan sedikit program pada python, yaitu dengan mengambil posisi center dari kontur dan menciptakan batas maksimum dan minium dari kontur, seperti yang terlihat pada **gambar II.4**.



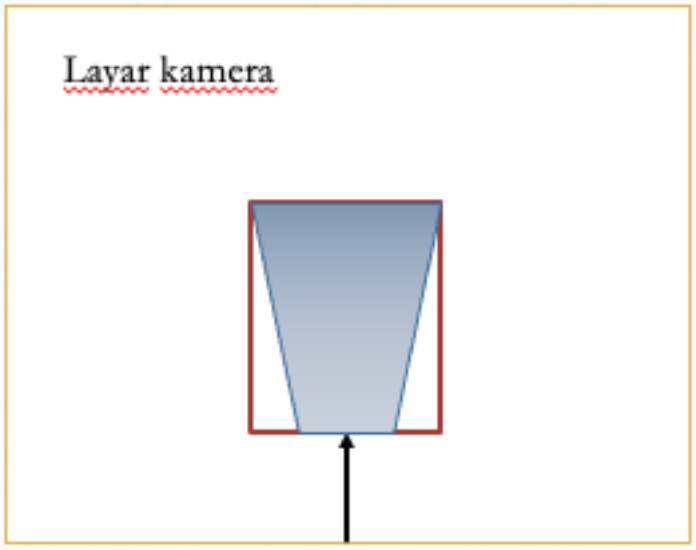
Gambar III.4. Ilustrasi posisi center dari kontur yang di dalam dan di luar jangkauan keran

III.4. Menentukan Volume Gelas

Penentuan tugas volume dilakukan dengan menyediakan sampel gelas yang berbeda ukuran area box konturnya, yang nanti akan dipakai untuk pendeteksian gelas dengan ukuran yang berbeda dan menghasilkan output waktu terbukanya keran.

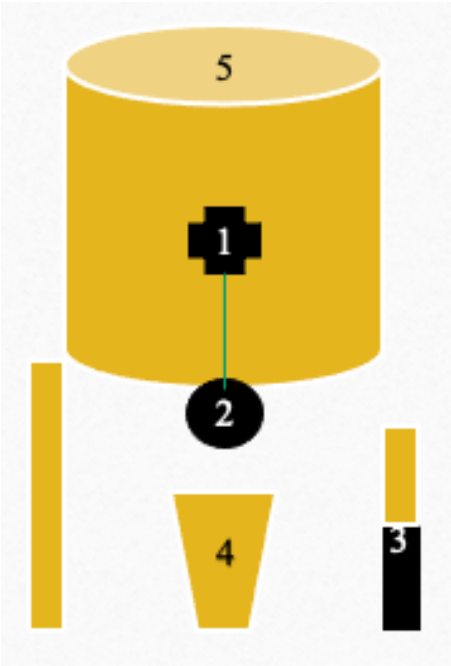
III.5. Kalibrasi Jumlah Pixel Pada Kamera Terhadap Jarak Gelas ke Kamera

Kalibrasi ini dilakukan dengan mencatat perubahan nilai jumlah pixel dengan variasi jarak antara kamera dengan objek.Seperti yagn terlihat pada **gambar II.4**, jarak pixel dari kamera ke gelas dihitug dari bawah layar kamera ke gelas. Jika jarak asli (dalam cm) berubah, maka jarak pixel juga akan berubah. Dari data inilah, nanti dikalibrasi, kemudian diplot dan dapatkan persamaannya.



Gambar III.5. Jarak pixel dari kamera ke gelas

III.6. Desain Hardware



Gambar III.6. Desain hardware dari mesin kopi

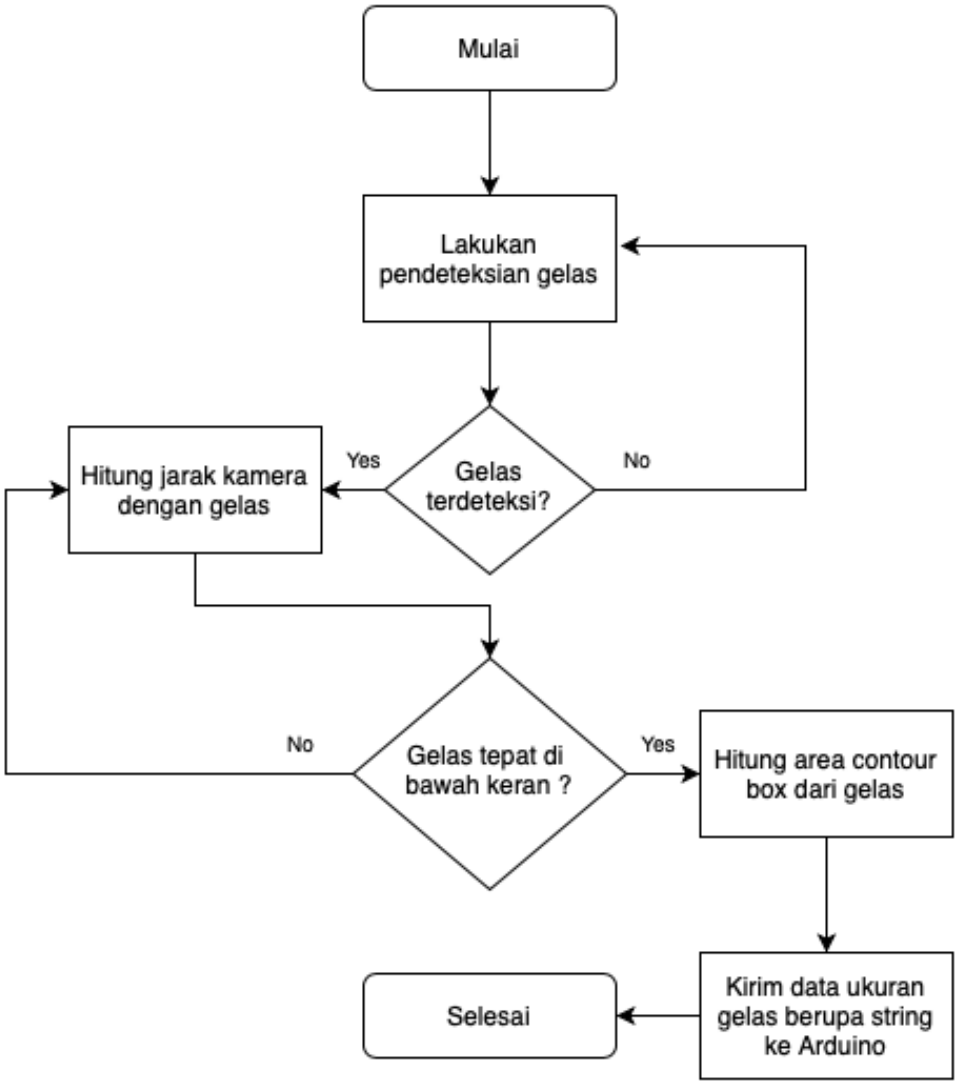
Masing-masing alat dan bahan yang ditunjukkan oleh nomor pada **gambar III.6**, yaitu:

1. Servo motor

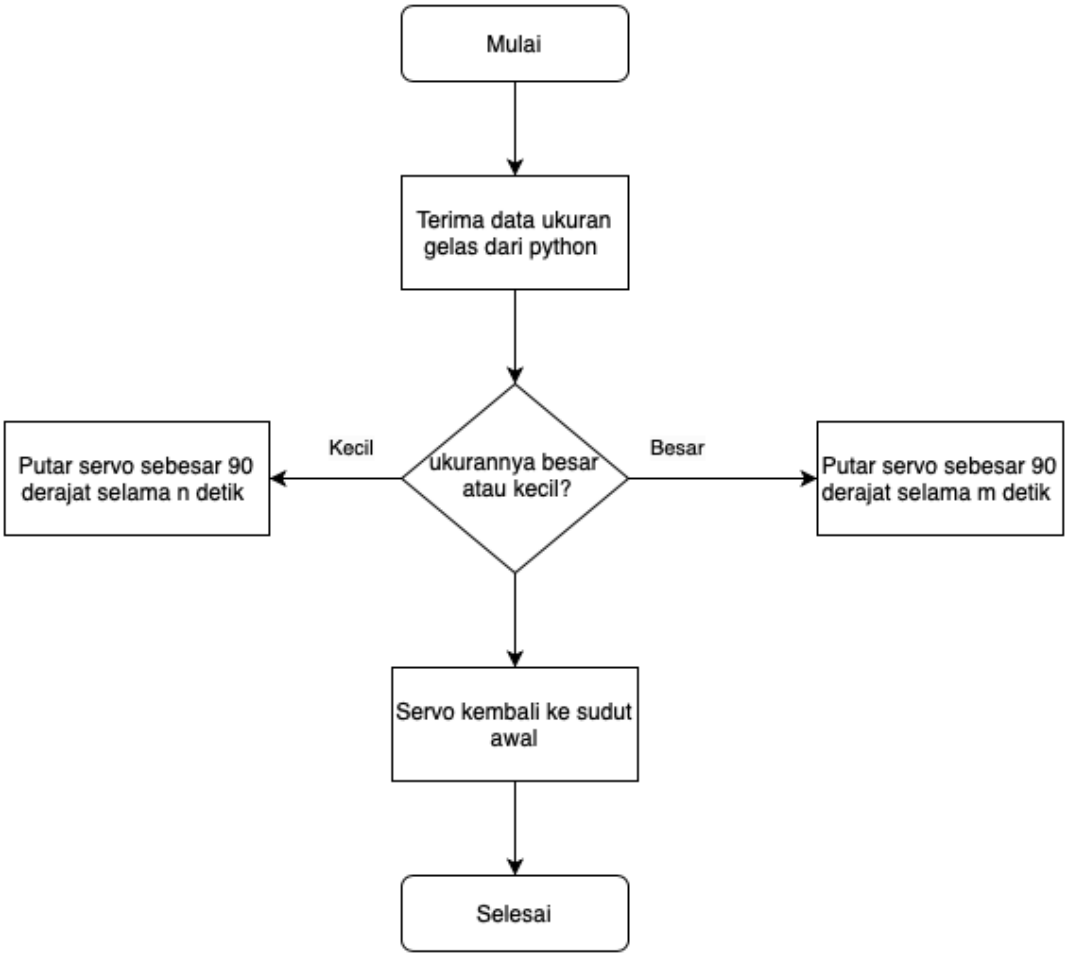
   Digunakan sebagai actuator untuk membuka keran
2. Keran
3. Handphone

   Digunakan sebagai kamera untuk mendeteksi gelas
4. Gelas platik
5. Tangki air

III.7. Flowchart Algoritma Pemrograman Python



Gambar II.4 Diagram alur pemrograman python untuk cara kerja mesin kopi otomatis



Gambar II.5 Diagram alur pemrograman Arduino untuk cara kerja mesin kopi otomatis

IV. Hasil Dan Pembahasan.

IV.1. Kalibrasi Jumlah Pixel Terhadap Jarak Kamera dengan Objek

Seperti yang terlihat pada **gambar III.1**, persamaan yang dihasilkan untuk mendapatkan jarak pixel pada kamera adalah $y = 6.2121x - 25.661$ dengan keliniearan R^2 sebesar 0.9987, dimana y adalah jarak pixel di kamera dan x adalah jarak dari kamera ke objek dalam cm.

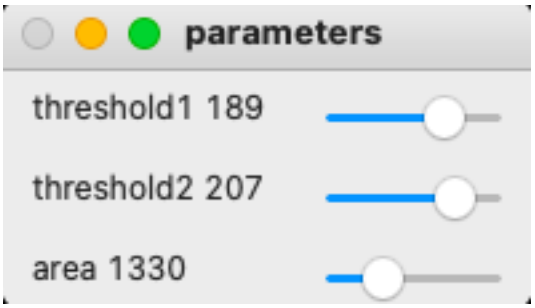
Gambar IV.1 Grafik hasil kalibrasi jumlah pixel terhadap jarak kamera dengan objek

IV.2. Memfilter Deteksi Tepi

Melakukan filtering sebelum dilakukan uji coba sangatlah penting karena akan membuat deteksi gelas menjadi tidak jelas. Seperti yang terlihat pada **gambar IV.5 (b)**, ukuran luas kontur atau box area tidak sesuai dengan ukuran gelasnya sehingga membuat tahap penentuan volume menjadi tidak konsisten. Apabila saya membuat

klasifikasi gelas yang lebih besar dibandingkan dengan gelas yang terdeteksi pada **gambar IV.5 (b)**, maka klasifikasi akan berpindah dan volume air yang akan dikeluarkan akan melebihi gelas yang kita taruh di bawah keran. Selain itu, apabila kontur melebar secara vertikal ke bawah, maka perhitungan jarak juga akan terganggu.

Untuk mengatasi masalah ini, maka diperlukan background homogen, seperti yang terlihat pada **gambar IV.3**, terdapat background berwarna kuning untuk menimalisir error kontur akibat variasi warna pada background. Pada program Python juga dibuat TrackBar untuk mengubah nilai upper dan lower threshold secara real time, seperti yang terlihat pada **gambar IV.2**.



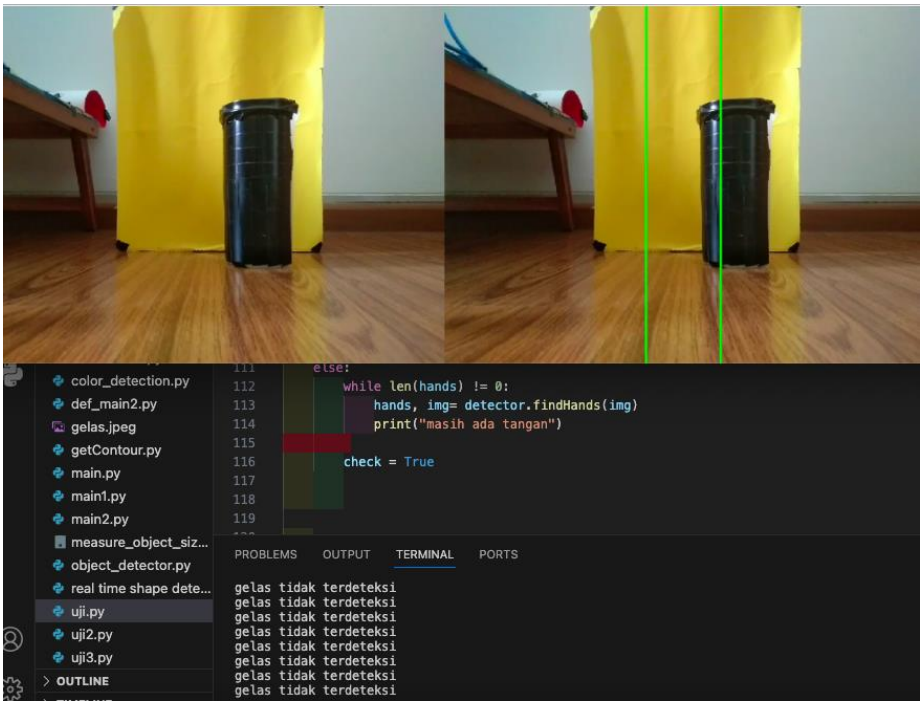
Gambar IV.2. TrackBar untuk mengubah upper threshold (threshold1), lower threshold (threshold 1), dan luas minimum area kontur yang bisa dideteksi secara real time

IV.3. Deteksi Gelas Arah Horizontal

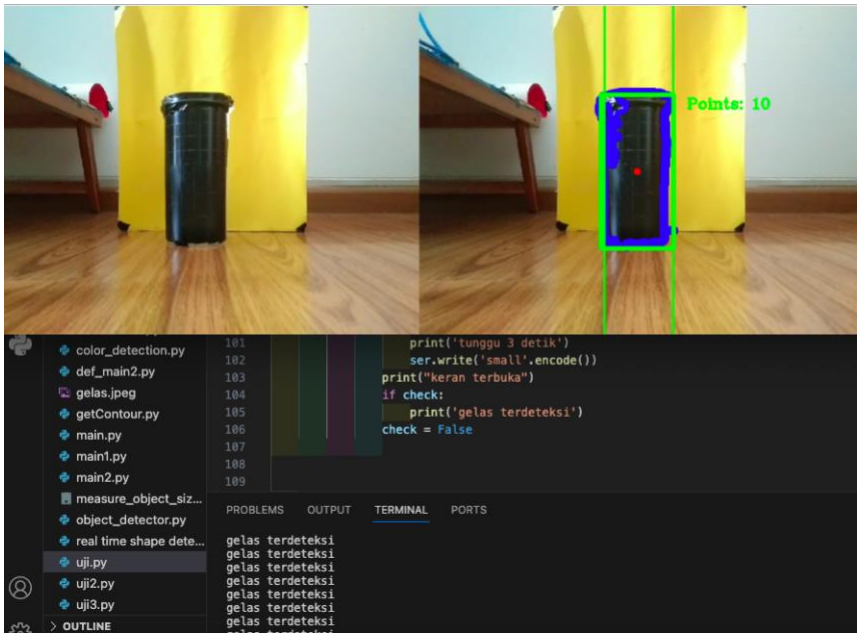
Seperti yang terlihat pada gambar **III.2**, apabila posisi center dari kontur tidak tepat di antara dua garis hijau yang sudah ditentukan, maka gelas tidak akan dianggap ada dan tidak akan dianalisis lebih lanjut untuk penentuan volumenya. Dua garis hijau

ini megammbarkan posisi maksimum dan minimum yang bisa diperoleh oleh gelas agar tidak keluar dari jangkuan aliran air dari keran.

(a)



(b)

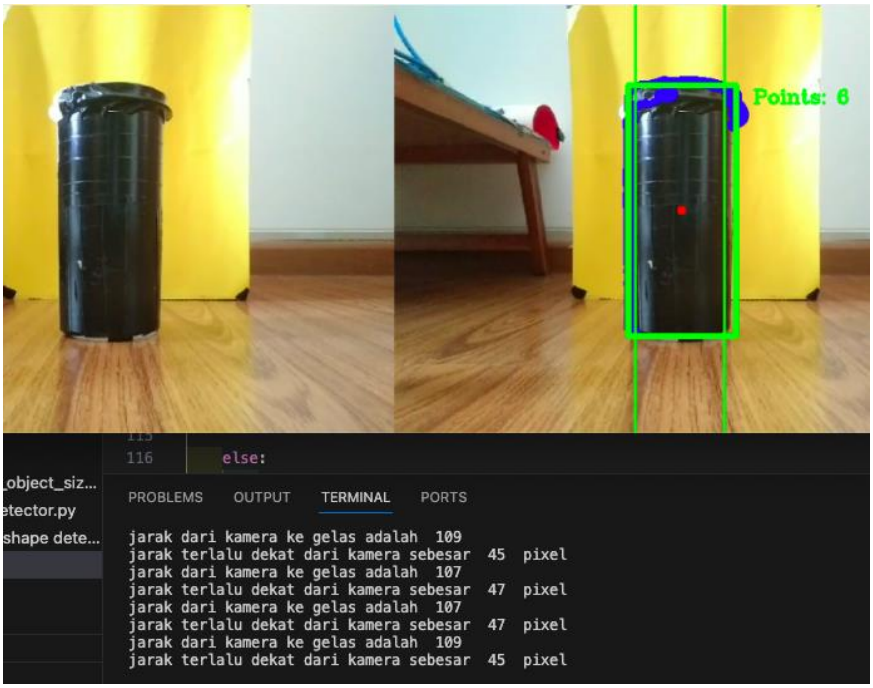


Gambar IV.3. Hasil gelas yang (a) tidak terdeteksi dan (b) terdeteksi

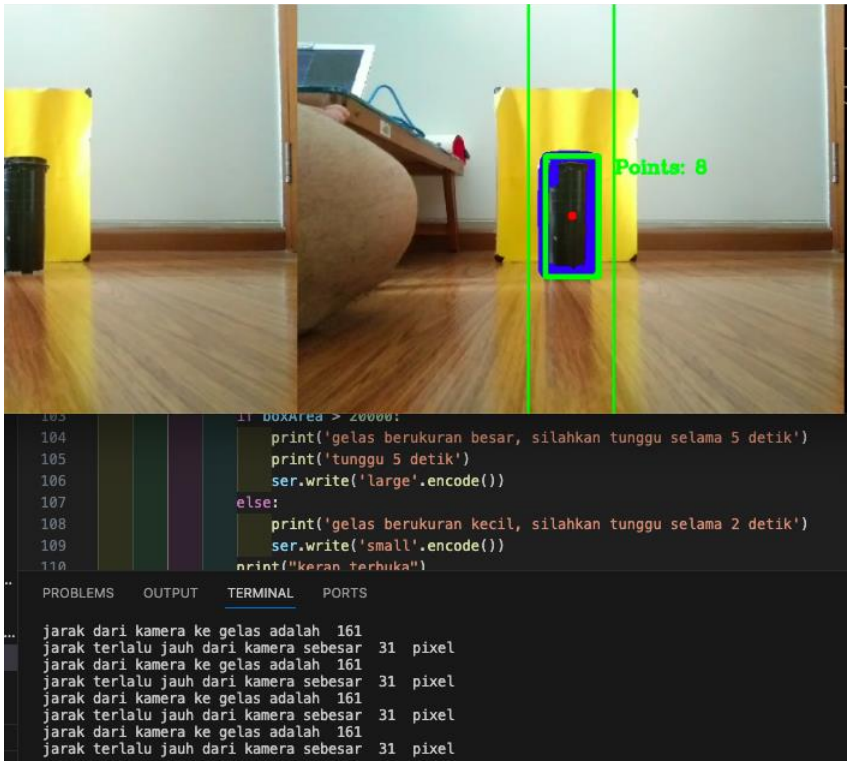
IV.4. Deteksi Jarak Antara Gelas Dengan Kamera atau Arah Vertikal

Untuk menentukan jarak maksimum dan minimum, maka diperlukan kalibrasi jarak menggunakan persamaan yang peroleh dari **gambar IV.1**. Jarak maksimum dan minimum bisa ditentukan dengan bebas asalkan jarak maksimum dan minimum yang digunakan tidak membuat gelas yang digunakan tidak terpotong oleh layar. Seperti halnya pada **IV.2**, jarak maksimum dan minimum ini disesuaikan dengan posisi keran sehingga tidak keluar dari diameter bagian atas gelas.

(a)



(b)

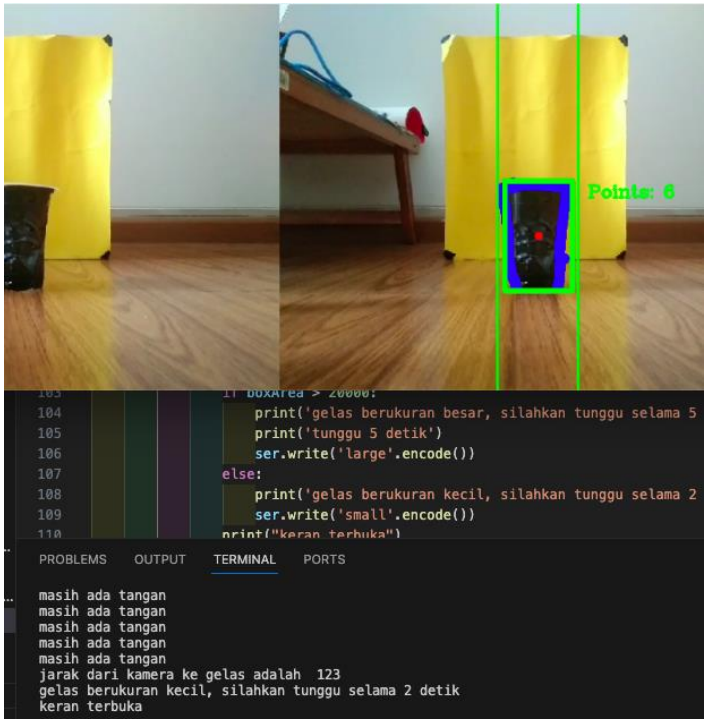


Gambar IV.4. Hasil deteksi jarak apabila (a) terlalu dekat, dan (b) terlalu jauh

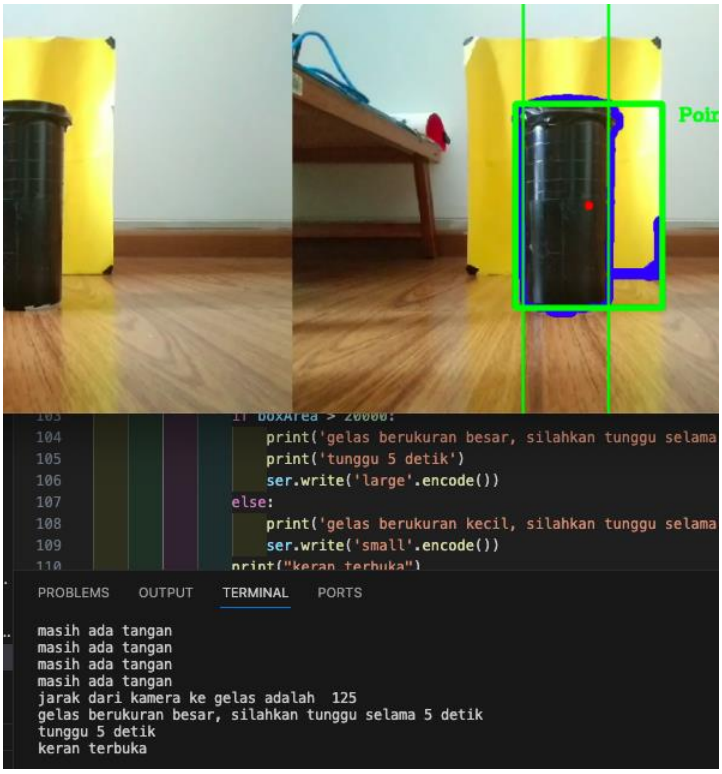
IV.5. Penentuan Volume Gelas

Setelah Jarak Vertiakal dan Horizontal terpenuhi, tahap selanjutnya adalah penentuan volume gelas apakah gelas tersebut termasuk ke gelas kecil atau gelas besar. Hal ini diperlukan karena data yang akan dikirimkan ke Arduino akan menentukan seberapa lama keran akan terbuka melalui servo motor. Apabila kita menentukan volume yang sama untuk gelas kecil dan gelas besar, maka salah satu kemungkinan yang terjadi adalah gelas kecil tidak memuat volume atau gelas besar kekurangan volume yang seharusnya. Pada projek ini, kalibrasi untuk menentukan volume gelas tidak dilakukan sehingga output yang dikeluarkan hanya sebatas permisalan saja. Seperti yang terlihat pada **gambar IV.4**, apabila volume terdeteksi, maka output yang dihasilkan merupakan ukuran gelas dan waktu yang diperlukan keran untuk terbuka sehingga bisa memenuhi volume gelas yang kita deteksi.

(a)



(b)



Gambar IV.5. Hasil deteksi volume dan output waktu yang dikeluarkan saat gelas bervolume (a) kecil dan (b) besar

V. Kesimpulan

Berdasarkan hasil kalibrasi, seperti yang terlihat pada **gambar IV.1**, didapatkan persamaan untuk menentukan jarak asli berdasarkan pixel kamera, yaitu persamaan yang dihasilkan untuk mendapatkan jarak pixel pada kamera adalah $y = 6.2121x - 25.661$ dengan keliniearan R^2 sebesar 0.9987.

Lalu, untuk menimalisisr kemungkinan kesalahan deteksi kontur akibat perbedaan intensitas gradien yang terlalu kecil bisa dilakukan filtering, seperti menentukan background yang homogen dan mengkalibrasi gambar canny dengan menentukan upper dan lower threshold yang tepat sampai hanya menyisakan deteksi tepi dari bentuk gelas bagian luar saja.

Setelah langkah kalibrasi dan filtering, maka hasil yang diperoleh akan konsisten dan tidak menyebabkan algoritma yang telah ditentukan menyimpang secara tiba-tiba, seperti penentuan jarak yang tidak konsisten akibat deteksi kontur gelas yang memanjang ke bawah akibat ada tepi acak yang terdeteksi menyambung dengan tepi gelas bagian bawah dan penentuan volume yang membuat output waktu yang dikeluarkan tidak sesuai dengan kelasnya.

Referensi

Li, Q., Wang, B., & Fan, S. (2009). Browse Conference Publications Computer Science and Engineer ... Help Working with Abstracts An Improved CANNY Edge Detection Algorithm. In 2009 Second International Workshop on Computer Science and Engineering proceedings : WCSE 2009 : 28–30 October 2009, Qingdao, China (pp. 497–500). Los Alamitos, CA: IEEE Computer Society

Otsu N. A threshold selection method from gray-level histograms. IEEE Trans Systems, Man and Cybernetics,9(1):62-66,1979.

T. Lindeberg (1998) "Edge detection and ridge detection with automatic scale selection", International Journal of Computer Vision, 30, 2, pages 117–154.

Umbaugh, Scott E (2010). Digital image processing and analysis : human and computer vision applications with CVIPtools (2nd ed.). Boca Raton, FL: CRC Press. ISBN 978-1-4398-0205-2.

Zhou, P., Ye, W., & Wang, Q. (2011). An Improved Canny Algorithm for Edge Detection. Journal of Computational Information Systems, 7(5), 1516-1523.

Lampiran

1. **Kode Python**

Kode Utama

```
import cv2
from def_main import *
from cvzone.HandTrackingModule import HandDetector
import paho.mqtt.client as mqtt
import time
import serial

wCap = 640
hCap=  480
cap = cv2.VideoCapture(1)
cap.set(10,160)
cap.set(3, wCap)
cap.set(4,hCap)

# inisialisasi detektor tangan
detector = HandDetector(detectionCon=0.8, maxHands=1)

# membuat komunikasi pyserial dengan python
ser = serial.Serial('/dev/cu.usbserial-1410', 115200) # serial.Serial(port, baundrate)

def empty(a):
    pass

check = True


# membuat window trackbar
cv2.namedWindow("parameters")                    # nama window trackbar
```

```
cv2.resizeWindow("parameters", 640, 240)                    # ukuran window trackbar
cv2.createTrackbar("threshold1", "parameters", 189, 255, empty)  # membuat trackbar threshold1 dengan nilai
cv2.createTrackbar("threshold2", "parameters", 207, 255, empty)  # membuat trackbar threshold1 dengan nilai
cv2.createTrackbar("area", "parameters", 1330, 5000, empty)      # membuat trackbar area dengan nilai


while True:

    succes, img = cap.read()
    imgContour = img.copy()

    hands, img= detector.findHands(img)

    # memperhalus gambar
    imgBlur = cv2.GaussianBlur(img, (7,7), 1)

    # convert BGR ke Gray sebagai input untuk imgCanny
    imgGray = cv2.cvtColor(imgBlur, cv2.COLOR_BGR2GRAY)

    # mendapatka nilai threshold1 dan threshold 2 menggunakan trackbar secara real-time
    threshold1 = cv2.getTrackbarPos('threshold1', 'parameters')
    threshold2 = cv2.getTrackbarPos('threshold2', 'parameters')

    # membuat image tepi dari grayScale image
    imgCanny = cv2.Canny(imgGray, threshold1, threshold2)

    # mengurangi noise dan overlap dari canny image
    kernel = np.ones((5,5))
    imgDil = cv2.dilate(imgCanny, kernel, iterations=1)
```

```
    contPointx, contPointy, contWidth, contHeight, lenContour =
getCountors(imgDil, imgContour)
    boxArea = contHeight * contWidth

    if len(hands) == 0:
        if contPointx and check:

            distance = hCap - (contPointy + contHeight)
            upperRange = 130
            lowerRange = 121
            # print(contWidth, contHeight, distance, boxArea)
            print("jarak dari kamera ke gelas adalah ", distance)

            if distance > upperRange:
                print("jarak terlalu jauh dari kamera sebesar ", distance - upperRange, "
pixel")
            elif distance < lowerRange:
                print("jarak terlalu dekat dari kamera sebesar ", lowerRange - distance , "
pixel")

            if upperRange > distance > lowerRange:

                if boxArea > 20000:
                    print('gelas berukuran besar, silahkan tunggu selama 5 detik')
                    print('tunggu 5 detik')
                    ser.write('large'.encode())
                else:
                    print('gelas berukuran kecil, silahkan tunggu selama 2 detik')
                    ser.write('small'.encode())
                print("keran terbuka")

                check = False
```

```
else:
    while len(hands) != 0:
        hands, img= detector.findHands(img)
        print("masih ada tangan")

    check = True




cv2.line(imgContour, (wCap//2+50, 0), (wCap//2+50, hCap), (0, 255, 0), 2)
cv2.line(imgContour, (wCap//2-50, 0), (wCap//2-50, hCap), (0, 255, 0), 2)
# membuat image menyatu secara horizontal

imgStack = stackImages(0.8, ([img, imgContour, imgCanny]))



cv2.imshow('original', imgStack)
if cv2.waitKey(1) & 0xFF == ord("q"):
        break
```

Kode Untuk Fungsi

```
import cv2
import numpy as np

wCap = 640
hCap = 480

# fugnsi untuk menyatukan gambar secara horizontal
def stackImages(scale,imgArray):
    rows = len(imgArray)
    cols = len(imgArray[0])
    rowsAvailable = isinstance(imgArray[0], list)
    width = imgArray[0][0].shape[1]
    height = imgArray[0][0].shape[0]
    if rowsAvailable:
        for x in range ( 0, rows):
            for y in range(0, cols):
                if imgArray[x][y].shape[:2] == imgArray[0][0].shape [:2]:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale,
scale)
                else:
                    imgArray[x][y] = cv2.resize(imgArray[x][y],
(imgArray[0][0].shape[1], imgArray[0][0].shape[0]), None, scale, scale)
                if len(imgArray[x][y].shape) == 2: imgArray[x][y]= cv2.cvtColor(
imgArray[x][y], cv2.COLOR_GRAY2BGR)
        imageBlank = np.zeros((height, width, 3), np.uint8)
        hor = [imageBlank]*rows
        hor_con = [imageBlank]*rows
        for x in range(0, rows):
            hor[x] = np.hstack(imgArray[x])
        ver = np.vstack(hor)
    else:
        for x in range(0, rows):
            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:
```

```
            imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)
        else:
            imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1],
imgArray[0].shape[0]), None,scale, scale)
        if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x],
cv2.COLOR_GRAY2BGR)
    hor= np.hstack(imgArray)
    ver = hor
  return ver


def getCountors(img,imgContour):
  contours, _ = cv2.findContours(img, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)

  if len(contours) != 0:
    for cnt in contours:
      area = cv2.contourArea(cnt)
      areaMin = cv2.getTrackbarPos('area', 'parameters')

      if area > areaMin:
        peri = cv2.arcLength(cnt, True)
        approx = cv2.approxPolyDP(cnt, 0.02*peri, True)
        x, y, w, h = cv2.boundingRect(approx)
        if wCap//2 + 50 > x + w//2 > wCap//2 - 50 and w < 200:
          cv2.drawContours(imgContour, cnt, -1, (255, 0, 0), 7)
          cv2.rectangle(imgContour, (x, y), (x + w, y + h), (0, 255, 0), 5)
          cv2.circle(imgContour, (x + w//2, y + h//2), 5, (0, 0, 255), -1)
          cv2.putText(imgContour, 'Points: ' + str(len(approx)), (x + w + 20, y +
20), cv2.FONT_HERSHEY_COMPLEX, .7,
                      (0, 255, 0), 2)
          # cv2.putText(imgContour, 'Area: ' + str(int(area)), (x + w + 20, y +
45), cv2.FONT_HERSHEY_COMPLEX, .7,
          #            (0, 255, 0), 2)
```

return x, y, w, h, len(contours)
return 0, 0, 0, 0, 0

2. Kode Arduinio

```
#include <Servo.h>

int servo; // Buffer untuk menyimpan string yang
diterima

String cupSize;

Servo tapWater;

unsigned long previousMillis = 0;
unsigned long interval = 10; // Interval waktu yang
diinginkan antara setiap pembacaan data (dalam
milidetik)


int delay_ = 30;

void setup() {

  tapWater.attach(3);
  tapWater.write(90);

  Serial.begin(115200); // Inisialisasi komunikasi
Serial
}

void loop() {

    while (Serial.available() == 0) {
    }
```

```
    cupSize = Serial.readStringUntil(' '); //
pengambilan data

    if (cupSize == "small") {
      tapWater.write(0);
      Serial.print("tunggu 3 detik");
      delay(3000);
      for (int i = 0; i < 90; i++) {
        tapWater.write(i);
        delay(10);
      }


    }

    else if (cupSize == "large") {
      tapWater.write(0);
      Serial.print("tunggu 5 detik");
      delay(5000);
      for (int i = 0; i < 90; i++) {
        tapWater.write(i);
        delay(10);
      }
    }

    // Tampilkan hasil
    Serial.print("tunggu 5 detik");

}
```