BAB IV

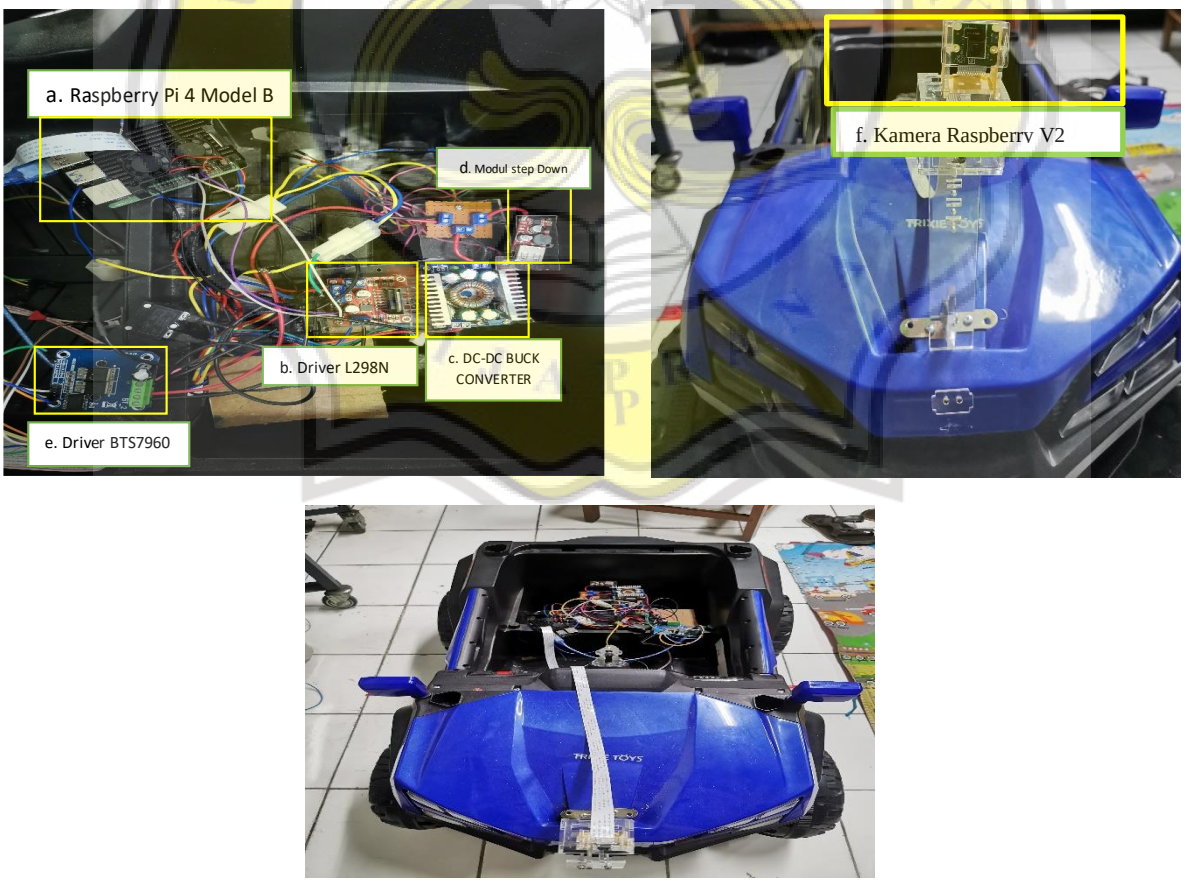
HASIL DAN PEMBAHASAN

4.1. Pendahuluan

Pada bab ini akan dijelaskan hasil hasil *prototype Autonomous Car*, program, dan pengujian alat yang dilakukan untuk pendeteksian objek *Street Mark* yang digunakan pada *Autonomous Car* . Uji hasil dan pembahasan ini akan menyajikan hasil prototype Alat, Program yang digunakan, dan perbandingan sistem Default OCR dengan OCR yang sudah dimodifikasi atau disempurnakan.

4.2. Prototype Alat

Dalam penelitian ini, menghasilkan bentuk *hardware* yang digunakan untukTugas Akhir. Hasil *hardware* ini sesuai dengan perancangan alat pada Bab III. Gambar 4.1 menunjukkan *hardware* dari *Autonomous Car*.



Gambar 4.1 Hardware *Autonomous Car*

4.3. Program

Berikut ini merupakan pembahasan program untuk sistem pergerakan *Autonomous Car* saat bergerak mengikuti lintasan.

main.py

```
1  import cv2
2  import numpy as np
3  import serial
4  import RPi.GPIO as GPIO
5
```

Langkah pertama, import *library* yang dibutuhkan. Fungsi ***import*** Adalah untuk memasukkan library diataranya ada cv2 , serial, import sendiri adalah fungsi untuk memasukkan program tersendiri yang sebelumnya dibuat.

```
6  if __name__ == '__main__':
7      GPIO.setwarnings(False)
8      kernel = np.ones((5,5),np.uint8)
9      cap = cv2.VideoCapture(0)
10     panjang=320
11     lebar=240
12     cap.set(3,panjang)
13     cap.set(4,lebar)
14     in1 = 24
15     in2 = 23
16     en = 25
17     GPIO.setmode(GPIO.BCM)
18     GPIO.setup(in1,GPIO.OUT)
19     GPIO.setup(in2,GPIO.OUT)
20     GPIO.setup(en,GPIO.OUT)
21     GPIO.output(in1,GPIO.LOW)
22     GPIO.output(in2,GPIO.LOW)
23     p=GPIO.PWM(en,1000)
24     kondisinya=0
25     p.start(100)
26     final=0
27     p.ChangeDutyCycle(100)
28     ser = serial.Serial('/dev/ttyUSB0', 115200, timeout=0.05)
```

Dilihat dari gambar 4.2 diatas adalah bagian program untuk pembacaan serial real time video yang diimplementasikan dalam *Autonomous Car* dimana program diatas ini berisi penambahan prosesing sebagai pendeteksi pola jalur lintasan *Street Mark* tampilan hasil deteksi ini akan ditampilkan dalam layar monitor. Sehingga jalur terdeteksi dan

pembacaan akan langsung ditampilkan pada monitor .

```
30
31      def nothing(x):
32          pass
33      cv2.namedWindow('HueComp')
34      cv2.namedWindow('SatComp')
35      cv2.namedWindow('ValComp')
36      cv2.namedWindow('closing')
37      cv2.namedWindow('tracking')
38
39      cv2.createTrackbar('hmin', 'HueComp',168,179,nothing)
40      cv2.createTrackbar('hmax', 'HueComp',179,179,nothing)
41
42      cv2.createTrackbar('smin', 'SatComp',40,255,nothing)
43      cv2.createTrackbar('smax', 'SatComp',255,255,nothing)
44
45      cv2.createTrackbar('vmin', 'ValComp',26,255,nothing)
46      cv2.createTrackbar('vmax', 'ValComp',129,255,nothing)
47
48      def sortSecond(val):
49          return val[1]
```

Program pada gambar 4.3 diatas ini adalah program metode *HSV* yang memiliki fungsi untuk mendeteksi lintasan *Street Mark* yang memiliki fungsi untuk mengkalibrasikan suatu objek agar *Autonomous Car* dapat bergerak secara mengikuti jalur.

main.py

```
50      while True:
51          nilai=""
52          if(ser.in_waiting>0):
53              while ser.in_waiting:
54                  nilai =str(ser.readline())
55                  nilai=nilai.replace(r"\r\n","")
56                  nilai=nilai.replace("b","")
57                  nilai=nilai.replace("'","")
58                  if(nilai!="" ):
59                    akhir=nilai
60          if(int(akhir)>200 and int(akhir)<580):
61              final=int(akhir)
62          print(final)
```

Program diatas adalah progam untuk mententukan tingkat kepresisian suatu sudut dalam menentukan titik awal *Autonomous Car* terletak.

main.py

```
        buzz = 0
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.LOW)

        ser.write(b"m\n")
        _, frame = cap.read()
        hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
        hue,sat,val = cv2.split(hsv)
        hmn = cv2.getTrackbarPos('hmin','HueComp')
        hmx = cv2.getTrackbarPos('hmax','HueComp')


        smn = cv2.getTrackbarPos('smin','SatComp')
        smx = cv2.getTrackbarPos('smax','SatComp')


        vmn = cv2.getTrackbarPos('vmin','ValComp')
        vmx = cv2.getTrackbarPos('vmax','ValComp')
        hthresh = cv2.inRange(np.array(hue),np.array(hmn),np.array(hmx))
        sthresh = cv2.inRange(np.array(sat),np.array(smn),np.array(smx))
        vthresh = cv2.inRange(np.array(val),np.array(vmn),np.array(vmx))
        tracking = cv2.bitwise_and(hthresh,cv2.bitwise_and(sthresh,vthresh))
```

```

    dilation = cv2.dilate(tracking,kernel,iterations = 1)
    #dilation1 = cv2.dilate(kebalikan,kernel,iterations = 1)
    closing = cv2.morphologyEx(dilation, cv2.MORPH_CLOSE, kernel)
    closing = cv2.GaussianBlur(closing,(5,5),0)
    #closing1 = cv2.morphologyEx(dilation1, cv2.MORPH_CLOSE, kernel)
    #closing1 = cv2.GaussianBlur(closing1,(5,5),0)
    contours, hierarchy = cv2.findContours(closing,
    cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_TC89_L1)
    result = frame.copy()
    polygonelist = []
    kanan=[]
    kiri=[]
    kananrev=[]
    kirirev=[]
    xkanan=0
    xkiri=0
    ykanan=0
    ykiri=0
```

main.py

```
    ykiri=0
    #print("baru")
    for cntr in contours:
        perimeter = cv2.arcLength(cntr,True)
        epsilon = 0.005*cv2.arcLength(cntr,True)
        approx = cv2.approxPolyDP(cntr,epsilon,True)
        #print(approx)
        polygonelist.append(approx)
    cv2.drawContours(frame, polygonelist, -1, (0, 255, 0), 2)
    for x in range(0,len(polygonelist)):
        for a in range(0,len(polygonelist[x])):
            for b in range(0,len(polygonelist[x][a])):
                if(x==0):
                    kanan.append(polygonelist[x][a][b])
                    kananrev.append(polygonelist[x][a][b])
                elif(x==1):
                    kiri.append(polygonelist[x][a][b])
                    kirirev.append(polygonelist[x][a][b])
    #for k in range(0,len(kanan)):
    #    print(kanan[k][1])
    kanan.sort(key=sortSecond)
    kiri.sort(key=sortSecond)
    kananrev.sort(key=sortSecond,reverse=True)
    kirirev.sort(key=sortSecond,reverse=True)
```

Pada Gambar 4.4 adalah progam pergerakan motor dc dalam melakukan pergerakan *Autonomous Car* untuk mententukan antara berbelok kanan atau kiri. Pergerakan ini ditentukan dengan metode *HSV* dalam penentuan *Hue,Saturation* dan *Value* untuk mencapai nilai maksimal dan minimal dalam pantulan cahaya.

main.py

```
129         for k in range (0,2):
130             if(k==0):
131                 xkanan=kanan[k][0]
132                 xkiri=kiri[k][0]
133                 ykiri=kanan[k][1]
134                 ykanan=kiri[k][1]
135             elif(k==1):
136                 if(kanan[k][0]<xkanan):
137                     xkananatas=kanan[k][0]
138                     ykananatas=kanan[k][1]
139                 else:
140                     xkananatas=xkanan
141                     ykananatas=ykanan
142                 if(kiri[k][0]>xkiri):
143                     xkiriatas=kiri[k][0]
144                     ykiriatas=kiri[k][1]
145                 else:
146                     xkiriatas=xkiri
147                     ykiriatas=ykiri
```

main.py

```
149         for k in range (0,2):
150             if(k==0):
151                 xkanan=kananrev[k][0]
152                 xkiri=kirirev[k][0]
153                 ykiri=kananrev[k][1]
154                 ykanan=kirirev[k][1]
155             elif(k==1):
156                 if(kananrev[k][0]<xkanan):
157                     xkananbawah=kananrev[k][0]
158                     ykananbawah=kananrev[k][1]
159                 else:
160                     xkananbawah=xkanan
161                     ykananbawah=ykanan
162                 if(kiri[k][0]>xkiri):
163                     xkiribawah=kirirev[k][0]
164                     ykiribawah=kirirev[k][1]
165                 else:
166                     xkiribawah=xkiri
167                     ykiribawah=ykiri
168         misskiri=int(panjang/2)-xkiriatas
169         misskanan=xkananatas-int(panjang/2)
170         print(misskiri,misskanan,abs(misskanan-misskiri))
171         posisi="Kanan"
```

main.py

```
173         if(abs(misskanan-misskiri)<=75):
174             posisi="Tengah"
175         else:
176             if(misskiri>misskanan):
177                 posisi="Kiri"
178                 kondisinya=1
179                 GPIO.output(in1,GPIO.LOW)
180                 GPIO.output(in2,GPIO.HIGH)
181                 print("kiri")
182             else:
183                 posisi="Kanan"
184                 kondisinya=2
185                 GPIO.output(in1,GPIO.HIGH)
186                 GPIO.output(in2,GPIO.LOW)
187                 print("kanan")
```

main.py

```
188         if(kondisinya==1 and posisi=="Tengah"):
189             if(final>340 and final<450):
190                 GPIO.output(in1,GPIO.LOW)
191                 GPIO.output(in2,GPIO.LOW)
192             else:
193                 GPIO.output(in1,GPIO.HIGH)
194                 GPIO.output(in2,GPIO.LOW)
195                 print("Balik arah")
```

main.py

```
196         elif(kondisinya==2 and posisi=="Tengah"):
197             if(final>340 and final<450):
198                 GPIO.output(in1,GPIO.LOW)
199                 GPIO.output(in2,GPIO.LOW)
200             else:
201                 GPIO.output(in1,GPIO.LOW)
202                 GPIO.output(in2,GPIO.HIGH)
203                 print("Balik arah")
```

Pada Gambar 4.5 adalah progam *Autonomous Car* dalam melakukan pergerakan agar *Autonomous Car* bergerak tetap dalam kondisi didalam lintasan atau berjalan ditengah dan tidak keluar lintasan pada jalur *Street Mark* pada program diatas dijelaskan ada program HIGH dan LOW program ini sendiri untuk mengatur suatu kecepatan motor dc.

main.py

```
        cv2.putText(frame, posisi, (0,20), cv2.FONT_HERSHEY_SIMPLEX,
                    0.5, (255, 0, 0), 2, cv2.LINE_AA)
        cv2.circle(frame, (int(panjang/2),lebar), radius=2, color=(0, 0, 255), thickness=10)
        cv2.line(frame, (int(panjang/2),0), ((int(panjang/2),lebar)), (255, 0, 0), 2)
        cv2.line(frame, (xkananatas,ykananatas), ((int(panjang/2),lebar)), (0, 0, 255), 1)
        cv2.line(frame, (xkiriatas,ykiriatas), ((int(panjang/2),lebar)), (0, 0, 255), 1)
        cv2.line(frame, (xkananbawah,ykananbawah), ((int(panjang/2),lebar)), (0, 0, 255), 1)
        cv2.line(frame, (xkiribawah,ykiribawah), ((int(panjang/2),lebar)), (0, 0, 255), 1)
        #cv2.line(frame, (xkanantengan,ykanantengan), ((int(panjang/2),lebar)), (0, 0, 255),
        #cv2.line(frame, (xkiritengan,ykiritengan), ((int(panjang/2),lebar)), (0, 0, 255), 1)
        #cv2.imshow('HueComp',hthresh)
        #cv2.imshow('SatComp',sthresh)
        #cv2.imshow('ValComp',vthresh)
        cv2.imshow('closing',closing)
        cv2.imshow('tracking',frame)
        k = cv2.waitKey(5) & 0xFF
        if k == 27:
            break

    cap.release()

    cv2.destroyAllWindows()
```

Pada Gambar 4.6 adalah progam yang berfungsi untuk menampilkan trackbar yang berfungsi untuk mengatur *Hue*, *Saturation* dan *Value*

4.4. Tes Kecepatan kemudi *Autonomous Car* Berdasarkan Sudut Belok

Pada Pengujian ini dilakukan untuk mengetahui fungsi dari prototipe *Autonomous Car* yang memiliki penjelasan tentang stabilitas dan kemampuan sistem mobil self-driving untuk mendeteksi objek lintasan jalan. Dengan begitu, kita bisa melihat seberapa cepat sistem *Real-Time* mendeteksi objek lintasa jalan. Dengan sistem *Real-Time*, mobil self-driving dapat menghitung jarak deteksi berdasarkan area deteksi.

Untuk mengetahui kemampuan prototipe *Autonomous Car* bergerak sesuai jalur yang telah ditentukan. Sehingga perlu dilakukan pengujian kecepatan dan sudut belok. Pengujian ini bertujuan untuk mengetahui laju *Autonomous Car* secara linier. Kecepatan linier sendiri dipengaruhi oleh putaran motor DC yang digunakan pada prototipe *Autonomous Car* . Pengujian dilakukan dengan tachometer menggunakan robot yang berjalan pada lintasan yang telah ditentukan, dan kecepatan putaran roda dihitung menggunakan tachometer. Hasil

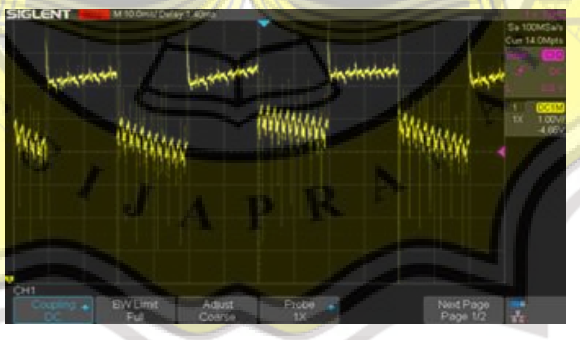
pengukuran didapatkan nilai kecepatan rata-rata dalam satuan RPM (revolutions per minute). Hasil pengukuran kecepatan putaran dapat dilihat pada Tabel I di bawah ini.

NO.	PWM duty cycle (%)	Rotation Speed (RPM)	*Autonomous Car* Speed Condition
1	30	125,5	Slow
2	40	210,8	Slow
3	50	293,5	Currently
4	60	274,8	Currently
5	70	418,9	Fast

Tabel I Hasil Pengukuran Kecepatan

Pengukuran dilakukan dengan mengatur duty cycle PWM dari 30% menjadi 70%. Hasil pengukuran disajikan pada Tabel I. Dapat dilihat pada kondisi kecepatan putar roda saat *Autonomous Car* berjalan di lintasan. Tes ini memengaruhi cara kami mengatur siklus tugas PWM-nya.

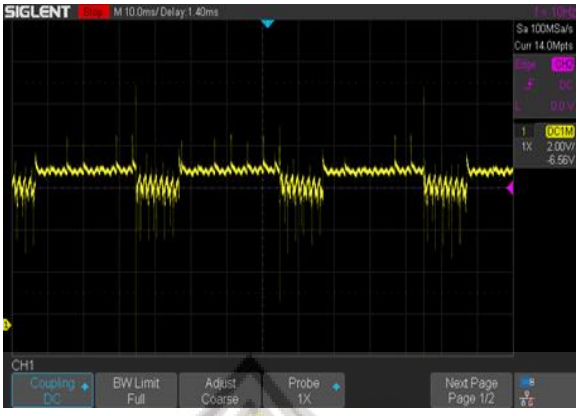
Untuk pengujian, *Autonomous Car* akan diperkuat dengan data dari sinyal keluaran. Hasil tersebut diperoleh melalui citra hasil pengukuran osiloskop. Gambar dari osiloskop nantinya akan menampilkan hasil pengukuran sinyal PWM, seperti terlihat pada Tabel



(a)

(b)

Gambar 4.7 (a)Signal duty cycle PWM 30%, (b) Akselerasi Output 125,5 RPM



(a)

(b)

Gambar 4.8(a) Siklus kerja sinyal PWM 70%, (b) Output Akselerasi 418,5 RPM

Berdasarkan hasil pengujian, dapat dianalisis bahwa *Autonomous Car* dapat berbelok dengan sudut maksimum. Nilai PWM menentukan apakah *Autonomous Car* dapat berjalan lambat atau cepat. Hasil pengukuran pertama dengan duty cycle PWM 30% ditunjukkan pada Gambar 8 (a) merupakan hasil sinyal pada osiloskop. Gambar 8 (b) menunjukkan kecepatan yang dihasilkan sebesar 125,5 RPM. Hasil pengukuran kedua dengan duty cycle PWM sebesar 70% yang ditunjukkan pada Gambar 9 (a) adalah hasil sinyal pada osiloskop. Gambar 9 (b) menunjukkan kecepatan yang dihasilkan 418,5 RPM.

4.5. Tes Warna *HSV* pada Mobil Self-Driving

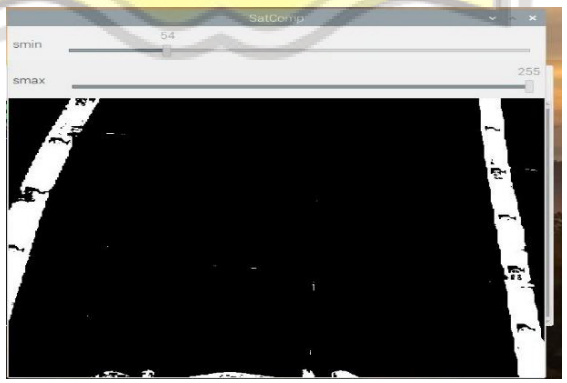
Pengujian *HSV* dilakukan untuk memeriksa pengolahan objek marka jalan pada jalur yang telah ditentukan. Dalam pengujian ini digunakan metode thresholding yaitu mengukur batas maksimum dan minimum. Pengujian ini dilakukan untuk membedakan warna putih dan hitam yang terdapat pada metode *HSV* yang memiliki nilai maksimum atau minimum.

Untuk mengetahui batas maksimum dan minimum dari metode *HSV*, dilakukan pengujian dengan membedakan filter dari benda hitam dan benda putih. Pengujian ini dilakukan dengan mengaktifkan program yang terdapat pada Raspberry Pi 4 Model B. Gambar 10,11, dan 12 merupakan pengujian metode *HSV*.

Gambar 4.9. Tes Threshold *Hue*

Hue Min	*Hue* Max	White Object	Black Object
155	179	Detected	Not Detected

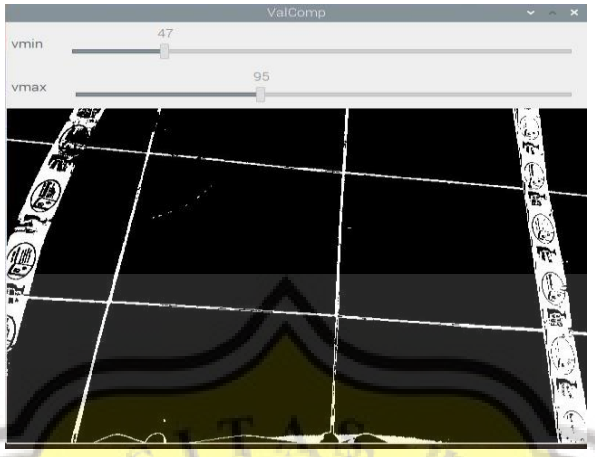
Tabel.II.Tes Threshold *Hue*



Gambar 4.10 Tes Threshold *Saturation*

Sat Min	Sat Max	White Object	Black Object
54	255	Detected	Not Detected

Tabel.III. Tes Threshold *Saturation*



Gambar 4.11 Tes Threshold *Value*

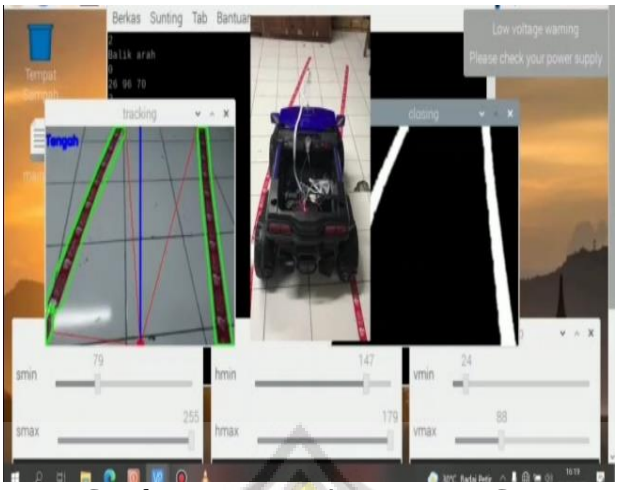
Sat Min	Sat Max	White Object	Black Object

{

Berdasarkan Gambar 4.9, 4.10 dan 4.11. Nilai kalibrasi thresholding ditentukan dengan mencari nilai minimum dan maksimum dari metode *HSV* (*Hue*, *Saturation*, *Value*), yang bertujuan untuk mendeteksi objek putih dan hitam di bawah pengaruh deteksi objek marka jalan

4.6. Hasil Akhir Deteksi lintasan Street Marak pada *Autonomous Car*

Pada pengujian terakhir ini, kita dapat melihat bahwa mobil otonom bergerak mengikuti objek marka jalan yang telah diproses berdasarkan metode *HSV* secara *Real-Time* sehingga muncul pelacakan pada setiap jalur yang terdeteksi kamera. Munculnya jalur tracking ini membuat mobil otonom berjalan dengan stabil mengikuti jalur yang sudah terlihat. Selain itu, tracking line berfungsi untuk menentukan sudut belok pada mobil otonom agar sistem *Self-Driving Car* dapat berfungsi dengan baik sesuai dengan algoritma yang telah dibuat.



Gambar 4.12. Tes *Autonomous Car*

Berdasarkan Gambar 4.12, pengujian telah dilakukan di dalam dan di luar ruangan. Untuk mendapatkan hasil yang maksimal sebaiknya uji coba mobil self-driving ini di dalam ruangan karena berdasarkan pengujian yang dilakukan di dalam ruangan tidak ada gangguan dengan intensitas sinar matahari. Sementara di pengujian di luar ruangan, kekuatan matahari sangat tinggi, sehingga kinerja metode *HSV* tidak optimal.