Jurnal Rekayasa Elektrika Vol. 18, No. 4, Desember 2022, hal. 255-262
ISSN. 1412-4785; e-ISSN. 2252-620X, Terakreditasi Dikbudristekdikti No. 105/E/KPT/2022
DOI: 10.17529/jre.v18i4.27495

Penerapan Algoritma HSV pada Autonomous Car untuk Sistem Self-Driving Berbasis Raspberry Pi 4

Florentinus Budi Setiawan, Padang Ufqi Sutrisno, Leonardus Heru Pratomo, dan Slamet Riyadi
Program Studi Teknik Elektro, Universitas Katolik Soegijapranata
Jl. Pawiyatan Luhur IV/1, Semarang 50234
e-mail: f.budi.s@unika.ac.id

***Abstrak*—Perkembangan teknologi di sektor transportasi di masa ini semakin krusial. Sehingga perusahaan berinovasi menciptakan mobil yang dapat berjalan sendiri dengan tingkat keamanan yang tinggi. Pada penelitian ini, kami merancang sistem *self-driving* untuk mobil RC skala 1:10 menggunakan komponen utama berupa Raspberry Pi 4 sebagai pengolahan citra untuk kendali otomatis pada *autonomous car*. Untuk mengatur pergerakan roda belakang dan steering menggunakan motor DC. Penelitian ini menerapkan *computer vision* yang dipakai untuk sistem navigasi agar dapat berjalan sesuai dengan lintasan. Permasalahan yang dijumpai pada penelitian sebelumnya adalah masih mengambil sampel lintasan terlebih dahulu yang dirasa kurang efisien karena pada jalan yang belum diambil sampelnya tidak dapat dilalui robot tersebut. Untuk memecahkan permasalahan ini maka peneliti menerapkan algoritma *Hue Saturation Value* (HSV) agar dapat mengikuti lintasan secara *real-time*. Algoritma HSV merupakan sistem untuk mendeteksi tepi garis lintasan dengan memproses gambar dari kamera Raspberry Pi. Dari hasil kalibrasi nilai *threshold* yang digunakan adalah sebesar H_{min}=135 dan H_{max}=179, S_{min}=70 dan S_{max}=255, dan nilai V sebesar V_{min}=53 dan V_{max}=106 agar dapat mendeteksi jalur lintasan secara jelas, baik di dalam ruangan maupun diluar ruangan. Algoritma HSV memiliki toleransi terhadap perubahan intensitas cahaya. Berdasarkan hasil pengujian dan implementasi robot ini dengan menggunakan kecerdasan buatan dapat bekerja sesuai dengan algoritma yang sudah dibuat dengan tingkat akurasi deteksi jalur yang cukup tinggi.**

Kata kunci: ***autonomous car, self-driving, hsv, computer vision, raspberry pi***

***Abstract*—The development of technology in the transportation sector at this time is increasingly crucial. Hence the company innovates to create a car that can run itself with a high level of security. In this study, we designed a self-driving system for a 1:10 scale RC car using the main component in the form of a Raspberry Pi 4 as image processing for automatic control of an autonomous car. To regulate the movement of the rear wheels and steering using a DC motor. This study applies Computer Vision which is used for the navigation system so that it can run according to the trajectory. The problem encountered in previous research is that it is still taking samples of the path first which is less efficient because on roads that have not been sampled the robot cannot pass. To solve this problem, the researchers applied the Hue Saturation Value (HSV) algorithm to able to follow the trajectory in real-time. The HSV algorithm is a system for detecting the edge of the trajectory line by processing images from the Raspberry Pi camera. From the calibration results, the threshold value used is H_{min}=135 and H_{max}=179, S_{min}=70 and S_{max}=255, and the V value is V_{min}=53 and V_{max}=106 in order to detect the trajectory clearly, both indoors and outdoors. The HSV algorithm has tolerant to changes in light intensity. That is the advantage of the HSV Algorithm. Based on the results of testing and implementation of this robot using artificial intelligence can work according to the algorithm that has been made with a fairly high level of path detection accuracy.**

Keywords: ***autonomous car, self-driving, hsv, computer vision, raspberry pi***

I. Pendahuluan

Perkembangan teknologi di sektor transportasi di masa ini semakin krusial. *Self-driving* bukan hanya sekedar mimpi tetapi menjadi sebuah kenyataan. Terbuktinya di masa ini perusahaan berinovasi menciptakan mobil yang dapat berjalan sendiri yang dapat membantu manusia untuk berkendara dari tujuan pertama sampai tujuan berikutnya [1], [2]. *Self-driving* adalah suatu cara yang bertujuan untuk meringankan aktivitas mengemudi. *Self-driving* juga disebut sebagai *autonomous car* atau tanpa pengemudi merupakan mobil yang beroperasi dan bernavigasi menggunakan kecerdasan buatan. Manfaat mobil *Self-driving* dikatakan dapat membantu mengatasi masalah kelalaian pengemudi yang berakibat kecelakaan mobil khususnya di indonesia [3], [4]. Saat merancang mobil *Self-driving* salah satu yang terpenting adalah kemampuan mengenali jalur yang merupakan sebuah

Received 10 August 2022; Revised 24 October 2022; Accepted 15 November 2022

parameter agar mobil tetap stabil dalam lintasan. Penelitian ini mengangkat topik tentang mobil *Self-driving* atau bisa juga disebut *autonomous car* yang berfokus untuk mendeteksi garis tepi jalur, parameter yang dipakai adalah pada garis tepi jalur menggunakan algoritma ruang warna *Hue Saturation Value* (HSV) [5].

Dalam penelitian ini algoritma HSV sanggup menangkap pola jalur pada jarak berapapun asalkan terlihat jelas oleh kamera *Raspberry Pi* itu sendiri, karena resolusi dan kualitas kamera memainkan peran penting dalam pemrosesan gambar digital dengan algoritma HSV ini [6]. Oleh karena itu, penelitian ini menciptakan sistem *self-driving* pada sebuah prototipe yang dapat bergerak secara otomatis sepanjang jalur dengan menggunakan kamera sebagai sensornya, yang sebelumnya pola atau jalur di proses oleh *computer vision* terlebih dahulu dengan menggunakan algoritma ruang warna HSV dan filter *Gaussian* untuk menghaluskan atau menghilangkan *noise* pada proses deteksi garis tepi jalur agar hasil lebih baik. Salah satu komponen terpenting pada penelitian ini adalah *Raspberry Pi* yang berfungsi sebagai komputer pengolahan data. *Raspberry Pi* adalah modul komputer kecil dengan *input* dan *output* yang mirip dengan papan mikrokontroler lainnya [7]. Meskipun *Raspberry Pi* berukuran kecil, namun bisa bekerja mengendalikan program berat yang dipakai dalam perkantoran, permainan komputer, dan pula bisa dipakai buat pemutar media menggunakan resolusi tinggi. Selain itu Raspberry Pi memiliki port GPIO yang diperlukan guna untuk menghubungkan motor DC pada roda dan *steering* maupun perangkat lainnya [8].

Penelitian ini merupakan penyempurnaan dari sistem *self-driving* dimana *autonomous car* yang dibuat oleh peneliti ini menggunakan kamera sebagai *input* pembacaan jalur dengan algoritma HSV yang memiliki tingkat akurasi yang tinggi dengan mengatur *threshold* untuk menyempurnakan pembacaan jalur *autonomous car* yang dibuat oleh peneliti ini apabila dibandingkan dengan penelitian sebelumnya [9], dimana sistem masih menggunakan sampel jalan yang kurang efisien karena pada jalan yang belum disampel tidak dapat dilalui oleh robot tersebut [10]. Penggunaan metode filter warna HSV ini digunakan karena robot dapat dioperasikan secara *real time* dan tidak perlu dilakukan adanya *training* model jalan yang hendak dilalui. Uji coba dari *autonomous car* dengan algoritma HSV ini akan sajikan pada bagian bab hasil dan pembahasan.

II. Metodologi Penelitian

Alur penelitian yang dilakukan penulis ditunjukkan pada Gambar 1. Metode penelitian ini meliputi studi literatur, perancangan dan pembuatan prototipe, hasil dan pembahasan, serta kesimpulan berdasarkan hasil pengujian yang telah dilakukan. Studi literatur dilakukan untuk mengidentifikasi masalah dan menemukan berbagai solusi untuk memecahkan masalah tersebut. Setelah mengetahui permasalahannya, *autonomous car* yang telah dirancang diimplementasikan mengikuti solusi dari permasalahan

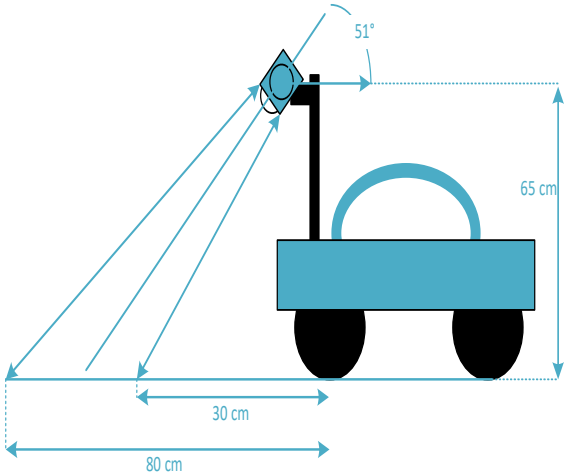
Gambar 1. Flowchart Penelitian

yang ditemukan. kelebihan penerapan algoritma HSV pada *autonomous car* ini bisa dikatakan memiliki tingkat akurasi pembacaan jalan yang cukup tinggi. Penelitian ini juga menguji fungsionalitas yang dilakukan untuk menentukan efek dari intensitas cahaya yang berbeda. Pada pengujian ini mencoba dua kondisi yaitu di *indoor* dan di *outdoor*.

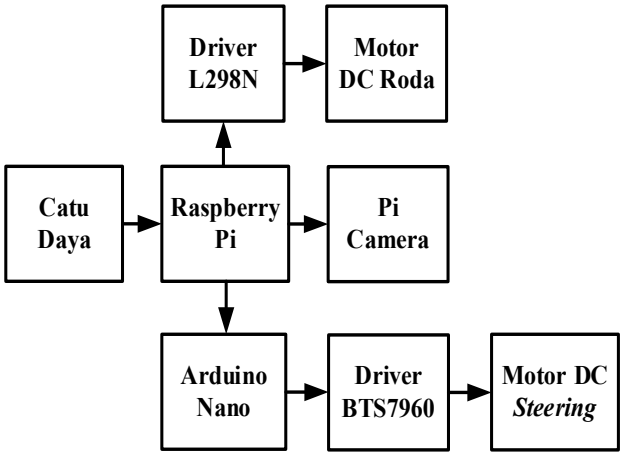
A. Perancangan Hardware

Hardware yang digunakan pada penelitian ini menggunakan mobil RC 4WD skala 1:10. Untuk kontrol *steering* dan kontrol kecepatan putaran pada roda menggunakan sistem aktuator yang dipakai yaitu motor DC. Sensor yang dipakai dalam penelitian ini adalah *Raspberry Pi camera v2* diletakkan di bagian depan mobil dengan menggunakan tempat buatan tangan berbahan dasar akrilik. Tempat kamera *Raspberry Pi* ini diletakkan 51^0 terhadap lantai dengan tinggi 65 cm di atas permukaan lantai. Pemasangan ini bertujuan agar kamera *Raspberry Pi* dapat mengidentifikasi objek yaitu berupa jalur lintasan. Gambar penempatan kamera bisa dilihat pada Gambar 2.

Dalam perancangan prototipe pada *autonomous car* menggunakan catu daya berupa *accu* dengan tegangan 12V kapasitas 12AH sebagai sumber tegangan *Raspberry Pi.* Kontroler dan mikrokontroler yang dipakai *Raspberry*



Gambar 2. Penempatan Kamera



Gambar 3. Diagram Blok Autonomous Car

Pi 4, Arduino Nano, Driver BTS7960 dan Driver L298N. Untuk mengontrol motor DC sebagai penggerak roda pada *autonomous car* menggunakan *Driver L298N*. Untuk penggerak *steering* dikontrol menggunakan *Driver BTS7960* yang terintegrasi langsung dengan program *Arduino Nano.* Beberapa komponen tersebut akan dikontrol menggunakan *Raspberry Pi 4* sebagai kontroler paling utama pada *autonomous car* ini. Diagram blok *autonomous car* bisa dilihat pada Gambar 3.

B. Perancangan Software

Pada perancangan *software* bahasa pemrograman yang digunakan adalah *python* dan *OpenCV. OpenCV* merupakan *library* yang sangat familiar pada pengolahan gambar pada *computer vision*. Menggunakan *computer vision* dapat membuat keputusan, mengambil tindakan, dan mengenali suatu objek. *OpenCV* merupakan sebuah *library* untuk *computer vision*. *OpenCV* dirancang untuk aplikasi *real-time*, dengan fungsi akuisisi yang baik untuk pemrosesan gambar atau video [11].

Mini computer Raspberry Pi memiliki tugas untuk mengolah gambar yang ditangkap oleh kamera untuk memberikan data koordinat objek. Mikrokontroler *Raspberry Pi* menggunakan algoritma HSV untuk melakukan operasi filter warna dan melakukan perhitungan koordinat objek. Proses diperintahkan menggunakan program bahasa *python*. Perintah pemrosesan gambar digital dapat menggunakan *library computer vision* [12].

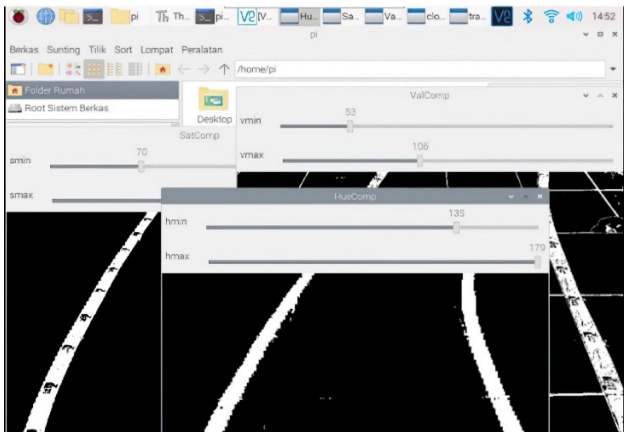
Gambar 4 menampilkan bentuk *interface* yang dirancang dalam desain sistem *self-driving* ini. Tampilan *interface* didesain guna menyederhanakan metode *thresholding* dan menetapkan nilai *minimum* dan *maximum* pada komponen H, S dan V.

Di *windows "Hue Filter "Sat Filter*" dan "*Val Filter"* memperlihatkan *trackbar* untuk mengubah variabel *minimum* dan *maximum* dari *threshold* masing masing komponen. *Windows* juga menampilkan hasil filter pada filter *hue, saturation*, dan *value*. *Windows* menampilkan hasil dari ketiga proses tersebut apakah objek terdeteksi atau tidak [13].

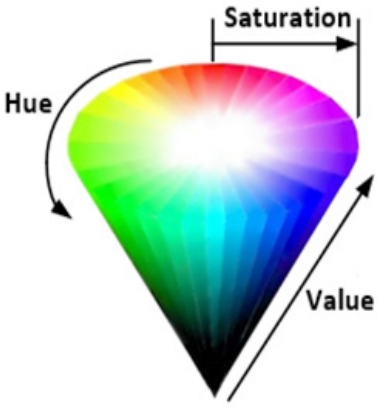
C. Algoritma HSV

HSV merupakan suatu algoritma untuk menentukan warna yang diinginkan. Dalam penelitian ini tingkat akurasi dan kemampuan mengenali jalur menggunakan teknik filter warna HSV sangat cocok. HSV merupakan sistem untuk mendeteksi tepi garis lintasan melalui proses gambar dari *Raspberry Pi camera v2* menggunakan proses *color filtering*. Keuntungan menggunakan algoritma ini adalah warna yang ditangkap lebih mendekati oleh penglihatan mata manusia [14]. Algoritma HSV sangat baik dalam mengidentifikasi warna primer, dan HSV toleran terhadap perubahan intensitas cahaya. Ini adalah keunggulan HSV dibandingkan ruang warna lainnya [15]. Ruang warna algoritma HSV memiliki komposisi yang merupakan kombinasi dari warna primer biru, merah dan hijau ditunjukkan seperti Gambar 5.

HSV memiliki tiga karakteristik pokok, yaitu *Hue, Saturation* dan *Value* yang mempunyai nilai yang berbeda [16]. Jenis tampilan warna *hue* merupakan lingkaran dengan sudut 360°. Nilai *hue* dirubah pada 0 sampai 255, dengan nilai 0 menjadi warna merah. *Saturation* menggambarkan intensitas warna yang mencakup nilai 0 sampai 255. Jika nilai *saturation* kurang akan menunjukan warna abu-abu. *Value* mencakup nilai 0 sampai 255, dengan 0 menjadi warna gelap dan nilai 255 menjadi warna cerah [17].



Gambar 4. Tampilan Interface



Gambar 5. Ruang Warna Algoritma HSV

Tabel 1. Notasi persamaan konversi nilai RGB

Notasi	Keterangan
H	Hue
S	Saturation
V	Value
R	Red
G	Green
B	Blue
Min	Nilai Minimal
Max	Nilai Maksimal

Nilai HSV diperoleh dari konversi nilai RGB. Konversi RGB dapat dijelaskan pada persamaan berikut, dengan notasi yang dijelaskan pada Tabel 1.

$$H = \tan\left[\frac{3(G-B)}{(R-G)+(R-B)}\right] \quad (1)$$

$$S = 1-\left[\frac{\min(R,G,B)}{V}\right] \quad (2)$$

$$V = \frac{R+G+B}{3} \quad (3)$$

Pada persamaan (1)-(3) ketika S=0 bahwa nilai H belum bisa ditentukan. Maka harus normalisasi nilai RGB lebih awal, persamaannya seperti berikut:

$$r = \frac{R}{R+G+B} \quad (4)$$

$$g = \frac{G}{R+G+B} \quad (5)$$

$$b = \frac{B}{R+G+B} \quad (6)$$

Dengan nilai normalisasi ini, maka persamaan konversi nilai dari RGB ke nilai HSV adalah seperti berikut:

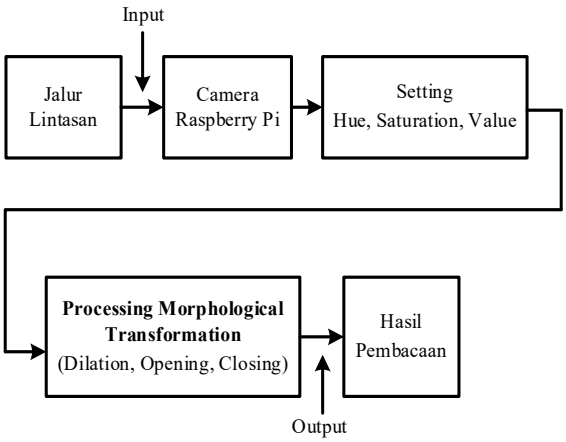
$$V = \max(r,g,b) \quad (7)$$

$$S = \begin{cases} 0, & \text{jika } V = 0 \\ 1-\dfrac{\min(r,g,b)}{V}, & \text{jika } V > 0 \end{cases} \quad (8)$$

$$H = \begin{cases} 0, & \text{jika } S = 0 \\ \dfrac{60*(g-b)}{S*V}, & \text{jika } V = r \\ 60*\left[2+\dfrac{(b-r)}{S*V}\right], & \text{jika } V = g \\ 60*\left[4+\dfrac{(r-g)}{S*V}\right], & \text{jika } V = b \end{cases} \quad (9)$$

$$H = H+360 \qquad \text{jika } H < 0 \quad (10)$$

Sistem yang digunakan pada HSV terdiri dari jalur lintasan. Setelah jalur lintasan yang akan dideteksi oleh



Gambar 6. Diagram Blok HSV

camera Raspberry Pi. Setelah kamera mendeteksi maka akan dilanjutkan dengan proses kalibrasi nilai *threshold* pada HSV. *Setting* nilai *threshold* difungsikan untuk menentukan deteksi pola lintasan. Kemudian akan diproses lagi dengan menggunakan *morphologi transformation* dengan melalui proses *dilation, opening* dan *closing*[18]. Setelah semua proses dilalui maka hasil akan terlihat dilayar monitor. Diagram blok HSV bisa dilihat pada Gambar 6.

D. Filter Gaussian

Filter *gaussian* untuk menghaluskan atau menghilangkan *noise* saat mendeteksi garis tepi jalur dengan mengubah gambar *input* menjadi gambar abu abu atau blur [19]. Gambar blur diperoleh dengan melakukan *konvolusi kernel gaussian* ditunjukan pada persamaan (11), (12).

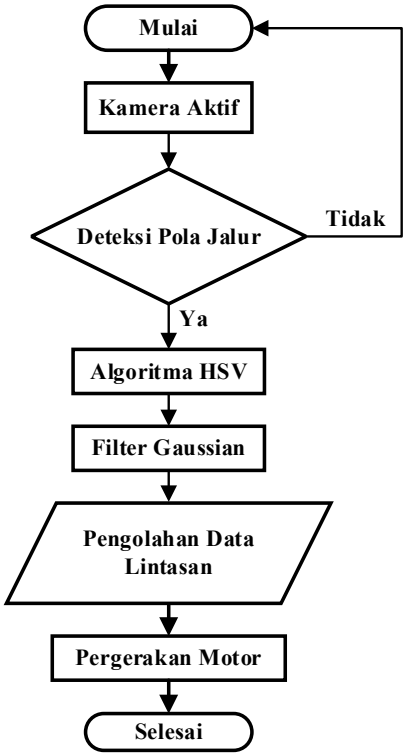
$$G(x,y) = F(x,y) * H(x,y) \quad (11)$$

$$H(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{X^2+Y^2}{2\delta^2}} \quad (12)$$

Pada persamaan (11) dan (12), x adalah posisi titik horizontal, y adalah posisi titik vertikal, $G(x,y)$ mewakili gambar yang dihaluskan, $F(x,y)$ mewakili gambar *input*, dan $H(x,y)$ mewakili filter *gaussian* yang dipilih, adalah σ nilai standar eror, e adalah konstanta dengan nilai 2,718281828, π adalah konstanta dengan nilai 3,14.

E. Proses Kerja Autonomous Car

Langkah awal pada *flowchart* kamera aktif dalam bentuk video. Selanjutnya kamera mendeteksi pola jalur. Jika pola tidak terdeteksi maka program tidak berjalan dan akan kembali ke proses awal [20]. Jika pola terdeteksi, maka proses algoritma HSV bekerja dimana HSV ini mampu mengolah ataupun memfilter warna yang dibutuhkan selanjutnya filter *gaussian* untuk menghaluskan atau menghilangkan *noise* pada proses deteksi garis tepi jalur agar hasil lebih baik. Kemudian proses selanjutnya ketika sudah berhasil di filter, hasil dari pembacaan akan



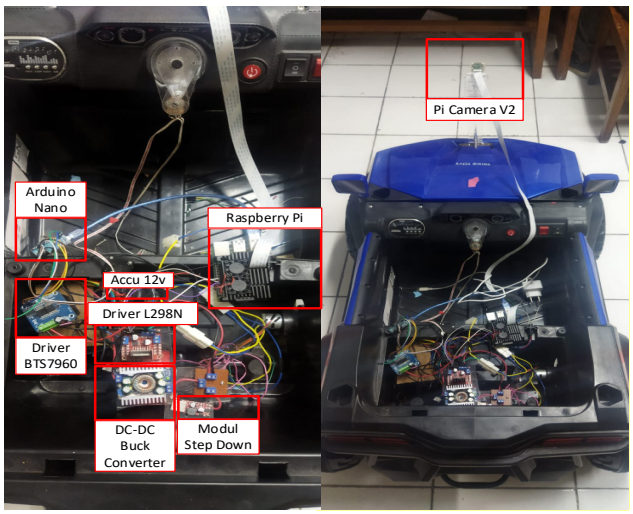
Gambar 7. Flowchart Proses Kerja Autonomous Car

diinterkoneksikan dengan motor penggerak yang akan bergerak sesuai dengan lintasan yang sudah diproses oleh kamera. Untuk melihat *flowchart* proses kerja *autonomous car* dapat dilihat pada Gambar 7.

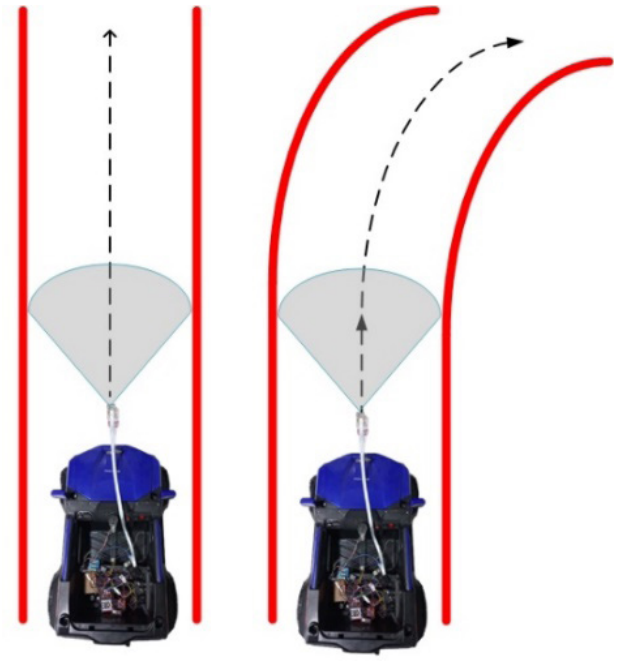
III. Hasil dan Pembahasan

Hasil akhir prototipe ditunjukkan pada Gambar 8. Terdiri dari beberapa komponen yang sudah terpasang seperti *Raspberry Pi, Arduino nano, Pi Camera v2, Driver L298N, Driver BTS7960, DC-DC Buck Converter, Modul step down* dan baterai dengan tegangan 12 V.

Dari hasil penelitian ini bisa didapatkan melalui beberapa proses pengujian yaitu dengan proses *color filtering*, pengujian fungsionalitas dan yang terakhir adalah pengujian deteksi jalur. Tujuan utama dari beberapa proses pengujian ini adalah agar sistem *self-driving* menggunakan algoritma HSV mendapat hasil yang maksimal. Dalam proses kalibrasi dapat mengatur terlebih dahulu pada nilai *threshold H*, *S* dan *V*. Pola lintasan yang digunakan pada *autonomous car* ini ditunjukkan pada Gambar 9. *Autonomous car* ini diuji dengan kondisi yang berbeda yaitu di *indoor* dan *outdoor* dan pola lintasan yang digunakan pada penelitian ini Menggunakan 2 jalur dengan kondisi berbelok dan kondisi lurus. Kelemahan dari sistem yang ditanamkan pada mobil ini adalah ketika dioperasikan pada kondisi pencahayaan yang berbeda maka harus dilakukan kalibrasi ulang untuk mendeteksi jalur yang hendak dibaca oleh kamera. Selama intesitas cahaya tidak terlalu ekstrim maka tidak perlu dilakukan kalibrasi nilai HSV.



Gambar 8. Prototipe Autonomous Car



Gambar 9. Pola Lintasan Autonomous Car

A. Pengujian Color Filtering

Dengan melakukan kalibrasi *min* ke *max* dan *max* ke *min*. Pada percobaan ini kalibrasi *min* ke *max* diberikan nilai 0, dan pengujian kalibrasi *max* ke *min* dengan memberikan nilai *max* 179 untuk *H* , 255 untuk *S* dan *V*. Dengan cara menggeser nilai *max* atau *min* sampai terjadinya perubahan pada objek pola. Tabel 2 menunjukkan uji coba *threshold H (hue)*. Tabel 3 menunjukkan uji coba *threshold S (saturation)*. Tabel 4 menunjukkan uji coba *threshold V (value)*.

Pada Tabel 2 sampai dengan Tabel 4 menggunakan metode *thresholding* dimana pada proses ini memerlukan setting nilai yang sesuai sehingga ditemukan nilai yang akurat. Kemudian dilakukan uji coba melalui perubahan nilai *min* ke *max* dan *max* ke *min*. Metode diperlukan untuk menentukan warna hitam maupun putih pada jalur.

Tabel 2. Uji Coba Threshold Hue

Kalibrasi	H_{min}	H_{max}	Warna Putih	Warna Hitam
Min ke Max	0	135	Terdeteksi	Tidak
Max ke Min	137	179	Tidak	Terdeteksi

Tabel 3. Uji Coba Threshold Saturation

Kalibrasi	Smin	Smax	Warna Putih	Warna Hitam
Min ke Max	0	70	Terdeteksi	Tidak
Max ke Min	72	255	Tidak	Terdeteksi

Tabel 4. Uji Coba Threshold Value

Kalibrasi	Vmin	Vmax	Warna Putih	Warna Hitam
Min ke Max	0	53	Terdeteksi	Tidak
Max ke Min	60	255	Tidak	Terdeteksi

Tabel 5. Pengujian Fungsionalitas

Threshold Saturation	Threshold Value	Kondisi Indoor	Kondisi Outdoor
0-20	200-255	Jalur Tidak Terdeteksi	Jalur Tidak Terdeteksi
10-50	130-255	Jalur Terdeteksi	Terdeteksi Banyak Noise
70-255	53-106	Jalur Terdeteksi	Jalur Terdeteksi

B. Pengujian Fungsionalitas

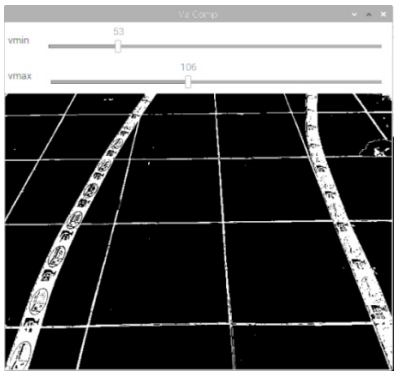
Pengujian fungsionalitas dilakukan untuk menentukan efek dari intensitas cahaya yang berbeda. Pada pengujian ini mencoba dua kondisi yaitu di *indoor* dan di *outdoor*. Pada pengujian *indoor* menggunakan penerangan lampu TL sedangkan pada pengujian di *outdoor* langsung menggunakan penerangan dari sinar matahari tanpa hambatan apapun.

Tabel 5 menunjukan bahwa menetapkan nilai *threshold* yang dibawah kondisi lokasi yang berbeda dapat mempengaruhi persepsi proses filter warna yang dilakukan. Tetapi jika nilai *threshold saturation* adalah 70-255 dan nilai *threshold value* 53-106 maka sistem dapat beradaptasi dengan kondisi pencahayaan di *indoor* dan *outdoor.*

C. Pengujian Deteksi Jalur

Gambar 10. Penentuan Nilai Hue

Gambar 11. Penentuan Nilai Saturation



Gambar 12. Penentuan Nilai Value

Pada penelitian ini nilai H sebesar H_{min}=135 dan H_{max}=179, nilai S sebesar S_{min}=70 dan S_{max}=255, dan nilai V sebesar V_{min}=53 dan V_{max}=106, agar alat dapat mendeteksi jalur yang ditentukan. Pada proses uji coba ini menggunakan metode *thresholding* untuk membedakan warna hitam dan putih. Penentuan nilai HSV bisa dilihat pada Gambar 10, 11 dan 12.

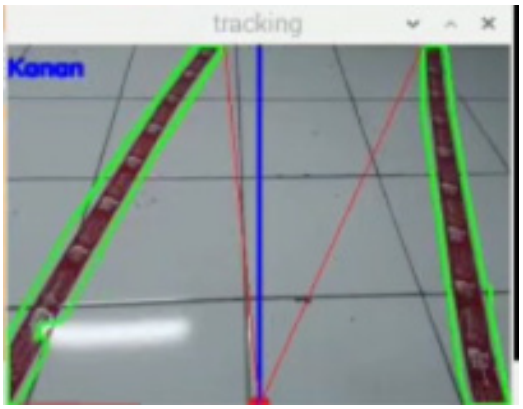
D. Hasil Keluaran

Hasil keluaran merupakan jendela hasil akhir dimana sudah melalui proses *thresholding* dan *gaussian* filter yang menghasilkan *filtering* warna yang bagus tanpa *noise*. Setelah melalui proses penentuan nilai HSV maka hasil akhir akan terlihat pada Gambar 13.

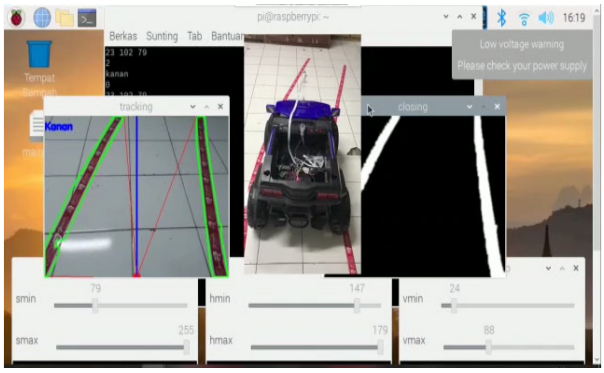
E. Hasil Percobaan Self-Driving

Pada tahap ini dilakukan uji coba *self-driving* menggunakan algoritma HSV terhadap jalur lintasan. Pada proses ini mobil dapat berjalan otomatis mengikuti jalur secara *real-time*. Pergerakan mobil ditentukan dari proses *tracking* menggunakan sumbu (*x,y*) dimana memanfaatkan titik tengah yang berwarna biru untuk mengatur kondisi pergerakan mobil. Jika sudut garis merah lebih besar ke kanan maka mobil akan berbelok kanan. Jika sudut garis merah lebih besar ke kiri maka mobil akan berbelok kiri. Jika posisi sudut sama besarnya maka mobil akan berjalan lurus. Untuk hasil *tracking* dan percobaannya dapat diamati pada Gambar 14 dan 15.

Gambar 13. Hasil Keluaran



Gambar 14. Hasil Tracking



Gambar 15. Hasil Percobaan Self-Driving

IV. Kesimpulan

Berlandaskan pengujian yang sudah dilakukan, bisa disimpulkan bahwa sistem *self-driving* pada *autonomous car* dengan menggunakan algoritma HSV berbasis *Raspberry Pi 4* model B berhasil dirancang sesuai spesifikasi desain sistem *self-driving* dan bisa berjalan sesuai rute yang sudah ditentukan. Dari hasil kalibrasi nilai *threshold* yang digunakan adalah sebesar H_{min}=135 dan H_{max}=179, nilai S sebesar S_{min}=70 dan S_{max}=255, dan nilai V sebesar V_{min}=53 dan V_{ma}=106 nilai pengaturan *threshold* ini digunakan disesuaikan dengan kondisi lapangan saat dilakukan uji coba robot ini. Dengan menggunakan nilai HSV yang dicantumkan diatas tidak bisa dijadikan patokan karena unutk mengoperasikan robot ini harus dilakukan kalibrasi karena kondisi di setiap lapangan akan memiliki intensitas cahaya yang berbeda beda. Keuntungan menggunakan algoritma HSV ini adalah warna yang ditangkap lebih mendekati oleh penglihatan mata manusia. selain itu HSV mentolerir perubahan intensitas cahaya. Penggunaan metode HSV ini lebih akurat dan bisa mengikuti lintasan secara *real-time* dibandingkan dengan menggunakan metode *neural network,* karena metode itu harus mengambil sampel gambar lintasan terlebih dahulu. Jika lintasan yang belum diambil sampelnya tidak dapat dilalui oleh *autonomous car* tersebut. Itulah keuntungan dari HSV dibandingkan dengan metode lainnya.

Referensi

[1] H. Thadeshwar, V. Shah, and M. Jain, “Artificial intelligence based self-driving car,” in *Proc. 4th International Conference on Computer, Communication and Signal Processing*, Jan 2021, pp. 1–5.

[2] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, “Self-driving cars,” *Computer*, vol. 50, no.12, pp.18–23, Dec. 2017.

[3] I. Ahmad and K. Pothuganti, “Design implementation of real time autonomous car by using image processing IoT,” in *Proc. 3rd Int. Conf. Smart Syst. Inven. Technol. ICSSIT 2020*, no. Icssit, pp. 107–113, 2020.

[4] T. A. S. Nielsen and S. Haustein, “On sceptics and enthusiasts: What are the expectations towards self-driving cars?,” *Transp. Policy*, vol. 66, no. April 2017, pp. 49–55, 2018.

[5] T. Zhang, H. M. Hu, and B. Li, “A Naturalness Preserved Fast Dehazing Algorithm Using HSV Color Space,” *IEEE Access*, vol. 6, no. c, pp. 10644–10649, 2018.

[6] A. A. Mahersatillah, Z. Zainuddin, and Y. Yusran, “Unstructured Road Detection and Steering Assist Based on HSV Color Space Segmentation for Autonomous Car,” in *Proc. 3rd Int. Semin. Res. Inf. Technol. Intell. Syst. ISRITI 2020*, pp. 688–693, 2020.

[7] K. Kato and M. Wada, “A Kinematic Analysis of a Omnidirectional Robot with the Active-caster Robotic Drive with a Ball Transmission. in *Proc. of Inter. Symp on System Integration*, pp. 1318-1323, Jan. 2019.

[8] B. C. Z. Blaga, M. A. Deac, R. W. Y. Al-Doori, M. Negru, and R. Danescu, “Miniature autonomous vehicle development on raspberry Pi,” in *Proc. - 2018 IEEE 14th Int. Conf. Intell. Comput. Commun. Process. ICCP 2018*, pp. 229–236, 2018.

[9] L. R. Manangka *et al.*, “Pengenalan Pola Lintasan Berbasis Neural Network Pada Prototype Self-Driving Car,” vol. 12, no. 2, pp. 67–72, 2020.

[10] T. D. Do, M. T. Duong, Q. V. Dang, and M. H. Le, “Real-Time Self-Driving Car Navigation Using Deep Neural Network,” in *Proc. 4th Int. Conf. Green Technol. Sustain. Dev. GTSD 2018*, pp. 7–12, 2018.

[11] Z. Wang, Y. Fan, and H. Zhang, “Lane-line Detection Algorithm for Complex Road Based on OpenCV,” in *Proc. 2019 IEEE 3rd Adv. Inf. Manag. Commun. Electron. Autom. Control Conf. IMCEC 2019*, no. Imcec, pp. 1404–1407, 2019.

[12] J. Huang, B. Kong, B. Li, and F. Zheng, “A new method of unstructured road detection based on HSV color space and road features,” in *Proc. 2007 Int. Conf. Inf. Acquis. ICIA*, vol. 255, pp. 596–601, 2007.

[13] Z. Sun, “Vision Based Lane Detection for Self-Driving Car,” in *Proc. IEEE Int. Conf. Adv. Electr. Eng. Comput. Appl. AEECA 2020*, pp. 635–638, 2020.

[14] S. O. Ali Chishti, S. Riaz, M. Bilal Zaib, and M. Nauman, “Self-Driving Cars Using CNN and Q-Learning,” in *Proc. 21st Int.*

Multi Top. Conf. INMIC , 2018.

[15] J. X. Zhao and M. Y. Liu, "A color HSV image edge detection method based on gradient extreme value," in *Proc. 2nd Int. Symp. Intell. Inf. Technol. Appl. IITA 2008*, vol. 3, pp. 381–384, 2008.

[16] K. Jo, Y. Jo, J. K. Suhr, H. G. Jung, and M. Sunwoo, "Precise Localization of an Autonomous Car Based on Probabilistic Noise Models of Road Surface Marker Features Using Multiple Cameras," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3377–3392, 2015.

[17] A. Aqthobilrobbany, A. N. Handayani, D. Lestari, Muladi, R. A. Asmara, and O. Fukuda, "HSV Based Robot Boat Navigation System," in *Proc. Int. Conf. Comput. Eng. Network, Intell. Multimed. 2020*, pp. 269–273, 2020.

[18] R. Mondal, M. S. Dey, and B. Chanda, "Image Restoration by Learning Morphological Opening-Closing Network," *Math. Morphol. - Theory Appl.*, vol. 4, no. 1, pp. 87–107, 2020.

[19] Y. Lin, H. Zhao, C. Ye, and H. Ding, "A computationally efficient and robust kinematic calibration model for industrial robots with kinematic parallelogram," in *Proc. IEEE Int. Conf. Robot. Biomimetics, ROBIO 2017*, pp. 1–6, 2018.

[20] W. Lim, S. Member, S. Lee, S. Member, M. Sunwoo, and K. Jo, "Hierarchical Trajectory Planning of an Autonomous Car Based on the Integration of a Sampling and an Optimization Method," vol. 19, no. 2, pp. 613–626, 2018.