BAB IV

HASIL DAN PEMBAHASAN

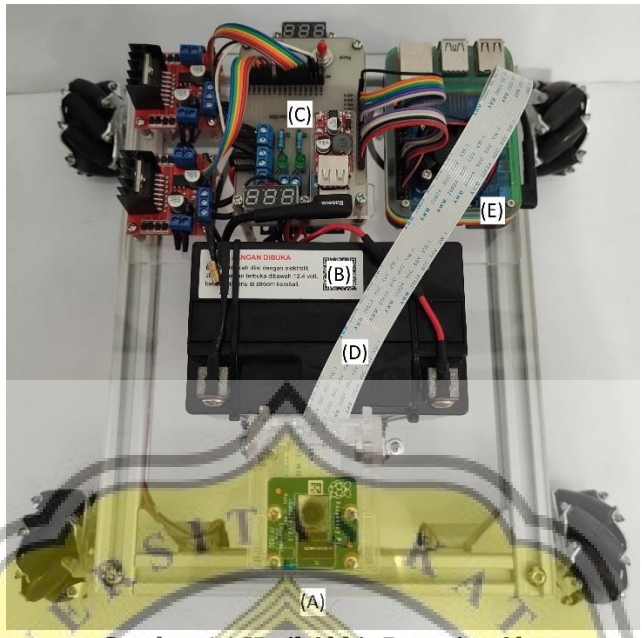
4.1. Pendahuluan

Bab ini membahas tentang hasil akhir prototipe alat, program, dan hasil dari pengujian alat dalam proyek *pattern recognition* untuk deteksi posisi pada AGV berbasis Raspberry Pi 4 model B. Pengamatan posisi dari AGV dilakukan dengan memasang sensor yaitu Raspberry Pi *Camera Module* versi 2 yang dihubungkan langsung dengan mikroprosesor Raspberry Pi 4 Model B menggunakan bahasa pemrograman *Python*.

Dengan menampilkan hasil tangkapan kamera pada layar monitor atau PC dengan menggunakan aplikasi VNC *Viewer*, maka posisi dari AGV dapat dideteksi. Hal itu dilakukan untuk membuktikan hasil percobaan telah sesuai dengan dasar teori pada Bab II dan perancangan alat pada Bab III.

4.2. Hasil Akhir Prototipe Alat

Prototipe alat dapat dilihat seperti pada Gambar 4.1 dibawah ini. Pada Gambar 4.1(A) merupakan Raspberry Pi *Camera*, Gambar 4.1(B) merupakan *accu*, Gambar 4.1(C) merupakan rangkaian pembagi tegangan, Gambar 4.1(D) merupakan kabel konektor 15 pin I/O, Gambar 4.1(E) merupakan Raspberry Pi.



Gambar 4.1 Hasil Akhir Prototipe Alat

Penelitian ini menggunakan pola huruf berwarna hitam yang telah dicetak sebelumnya pada kertas A4 berwarna putih dan objek benda (patung) sebagai *segment* seperti pada Gambar 4.2 yang nantinya akan dideteksi kamera. *Segment* tersebut digunakan sebagai tolok ukur dalam kepresisian dari deteksi posisi AGV yang dilakukan.

Gambar 4.2 Pola Huruf dan Objek yang Digunakan Dalam Penelitian

4.3. Program

Pada sub-bab ini akan dipecah menjadi 3 pembahasan yaitu deskripsi, *flowchart* program, dan penjelasan program. Pada bagian deskripsi akan diterangkan tentang tahap-tahap yang dilakukan dalam proyek tugas akhir ini. Kemudian, pada bagian *flowchart* program akan dipaparkan tentang penyusunan algoritma program yang telah disusun. Dan pada bagian penjelasan program akan dijabarkan lebih detail tentang fungsi-fungsi program yang digunakan.

4.3.1. Deskripsi

Proyek tugas akhir ini merupakan sistem pengenalan pola (*pattern recognition*) maupun objek. Dengan menggunakan Raspberry Pi *Camera* dan Raspberry Pi untuk mendeteksi posisi dari AGV.

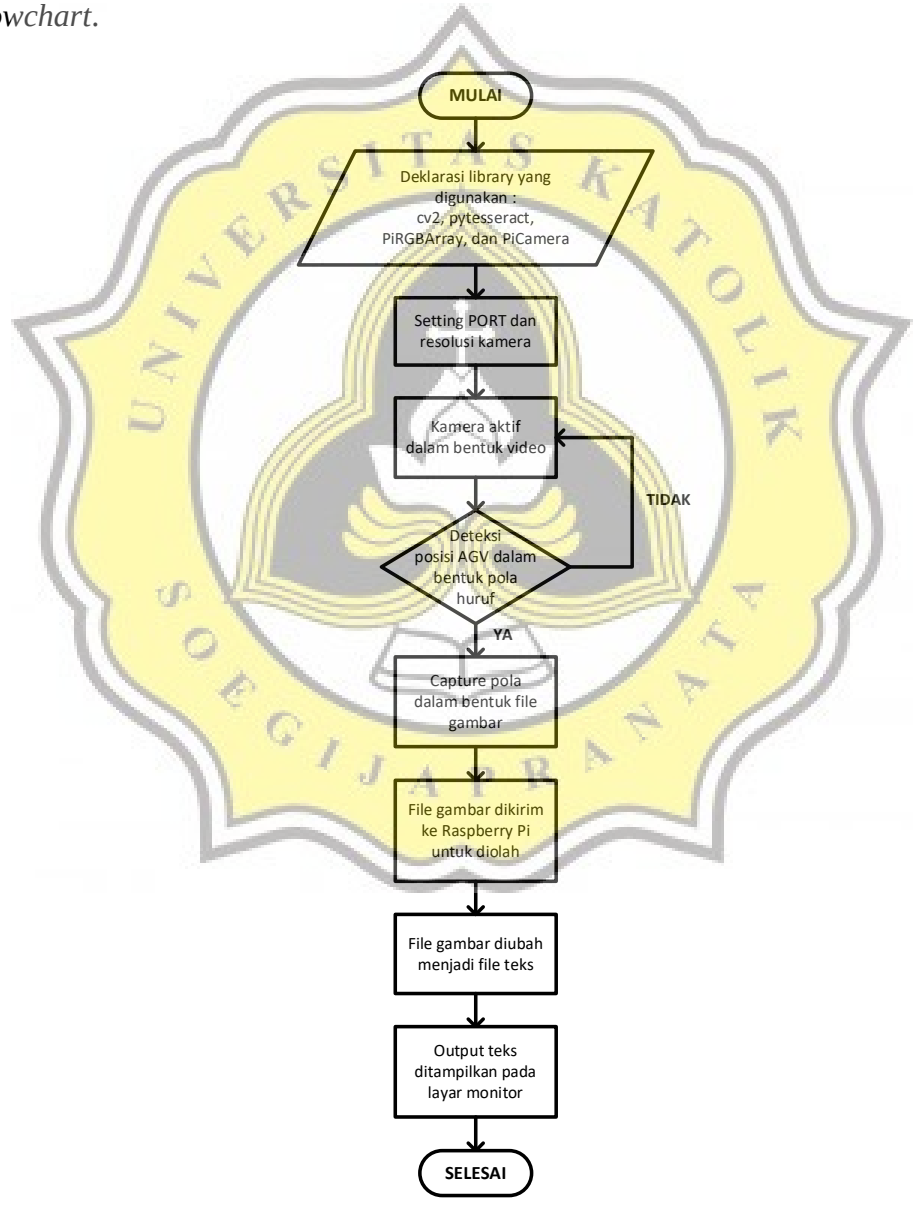
Raspberry Pi *Camera* dipasang pada 2-*lane* MIPI CSI *camera port* yang sudah tersedia pada Raspberry Pi menggunakan kabel konektor FFC 15 pin I/O. Kemudian Raspberry Pi mendapat *supply* daya dari *accu* yang diturunkan tegangannya ke 10 Volt dan arus ke 3 Ampere melalui rangkaian *step-down chooper*.

Tahap awal yaitu pola huruf ditangkap oleh kamera, lalu diolah oleh sistem Raspberry Pi. Setelah itu, Raspberry Pi akan memberi keluaran berupa titik koordinat (x,y) yang selanjutnya akan dikirimkan untuk menggerakkan motor sesuai perintah yang telah ditentukan. Dalam perjalanan AGV menuju titik koordinat (x`,y`) yang telah dirancang sebelumnya, kamera akan menangkap 3 *segment* atau bagian yang dijadikan sebagai tolok ukur dari pergerakan AGV. Bagian yang dijadikan sebagai tolok ukur yaitu patung. Dengan adanya tolok ukur

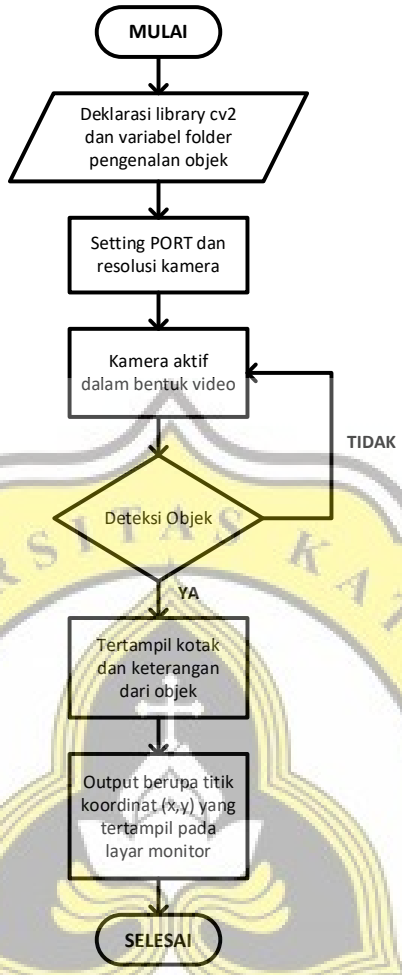
tersebut, maka dapat dihasilkan perpindahan yang lebih presisi menuju titik koordinat (x`,y`).

4.3.2. *Flowchart* Program

Sebelum menuliskan kode program, pada Gambar 4.3 dan Gambar 4.4 di bawah ini akan dipaparkan tentang penyusunan algoritma program dalam bentuk *flowchart*.



Gambar 4.3 *Flowchart* Program OCR *Capture*



Gambar 4.4 *Flowchart* Program *Object Detection*

4.3.3. Penjelasan Program

Dalam proyek penelitian ini, program akan dibagi menjadi 2 bagian. Pada bagian pertama akan diterangkan tentang program OCR *capture*, program ini terkhusus digunakan saat mendeteksi dengan cara menangkap pola (huruf) sebagai penentu posisi dari AGV. Dan untuk bagian kedua akan diterangkan tentang program *object detection*, program ini terkhusus digunakan saat mendeteksi objek (patung) sebagai tolok ukur dari pergerakan AGV.

a. Program OCR *Capture*

```
import cv2
import pytesseract
from picamera.array import PiRGBArray
from picamera import PiCamera
```

Untuk menangkap pola (huruf), mula – mula perlu diaktifkannya beberapa *library*. Pada baris pertama digunakan untuk mengolah gambar dan video hingga mampu mengekstrak informasi didalamnya dengan memanggil "**cv2**". Lalu pada baris kedua dipanggil "**pytesseract**" yang berfungsi untuk mengekstrak atau mengubah gambar menjadi huruf. Kemudian untuk fungsi "**from picamera.array import PiRGBArray**" pada baris ketiga, digunakan untuk mengumpulkan beberapa tangkapan gambar dengan ukuran dan tipe data yang sama secara urut. Dan untuk fungsi "**from picamera import PiCamera**" pada baris keempat digunakan untuk mengaktifkan kamera.

```
camera = PiCamera()
camera.resolution = (640, 640)
camera.framerate = 30

rawCapture = PiRGBArray(camera, size=(640, 640))
```

Tahap berikutnya, *setting* variabel "**PiCamera**" sebagai kamera. Lalu *setting* resolusi untuk ukuran bingkai kamera pada lebar 640 dan tinggi 640. Kemudian *setting* pada 30 FPS (*Frame Per Second*). Selanjutnya, gabungkan variabel – variabel di atas dalam satu variabel "**PiRGBArray(camera, size=(640, 640))**" sebagai tangkapan gambar mentah (**rawCapture**) sebelum diolah lebih lanjut.

```
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    image = frame.array
    cv2.imshow("Frame", image)

    key = cv2.waitKey(1) & 0xFF
    rawCapture.truncate(0)

    if key == ord("s"):
        conf = r'--oem 3 --psm 6 outputbase digits'
        text = pytesseract.image_to_string(image,config=conf)
        print(text)
        cv2.imshow("Frame", image)
        cv2.waitKey(0)
```

Kemudian, masuk dalam pengulangan program. Pada bagian ini program akan mulai membaca bingkai kamera yang telah di*setting*. Dalam "**`camera.capture_continuous`**" terdapat tiga fungsi. Fungsi "**`rawCapture`**" digunakan untuk *setting* bingkai pada kamera. Fungsi "**`format="bgr"`**" digunakan untuk *setting* RGB. Dan fungsi "**`use_video_port=True`**" digunakan untuk menyalakan kamera dalam bentuk video. Kemudian *setting* variabel "**`frame.array`**" sebagai gambar. Yang selanjutnya akan ditampilkan dalam layar monitor.

Setting "**`cv2.waitKey(1) & 0xFF`**" sebagai tombol pada *keyboard* untuk melakukan perintah. Jika tombol ditekan selama waktu yang ditentukan, maka program akan berlanjut.

Selanjutnya untuk fungsi "**`key == ord("s")`**", menggunakan tombol "**`s`**" untuk melakukan perintah. Jika tombol "**`s`**" ditekan, kamera akan menangkap gambar. Pada fungsi "**`text = pytesseract.image_to_string(image,config = conf)`**", gambar yang ditangkap akan diubah dan ditampilkan dalam bentuk teks sebagai keterangan. Melalui fungsi "**`print(text)`**", keterangan tersebut akan ditampilkan dalam terminal pada layar monitor. Untuk fungsi

"**`cv2.imshow("Frame", image)`**", akan ditampilkan hasi dari tangkapan gambar oleh kamera. Dan fungsi "**`cv2.waitKey(0)`**" berfungsi untuk menunggu perintah selanjutnya.

```
cv2.destroyAllWindows()
```

Pada baris terakhir dari program di atas, digunakan atau berfungsi untuk menghentikan *Open* CV. Dengan menekan tombol "**Esc**" pada *keyboard*, *Open* CV dan kamera akan otomatis berhenti(*off*).

b. Program *Object Detection*

```
import cv2
```

Untuk menangkap objek (patung), mula – mula perlu diaktifkannya *library* dari *Open* CV. Pada baris pertama, "**`cv2`**" dipanggil untuk mengolah gambar dan video hingga mampu mengekstrak informasi didalamnya.

```
path = 'haarcascades/patung.xml'
cameraNo = 0
objectName = 'Patung'
frameWidth = 640
frameHeight = 480
cameraBrightness = 50
color = (0,255,50)
```

Kemudian deskripsikan variabel yang akan digunakan. Pada variabel "**`path`**" digunakan untuk memanggil folder "**`folderobject/patung.xml`**". Berisikan beberapa sampel gambar yang akan dipakai sebagai pengenalan objek (*object detection*). Atur PORT 0 pada variabel "**`cameraNo`**" yang digunakan sebagai *input*

kamera. Variabel "**objectName**" digunakan untuk memberi keterangan nama objek dalam bentuk teks yang akan ditampilkan pada bagian bawah dari kotak deteksi. Selanjutnya, *setting* untuk resolusi kamera pada lebar 640, tinggi 480, dan kecerahan 50. Pada program ini, kotak deteksi menggunakan warna merah dengan *setting* RGB "**(0,255,50)**".

```
cap = cv2.VideoCapture(cameraNo)
cap.set(3, frameWidth)
cap.set(4, frameHeight)

cascade = cv2.CascadeClassifier(path)
```

Lalu *setting* fungsi "**cv2.VideoCapture(cameraNo)**" sebagai cap. Dilanjut dengan mengatur cap untuk lebar dan tinggi bingkai kamera. *Setting* juga fungsi "**cv2.CascadeClassifier(path)**" sebagai cascade. Pada variabel "**path**" berfungsi untuk mendeteksi objek yang diinginkan sebelumnya.

```
while True:
    cap.set(10, cameraBrightness)
    success, img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    scaleVal = 1 + (0.4)
    neig = 8
    objects = cascade.detectMultiScale(gray,scaleVal,neig)

    for (x,y,w,h) in objects:
        area = w*h
        minArea = 1000
        if area > minArea:
            cv2.rectangle(img,(x,y),(x+w,y+h),color,2)
            cv2.putText(img,objectName,(x,y-5),cv2.FONT_HERSHEY_COMPLEX_SMALL,1,color,1)
            roi_color = img[y:y+h, x:x+w]
            print("X:",x," ","Y:",y)

    cv2.imshow ("Result", img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Selanjutnya atur cap untuk kecerahan pada bingkai kamera. *Setting* juga agar kamera dapat berhasil untuk menampilkan gambar.

Pertama - tama, ubah gambar menjadi skala abu seperti pada variabel "`cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)`". Lalu *setting* variabel "`scaleVal`" menjadi 1 + (0.4) dan variabel "`neig`" menjadi 8. Pada variabel "`cascade.detectMultiScale (gray,scaleVal,neig)`", digunakan untuk mendeteksi objek patung yang didapat dari tangkapan gambar oleh kamera.

Kemudian, masuk dalam pengulangan program. Pada bagian ini, akan menampilkan objek dan kotak deteksi. Terdapat variabel x, y, w, dan h. Digunakan sebagai pengukuran x, y, lebar, dan tinggi dari objek yang dideteksi. Lalu *setting* area dan minimal area. Kotak akan mendeteksi objek dengan syarat minimal area deteksi pada kamera yaitu 1000. Fungsi "`cv2.rectangle(img,(x,y),(x+w,y+h),color,2)`" digunakan untuk menampilkan kotak deteksi. Lalu, pada fungsi "`cv2.putText (img, objectName, (x,y-5), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, color, 1)`" digunakan untuk menampilkan nama objek dari tangkapan kamera. Dan fungsi `print("X:",x," ","Y:",y)` digunakan untuk menampilkan skala x dan y, sehingga diketahui posisi atau titik koordinat dari objek tersebut.

Untuk fungsi "`cv2.imshow ("Result", img)`" digunakan untuk menampilkan hasil dan tangkapan gambar. *Setting* "`if cv2.waitKey(1) & 0xFF == ord('q')`" sebagai tombol "`q`" pada *keyboard* untuk melakukan perintah berhenti atau *off*.

```
cap.release()
cv2.destroyAllWindows()
```

Pada fungsi "**`cap.release()`**" dan "**`cv2.destroyAllWindows()`**" digunakan untuk menghentikan *Open* CV. Dengan menekan tombol "**Esc**" pada *keyboard*, *Open* CV dan kamera akan otomatis berhenti.

4.4. Hasil Pengujian Alat

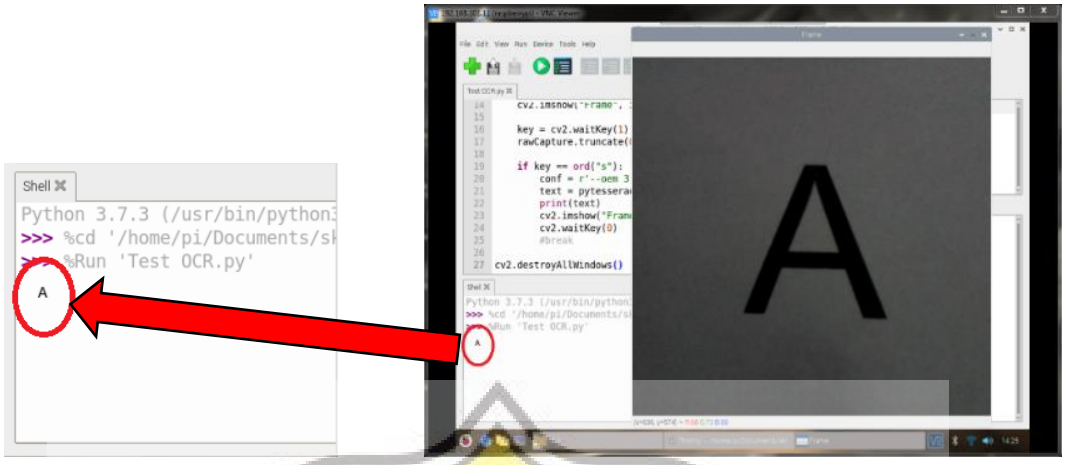
Setelah program-program tersebut berhasil di-*upload* dan tidak ada *error*, maka dilakukan pengujian. Dalam pengujian tersebut didapatkan hasil dari beberapa parameter yang telah ditentukan sebelumnya.

4.4.1. Pengujian Program OCR *Capture*

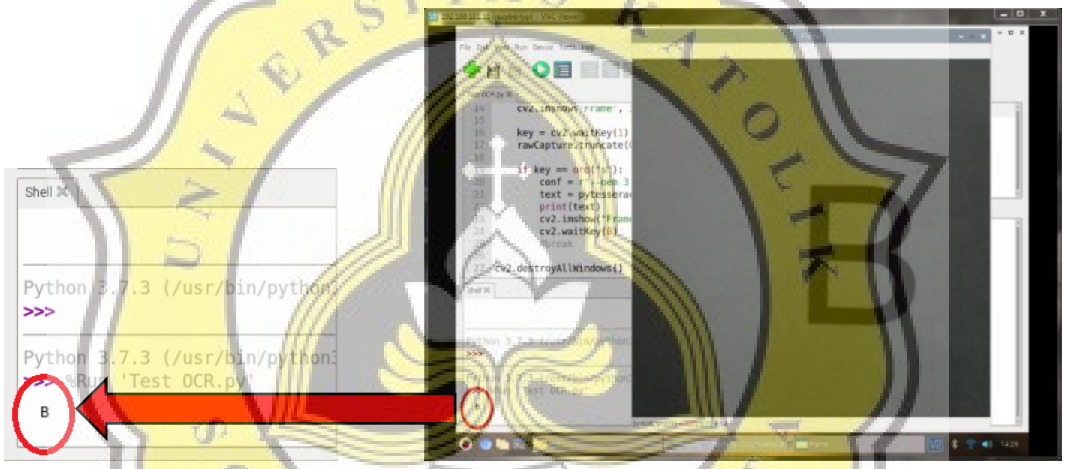
Pada program OCR *capture* dilakukan pengujian saat menangkap gambar pola, dimana yang dipakai adalah huruf "A", "B", "C", "D", dan "P". Pola "A", "B", "C", dan "D" digunakan sebagai titik-titik tujuan dalam penyimpanan barang oleh AGV. Sedangkan pola "P" digunakan sebagai titik posisi parkir pada robot AGV jika tidak digunakan atau saat pertama kali akan dijalankan.

Pengujian program OCR *capture* ini dapat dilakukan dengan cara menangkap gambar pola dari kamera. Kemudian *file* gambar tersebut diolah dan ditampilkan menjadi *file* teks pada layar monitor. Keluaran berupa *file* teks itulah yang digunakan sebagai titik koordinat (x,y) dari AGV.

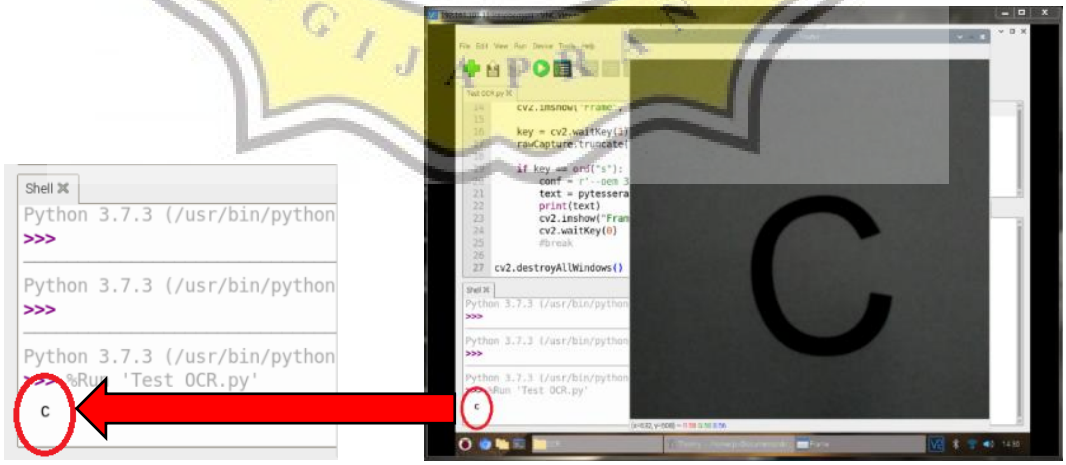
Hasil uji pemrosesan dalam mengubah *file* gambar menjadi *file* teks yang ditampilkan pada layar monitor, dapat dilihat pada Gambar 4.5, Gambar 4.6, Gambar 4.7, Gambar 4.8, dan Gambar 4.9 di bawah ini.



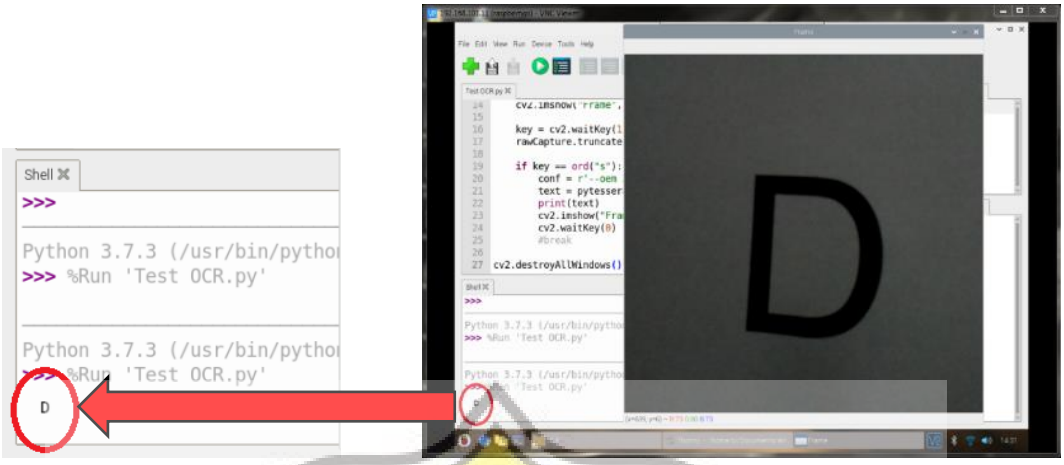
Gambar 4.5 Hasil Uji Olah Gambar Menjadi Teks pada *Capture* Pola "A"



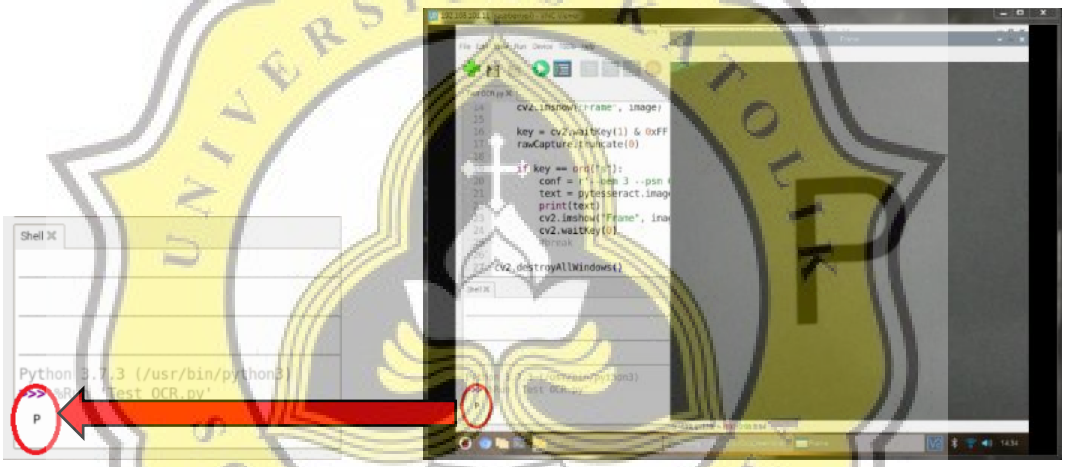
Gambar 4.6 Hasil Uji Olah Gambar Menjadi Teks pada *Capture* Pola "B"



Gambar 4.7 Hasil Uji Olah Gambar Menjadi Teks pada *Capture* Pola "C"



Gambar 4.8 Hasil Uji Olah Gambar Menjadi Teks pada *Capture* Pola "D"



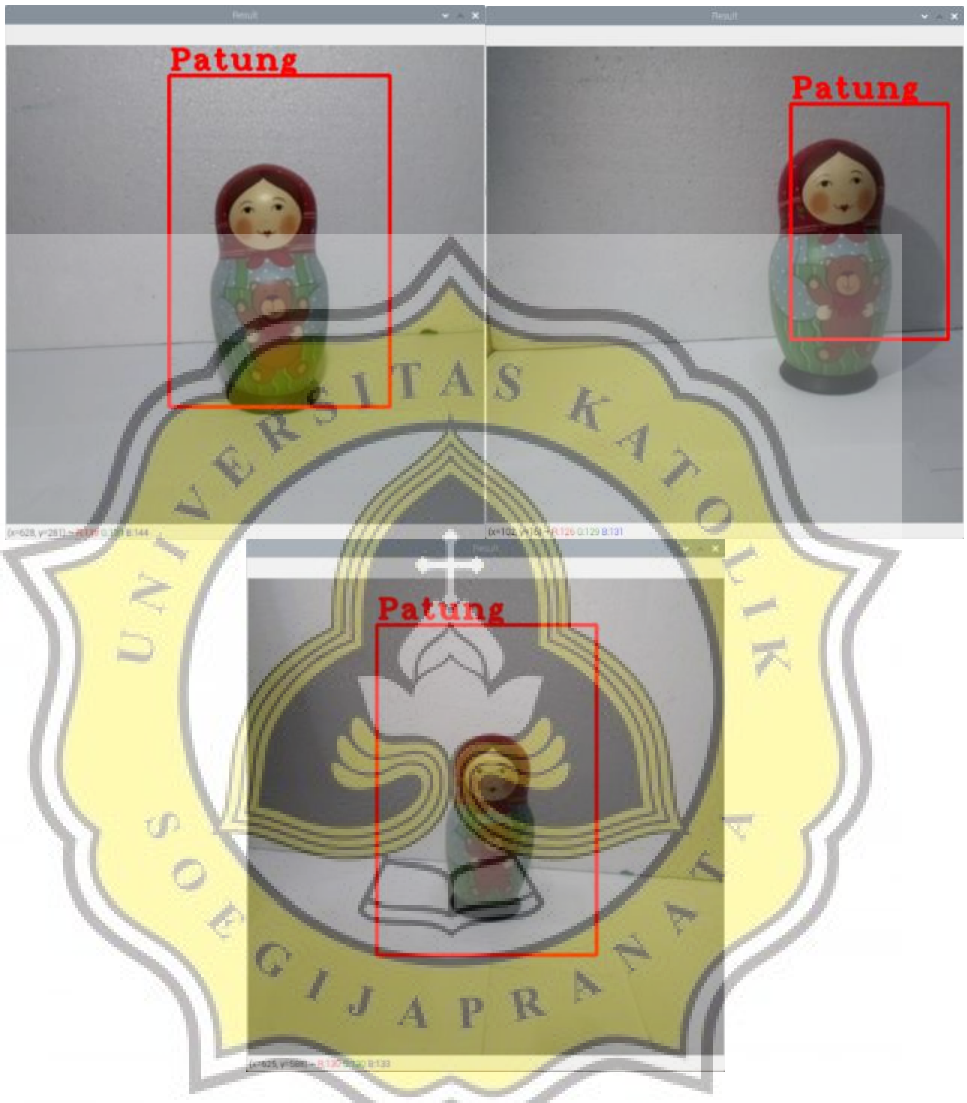
Gambar 4.9 Hasil Uji Olah Gambar Menjadi Teks pada *Capture* Pola "P"

4.4.2. Pengujian Program *Object Detection*

Lalu untuk program *object detection*, dilakukan pengujian ketepatan program dalam mendeteksi objek. Objek yang digunakan pada projek ini adalah patung. Sebelumnya, data masukan berupa beberapa *file* gambar dari patung disimpan dalam satu folder dan diberikan nama atau keterangan "Patung".

Hasil uji deteksi objek yaitu dengan menampilkan kotak yang di atasnya bertuliskan "Patung" dari objek yang dideteksi. Jika terdeteksi, akan tertampil

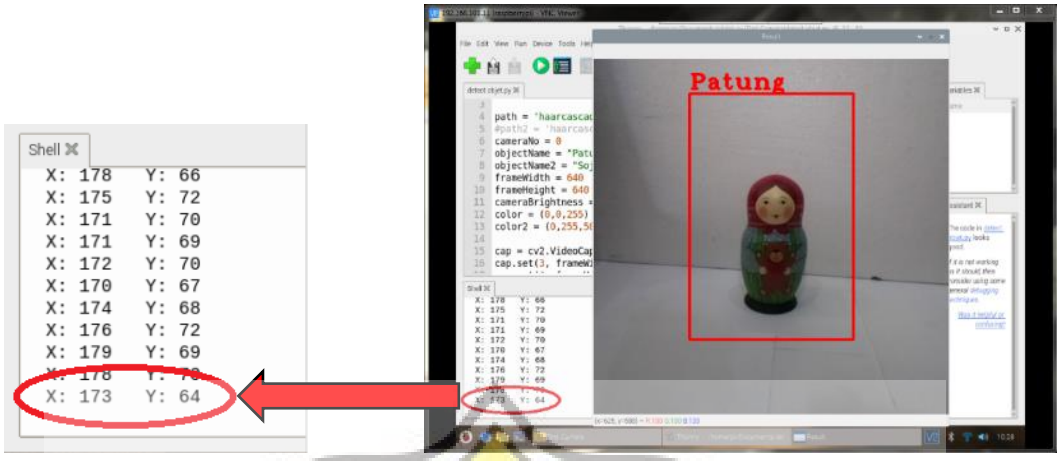
pada layar monitor seperti pada Gambar 4.10. Jika tidak, maka akan tertampil seperti Gambar 4.11.



Gambar 4.10 Hasil Pengujian *Object Detection* pada Objek Patung Terdeteksi

Gambar 4.11 Hasil Pengujian *Object Detection* pada Objek Patung Tidak Terdeteksi (a) Intensitas Cahaya Terlalu Rendah, (b) Intensitas Cahaya Terlalu Tinggi (c) Objek Terlalu Dekat

Dan gambar di bawah ini merupakan hasil titik koordinat (x,y) dari objek tersebut pada bingkai "*Shell*". Titik koordinat objek tersebut ditunjukkan pada titik x dan y paling bawah atau yang muncul paling akhir. Seperti yang terlingkar pada Gambar 4.12.



Gambar 4.12 Hasil Pengujian Titik Koordinat Berupa X dan Y