DESAIN DAN IMPLEMENTASI PENGOLAHAN CITRA UNTUK KENDALI MOBILE ROBOT SECARA REALTIME BERBASIS LABVIEW DAN NI MY RIO-1900

Junnisha R. Fatimah[1], Afaf Fadhil Rifa'i, ST., MT.[2], Adhitya Sumardi Sunarya S.Si., M.Si.[3]
Jurusan Teknik Mesin dan Manufaktur, Konsentrasi Teknik Elektromekanik
Politeknik Manufaktur Bandung
Jl. Kanayakan No. 21 – Dago, Bandung - 40135
Phone/Fax : 022. 250 0241 / 250 2649
Email: junnisha27@gmail.com[1] , afaffadhil@gmail.com[2], adhitya102@yahoo.com[3]

ABSTRAK

Persepsi visual manusia mampu membangun suatu sistem kecerdasan. Dimulai dari melihat, mengolah, hingga mengekstrak ciri-ciri objek yang dilihat sampai mengenali objek tersebut. Dari persepsi visual manusia ini, sistem kecerdasan tersebut dapat diterapkan untuk mengendalikan suatu robot. Sensor kamera dapat dijadikan sebagai indera visual pada *mobile robot* untuk mendapatkan informasi data visual mengenai lingkungan. Selain didapatkannya data visual, diperlukannya *controller* untuk suatu pengolahan data citra dalam pengendalian *mobile robo*t tersebut. Kemampuan pengolahan data citra tersebut dilihat dari kemampuan maksimal *controller* yang digunakan untuk mendapatkan hasil pengolahan citra menjadi suatu pergerakan *mobile robot*.

Penelitian ini bermaksud untuk mengembangkan pengolahan data citra menggunakan *controller* NI *MyRio*-1900 dengan menggunakan metode *edge detection* pada *software LabVIEW* dibantu dengan VI *Express* khusus yaitu NI *Vision Acquisition* dan NI *Vision Assistant* untuk mengolah data citra menjadi suatu perintah untuk mengendalikan *mobile robot*. Penelitian ini menggunakan operator *Sobel* dengan *template* 3x3 untuk dapat mendeteksi tepi garis (*edge*) dari jalur yang digunakan dengan lebar jalur 300mm dan tebal garis sebesar 20 mm pada jalur. Dari jalur tersebut digunakan konsep matematika dasar untuk mendapatkan nilai *gradien* dari persamaan garis lurus dua titik untuk pergerakan posisi *mobile robot*.

Pada penelitian ini, *Mobile* robot memiliki kemampuan tangkapan citra dengan nilai 29,79 FPS saat robot mendeteksi dalam keadaan diam dan 29,484 FPS saat *mobile robot* mendeteksi dalam keadaan dinamis secara *realtime*. Dengan kemampuan proses dari mulai menangkap gambar, proses *edge detection*, hingga proses perintah arah pergerakan, *mobile robot* memiliki waktu rata-rata 70,75 ms saat mendeteksi tepi dan 79,62 ms saat tidak ada yang terdeteksi dalam 1 kali proses *looping* sistem. Tingkat keberhasilan untuk belok kanan dengan sudut 50^0 sebesar 86,78% dan untuk belok kiri sebesar 91,64%.

Kata kunci: NI *My Rio*-1900, *Edge Detection, Sobel, Gradien*, FPS.

1. PENDAHULUAN

Persepsi visual manusia dapat diterapkan menjadi sistem kecerdasan untuk mengendalikan suatu robot. Sensor kamera dapat dijadikan sebagai indera visual pada robot. Sensor menjadi salah satu bagian penting untuk mendapatkan informasi mengenai lingkungan yang dilalui robot[11]. Sensor kamera ini menjadi salah satu perhatian untuk dijadikan pengembangan penelitian pada *mobile robot* ini. Informasi data visual mengenai lingkungan dapat dengan mudah didapatkan. Selain didapatkannya data visual sebagai data citra, diperlukan juga *controller* untuk suatu pengolahan data citra dalam pengendalian *mobile robo*t tersebut. Kemampuan pengolahan data citra tersebut dilihat dari kemampuan maksimal *controller* yang digunakan untuk mendapatkan hasil pengolahan citra menjadi suatu pergerakan *mobile robot*.

Permasalahan yang terjadi yaitu mengenai kemampuan *controller* yang digunakan untuk pengolahan citra. Pada penelitian [3], [7], dan [6], *controller* yang digunakan untuk pengolahan citra pada pengendalian *mobile robot* adalah *controller Raspberry* PI 3. Sedikitnya 15 *frame per second* untuk pengambilan data citra melalui sensor kamera *webcam* menggunakan *controller Raspberry* PI 3 ini [3]. Selain itu, pengolahan data citra menggunakan *controller Raspberry* PI 3 masih terjadi simpangan dan keterlambatan. Ini terjadi karena semua proses sistem dilakukan oleh *Raspberry* PI 3 sehingga membebani dalam kemampuan proses pengolahan citranya. Simpangan yang terjadi ini berpengaruh pada pergerakan yang dilakukan *mobile robot.* Pergerakan tersebut salah satunya yaitu pergerakan untuk berbelok. Didapatkan simpangan sebesar 20% pada belokan penelitian [7]. Simpangan proses pada *controller Raspberry* PI 3 ini juga menyebabkan terjadinya keterlambatan dalam melangsungkan sistem pemrogramannya. Seperti pada penelitian [6], ketika sensor mendeteksi objek dengan keadaan diam, *controller Raspberry* PI 3 membutuhkan waktu 1,1 detik untuk memproses perintah tersebut sedangkan ketika *mobile robot* yang mendeteksi sambil bergerak

controller membutuhkan waktu 2,5 detik untuk eksekusi pergerakan *mobile robot* selanjutnya.

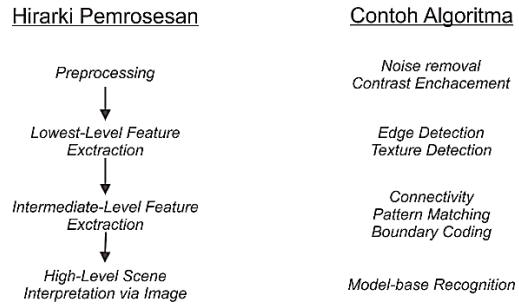
Pada akhirnya, pada penelitian tugas akhir ini dibuat suatu rancangan dan implementasi pemanfaatan pengolahan citra dengan menggunakan sensor kamera *webcam* untuk suatu *mobile robot* dengan *controller* berupa NI *My Rio*-1900. Jika dibandingkan dengan *Raspberry* PI 3, kelebihan dari *controller* ini adalah kemampuan proses pada pengolahan citra yang dapat meningkatkan nilai FPS mendekati FPS maksimal yang ditangkap oleh sensor *webcam*.Selain itu NI *My Rio*-1900 merupakan *controller* bawaan dari *National Instrument* dengan *software* khusus *LabVIEW for My Rio*. Untuk pengolahannya citranya tersendiri pada *LabVIEW* tersedia VI *Express* khusus berupa NI *Vision Acquisition* dan NI *Vision Assistant*. Sistem yang digunakan pada *LabVIEW* ini dapat menjadi dua bagian yang terpisah saat dijalankan, sehingga sistem tidak membebani proses yang dilakukan pada NI *My Rio*-1900. NI *My Rio*-1900 hanya menjalankan proses pengolahannya citranya saja dan pemberian sinyal pada aktuator. Sementara proses lainnya dapat dijalankan di *Personal Computer*. Sehingga pengaruh kemampuan proses pengolahan citra yang terjadi pada pergerakan *mobile robot* secara lurus atau berbelok maupun pengaruh keterlambatan eksekusi yang terjadi menggunakan *controller Raspberry* PI 3 ini dapat diminimalisir oleh kemampuan kecepatan proses pada *controller* NI *My Rio*-1900.

Pada penelitian ini *mobile robot* yang dibuat merupakan suatu *prototype* yang akan digunakan dalam lingkungan *indoor*. Biasanya dalam lingkungan *indoor* untuk suatu *mobile robot* memiliki jalur khusus yang digunakan. Jalur ini merupakan jalur yang terdiri dari dua garis kanan dan kiri yang memiliki ketebalan tertentu. Dengan informasi dari jalur *mobile robot* yang akan ditangkap citra nya oleh sensor *webcam*, maka penelitian ini memanfaatkan suatu metode *edge detection* untuk mendapatkan informasi tepi garis yang didapatkan dari garis jalur kanan ataupun kiri yang digunakan pada *mobile robot*. Informasi dari *edge detection* yang digunakan supaya *mobile robot* dapat dikendalikan secara otomatis mengikuti jalur yang digunakan sebagai lintasan pergerakan *mobile robot*.Selain dapat dikendalikan secara otomatis arah pergerakannya, *mobile robot* pada penelitian ini juga dapat dikendalikan secara manual menggunakan *joystick.*

2. TINJAUAN PUSTAKA

2.1 Pengolahan Citra

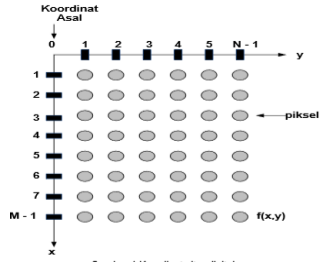
Pengolahan citra merupakan suatu proses memperbaiki kualitas citra sehingga mudah untuk diinterpretasi oleh manusia ataupun *computer*[14]. Pengolahan citra merupakan suatu proses dari *computer vision*. *Computer vision* merupakan proses untuk memulihkan beberapa informasi yang tidak memadai sehingga informasi didapat secara spesifik untuk menentukan solusi[10]. Pada gambar 2.1 merupakan proses umum dari *Computer vision.* Dimulai dari proses awal (*preprocessing*) biasanya berupa proses untuk menghilangkan *noise* atau untuk mempertajam citra. Proses selanjutnya adalah proses *Lowest Level Feature* mengenai deteksi yang dilakukan pada citra, setelah itu proses *Intermediate Level Feature*, lalu proses terakhir adalah proses *High Level Scene* untuk mengimplementasikan hasil citra yang diolah.



Gambar 2. 1 Proses Umum *Computer Vision*[14]

Pada pencitraan dan representasi gambar ini terdapat perangkat pencitraan. Banyak perbedaan dari perangkat dalam mengolah gambar digital. Salah satu diantaranya adalah kamera video. Kamera video menciptakan citra untuk rekam jejak komposisi manusia dari gambar dengan 30 *frame per second*, memungkinkan representasi gerakan objek dari waktu ke waktu yang disajikan dalam gambar atau *frame* tunggal. Sementara persepsi manusia untuk mendapatkan tangkapan gambar pada video secara halus adalah 60 *frame per second*[5].

Hasil sampling dan kuantisasi dari sebuah citra merupakan bilangan *real* yang membentuk matriks M baris dan N kolom[2]. Secara umum koordinat yang digunakan dalam citra sesuai pada gambar 2.2. Pada perpotongan matriks MxN tersebut disebutkan dengan piksel sebagai satuan dari sebuah citra. Piksel tersebut dapat berupa koordinat maupun intensitas atau warna. Nilai koordinat piksel (x,y) merupakan suatu fungsi f(x,y) pada titik piksel tersebut.



Gambar 2. 2 Koordinat Citra Digital [4]

2.2 *Edge Detection*

Edge detection atau deteksi tepi merupakan salah satu algoritma yang dapat digunakan dalam pengolahan citra[14]. Kata *edge* ataupun *edge point* merupakan poin dalam gambar di mana kecerahan berubah sangat tajam[15]. *Edge detection* telah digunakan oleh pengenalan objek, pelacakan target, segmentasi, kompresi data, dan juga membantu untuk pencocokan yang baik, seperti rekonstruksi gambar dan sebagainya[8].

Untuk melakukan *Edge detection* ini, terdapat beberapa operator yang dapat digunakan dengan

teknik konvolusi sebagai *filter* agar suatu tepi dapat dideteksi. Operator-operator tersebut diantaranya yaitu operator *gradient* sebagai operator turunan pertama dan operator turunan kedua. Operator turunan pertama diantaranya *Sobel*, *Prewitt*, dan *Roberts*. Sedangkan operator turunan kedua yaitu *Gaussian Filter* dan *Laplacian of Gaussian* [13].

2.3 Operator *Sobel*

Operator *Sobel* merupakan matriks dengan ukuran 3x3 piksel yang terdiri dari Gx dan Gy. Operator ini dirancang untuk mendapatkan respon secara maksimal terhadap tepi objek secara horizontal maupun vertikal[13]. Kelebihan dari operator *Sobel* ini ada pada kemampuan untuk mengurangi *noise* sebelum melakukan perhitungan *edge detection*[9].

Berikut merupakan persamaan (2-1) dan (2-2) sebagai persamaan matriks *Sobel*[5].

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad (2\text{-}1)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad (2\text{-}2)$$

Hasil operasi untuk masing-masing Gx dan Gy pada persamaan (2-3) dapat menentukan nilai magnitudonya.

$$G = \sqrt{{G_x}^2 + {G_y}^2} \quad (2\text{-}3)$$

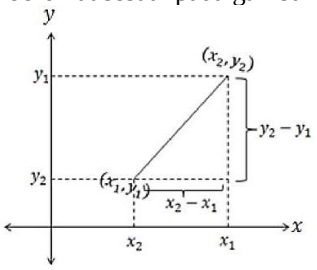
Untuk menghasilkan *gradien* dengan arah vertikal maupun horisontal maka dapat direpresentasikan dengan persamaan (2-4) dan (2-5) [13].

$$G_x = [f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1)] - [f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1)] \quad (2\text{-}4)$$

$$G_y = [f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)] - [f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)] \quad (2\text{-}5)$$

2.4 Gradien Garis Lurus Melalui dua Titik

Suatu nilai kemiringan dalam matematika dinamakan dengan "*gradien*". *Gradien* merupakan koefisien arah dari garis lurus yang dilambangkan dengan huruf m. Dalam koordinat kartesius, *gradien* diartikan sebagai nilai kemiringan yang membandingkan antara sumbu y (ordinat) dan sumbu x(absis) [1]. Koordinat sesuai pada gambar 2.3.



Gambar 2. 3 *Gradien* Garis 2 titik[1]

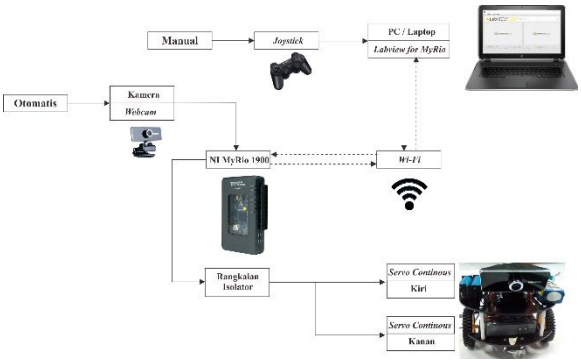
Berdasarkan koordinat kartesius pada gambar 2.3, maka didapatkan rumus *gradien* pada persamaan (2-6).

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_{2-x_1}} \quad (2\text{-}6)$$

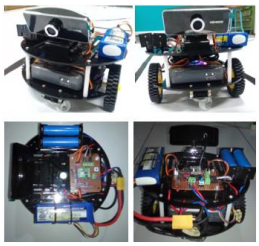
3. METODOLOGI PENELITIAN

3.1 Gambaran Umum Sistem

Sesuai yang diperlihatkan pada gambar 3.1, konsep dari *mobile robot* ini terdiri dari *Joystick* sebagai kendali manual, satu sensor kamera sebagai kendali otomatis, sebuah *controller* NI *My Rio*-1900 dan dua buah penggerak roda kanan maupun kiri. *Joystick* yang digunakan terhubung ke *personal computer* atau PC untuk mengatur kendali *mobile robot* secara manual, dan PC menampilkan tampilan gambar yang diambil kamera tanpa pengolahan citra, hanya sebagai tampilan pendukung apa yang sedang terdeteksi oleh *webcam*. *Webcam* terhubung pada *controller* NI *My Rio*-1900 karena pengolahan citra diolah melalui *controller* dan PC hanya sebagai antarmuka yang digunakan untuk melihat hasil olahan yang dilakukan. Sinyal yang dikirimkan dari *controller* menuju aktuator dilakukan melalui isolator yang terdiri dari rangkain *optocoupler* sehingga sebagai pengaman antara hubungan dari *controller* dan aktuator.



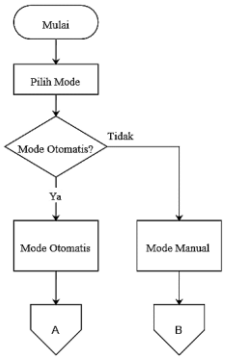
Gambar 3. 1 Gambaran Umum Sistem secara Keseluruhan



Gambar 3.2 Tampilan *Hardware Mobile Robot* yang digunakan

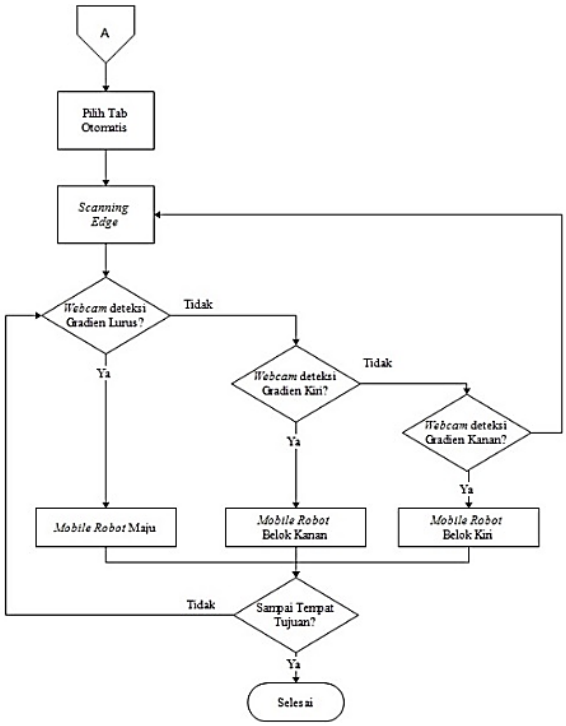
3.2 Alur Kerja Sistem

Berdasarkan gambaran umum sistem, terdapat mode otomatis dan mode manual. Pada tampilan antarmuka, mode dapat dipilih. Jika mode otomatis yang dipilih maka proses berlangsung pada bagian A. Namun, saat mode otomatis tidak dipilih, maka proses akan berlangsung secara manual pada bagian B.



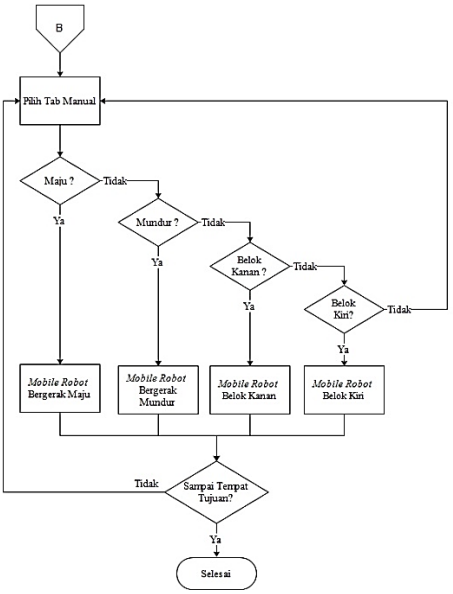
Gambar 3. 3 Alur Cara Kerja sistem Pilihan Otomatis atau Manual

Ketika mode otomatis sesuai gambar 3.4 pada proses A terpilih, tampilan *tab* otomatis dapat digunakan. Untuk sensor kamera atau *webcam* akan terus melakukan *scanning* sampai objek-objek sekitarnya terdeteksi. Setiap objek yang terdeteksi dapat dibaca oleh pengolahan citra, maka program akan dilanjutkan untuk eksekusi pergerakan robot.



Gambar 3. 4 Alur Cara Kerja sistem Pilihan Otomatis

Saat mode otomatis tidak dipilih, maka proses B yang akan berlangsung pada mode manual. Untuk melihat tampilan yang ada pada robot, *webcam* dapat digunakan untuk monitoring pergerakan robot. Pada mode manual ini digunakan *Joystick* sebagai pemberi inputan perintah untuk *mobile robot* bergerak. Terdapat perintah maju, mundur, belok kanan, maupun belok kiri sesuai pada alur kerja gambar 3.5.

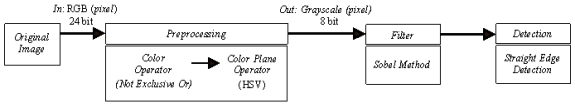


Gambar 3. 5 Alur Cara Kerja sistem Pilihan Manual

3.3 Alur Kerja Pengolahan Citra

Untuk pengambilan data *edge detection* pada *LabVIEW*, digunakan *library vi* khusus. *Library* ini adalah *Vision Assistant*. Pada VI ini dapat diatur algoritma yang diinginkan dalam pemrosesan citra yang dilakukan. *Library* khusus ini dapat ditemukan pada *vision and motion* > *vision express* > *vision assistant.*

Pada tabel 3.4 ini diperlihatkan pemrosesan citra yang digunakan pada tugas akhir ini. Dimulai dari gambar original yang didapatkan berupa RGB. Pada *red*, *green*, *blue* ini memiliki nilai masing-masing pada kisaran piksel 0-255 atau sebesar 24 bit memori. Memori pada citra warna RGB ini terlalu besar untuk diolah sehingga dari nilai piksel ini, citra diolah menjadi bentuk citra *grayscale* dimana nilai citra hanya menjadi 8 bit memori saja yaitu berkisar 0-255 piksel saja[12]. Tahapannya pada pemrosesan citra pada *vision assistant* ini sesuai pada gambar 3.6 yaitu dari citra warna atau gambar original yang didapatkan, lalu untuk tahap *preprocessing* tersebut digunakan *color operator* dan *color plane extraction*, selanjutnya digunakan *filter* dengan metode operator *Sobel*, dan yang terakhir untuk *edge detection* digunakan *straight edge* karena tepi yang digunakan berupa jalur dari sebuah garis lurus.

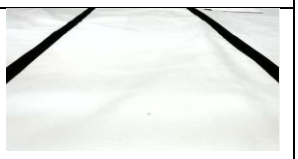
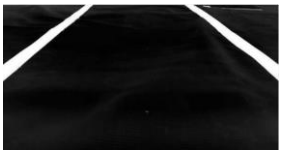
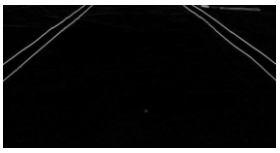
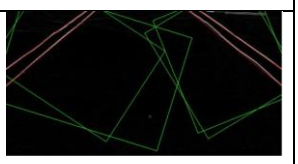
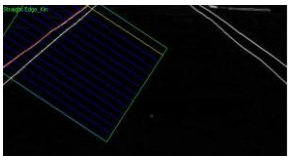
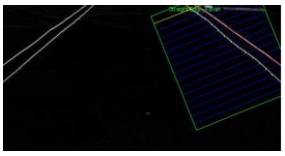


Gambar 3. 6 Blok Diagram Proses Pengolahan Citra

Proses tersebut dilakukan pada *software* pengolahan citra NI *Vision Assistant* dengan menggunakan *tool* yang tersedia pada *software* ini. Proses ini diperlihatkan pada gambar 3.7.

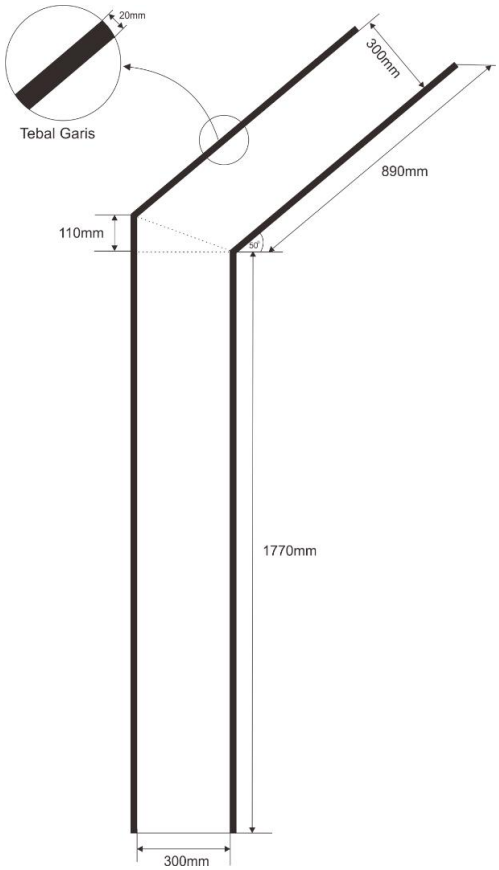
Gambar 3. 7 Tahapan Proses Pengolahan Citra pada *Vision Assistant*

Tabel 3. 1 Tahapan Proses Pengolahan Citra pada *Vision Assistant*

No	Keterangan	Gambar
1	*Original Image* (merupakan gambar asli yang digunakan)	
2	*Color Operator* (untuk mempertajam citra yang didapatkan dengan membalikan warna citra menggunakan logika *Not Exclusive Or*)	
3	*Color Plane Extraction* (mengubah nilai citra warna menjadi HSV (*Hue Saturation Value)* agar dapat diolah menggunakan filter)	
4	*Filter* (untuk mendapatkan nilai *edge* dengan menggunakan operator *Sobel*)	
5	*Straight Edge* (untuk pendeteksian tepi garis, tepi garis yang dideteksi yaitu bagian dalam dan luar garis untuk memperkuat nilai posisi koordinat piksel dari yang didapatkan pada pendeteksian tepi baris tersebut. Tepi garis yang terdeteksi ditandai dengan warna merah, dan pada sistem pemrograman yang digunakan garis tepi bagian dalam sebagai data untuk *range* jalur yang dilalui, namun jika tepi dalam tidak terdeteksi, maka tepi luar yang terdeteksi yang akan digunakan)	  

3.4 Jalur *Mobile Robot*

Jalur *Mobile robot* yang yang digunakan untuk dilalui sesuai pada gambar 3.8. Jalur yang digunakan berupa jalur lurus dan terdapat belokan ke kanan dengan sudut sebesar 50^0 dari lebar yang dipertahankan dengan nilai 300mm. Nilai sudut belokan tidak terlalu diperhatikan selama sudut lebih dari 45^0 dan lebar jalur tetap dengan nilai 300mm. Ketebalan garis jalur pada jalur yang digunakan adalah 20mm. Namun jika *mobile robot* mendeteksi garis jalur yang lebih tebal pun tidak menjadi masalah selama posisi tepi dalam garis jalur tetap sama, karena sistem yang digunakan mengutamakan nilai tepi bagian dalam untuk menjadikan nilai *range* yang digunakan untuk pergerakan arah *mobile robot* .

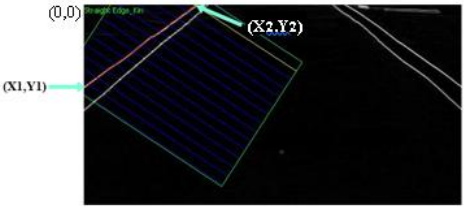


Gambar 3. 8 Jalur yang digunakan

3.5 Perhitungan Algoritma Perintah Kendali Otomatis pada *Mobile Robot*

Nilai yang didapatkan oleh *straight edge detection* berupa suatu koordinat piksel. Didapatkan koordinat untuk piksel jalur sebelah kanan maupun jalur sebelah kiri. Untuk mengubah hasil dari koordinat piksel tersebut menjadi algoritma pemrograman dalam pengolahan untuk memberikan perintah pada pergerakan aktuator yang digunakan, maka dalam tugas akhir ini digunakan metode perhitungan *gradien* pada persamaan garis lurus melalui dua titik.

Sesuai dengan gambar 2.3 mengenai koordinat kartesius pada *gradien* garis lurus melalui dua titik, bahwa dari jalur tepi garis yang dideteksi oleh *webcam* memberikan suatu nilai koordinat (x_1, y_1) dan (x_2, y_2). Namun nilai origin citra berada berkebalikan dengan nilai origin koordinat kartesius. Hal ini dapat dilihat pada gambar 3.9.



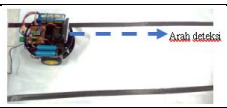
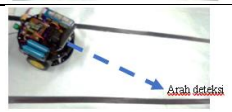
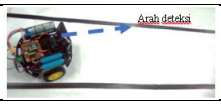
Gambar 3.9 *Edge Detection* pada Jalur Robot

Untuk mendapatkan algoritma perintah yang akan dibuat, maka ditentukan jalur origin terlebih dahulu sebagai pembanding dan *range* nilai *gradien* yang digunakan untuk posisi arah pergerakan *mobile robot*.

Sehingga robot mengetahui apa yang akan dilakukan jika robot ada dalam keadaan *range gradien* tersebut.

Tabel 3.2 berikut merupakan posisi yang digunakan sebagai pengambilan data *range* posisi origin robot untuk perintah pada jalur lurus yang dilalui robot. Pengambilan data tersebut hanya sebagian *sample* dari *range* yang digunakan. Data *range* yang digunakan sebagai algoritma program yang digunakan berdasarkan beberapa kali percobaan, sehingga nilai yang digunakan sebagai syarat untuk menggerakan arah *mobile robot* bergerak tidak hanya dari nilai *sample* ini saja.

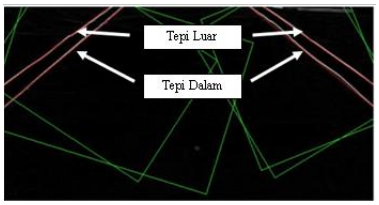
Tabel 3. 2 Pengambilan Data Posisi *Mobile robot* pada Jalur Lurus

No.	Posisi Pengambilan Data *Range*	Gambar
1	*Mobile robot* berada ditengah jalur.	
2	*Mobile robot* berada sisi sebelah kanan batas jalur.	
3	*Mobile robot* berada sisi sebelah kiri batas jalur.	
4	*Mobile robot* dalam posisi serong kanan dari jalur.	
5	*Mobile robot* dalam posisi serong kiri dari jalur.	

Posisi data *range* yang memberikan nilai koordinat (x_1, y_1) dan (x_2, y_2) diolah menjadi data *gradien* pada tepi garis jalur kanan maupun jalur kiri dengan perhitungan menggunakan persamaan (2-6).

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_{2-x_1}} \qquad (2\text{-}6)$$

Setelah nilai dari masing-masing sumbu koordinat didapatkan, lalu dimasukan kedalam rumus persamaan (2-6) untuk mendapatkan nilai *gradien*nya. Pada tabel berikut, hasil pengambilan data nilai *gradien* yang didapatkan dari posisi origin yang digunakan sebagai *range gradien*. Pada jalur kanan maupun jalur kiri, tepi yang digunakan untuk mengambil data adalah tepi luar dan tepi dalam. Sesuai yang diperlihatkan pada gambar 3.10, data yang didapatkan akan dijadikan sebagai nilai *range* untuk diteruskan menjadi perintah pada aktuator. Nilai yang digunakan pada algoritma program diprioritaskan nilai *gradien* dari tepi dalam yang terdeteksi, namun bila tepi dalam tidak terdeteksi maka nilai *gradien* yang akan digunakan sebagai batas *range* perintah merupakan nilai dari tepi luar.



Gambar 3. 10 Tepi Dalam dan Tepi Luar

3.6 Perancangan Penggerak *Mobile Robot*

Nilai *gradien* yang telah diberikan masing-masing *range* tersebut dikelompokan untuk memberikan perintah berupa memberikan nilai untuk mengatur nilai PWM pada aktuator. Tabel 3.8 berikut ini merupakan perintah yang diberikan dari hasil klasifikasi *range* yang didapatkan pada jalur lurus.

Tabel 3. 8 Klasifikasi *Range Origin Position*

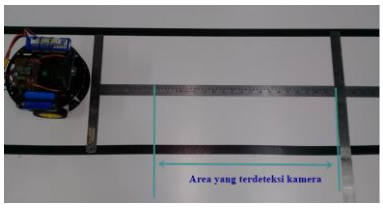
No.	Posisi Robot	Perintah	Nilai PWM	
			Roda Kanan	Roda Kiri
1	*Mobile robot* berada ditengah jalur.	Robot maju	-60	+60
2	*Mobile robot* berada sisi sebelah kanan batas jalur.	Robot berputar ke kiri	-50	-50
3	*Mobile robot* berada sisi sebelah kiri batas jalur.	Robot berputar ke kanan	+50	+50
4	*Mobile robot* dalam posisi serong kanan dari jalur.	Robot berputar ke kiri	-30	-30
5	*Mobile robot* dalam posisi serong kiri dari jalur.	Robot berputar ke kanan	+30	+30

Pada Implementasinya *mobile robot* tidak dapat melaju lurus karena rpm (*rotation per minute*) pada aktuator kanan dan kiri tidak sama, maka nilai PWM untuk roda kanan dan kiri saat *mobile robot* bergerak maju disesuaikan dengan menyamakan nilai rpm pada aktuator roda kanan maupun roda kiri dengan dengan menggunakan tachometer. Sehingga nilai PWM yang diberikan yaitu -60 untuk roda kanan dan +73 untuk roda kiri. Nilai nilai PWM pada program otomatis ini menjadi nilai default pada program yang digunakan. Menyesuaikan dengan kemampuan citra untuk mengolah deteksi tepi garis. Karena jalur yang digunakan hanya memiliki lebar 300mm maka kecepatan *mobile robot* untuk bergerak maju ataupun belok juga diperlambat agar saat perintah hasil olahan *gradien* menjadi pergerakan ke aktuator *mobile robot* tersebut tidak terlalu kencang untuk melaju serta agar *mobile robot* juga tidak mudah melenceng dari jalur.

4. HASIL DAN PEMBAHASAN

4.1 Jarak Deteksi Kamera *Webcam*

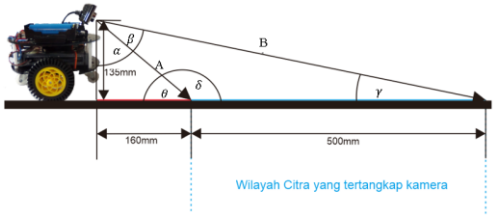
Kemampuan kamera untuk mendeteksi objek sekitar dengan resolusi yang digunakan kamera sebesar 544x288 0.81x32-bit RGB dengan 30 FPS maksimal dan cahaya ruangan yang digunakan di antara 50 hingga 100 lux. Kamera yang ditempatkan secara tetap pada mobile robot memiliki jarak deteksi sejauh 500mm, dengan tinggi lensa kamera pada mobile robot adalah 135mm. Gambar 4.1 dan 4.2 berikut memperlihatkan wilayah citra yang ditangkap kamera webcam.



Gambar 4. 1 Foto Keadaan Area yang akan dideteksi

Gambar 4. 2 Pengambilan Gambar Hasil Area yang terdeteksi kamera

Gambar 4.3 berikut menunjukan ukuran yang didapatkan dari hasil pengukuran panjang dan lebar wilayah citra yang ditangkap kamera *webcam*. Dari informasi yang diketahui tersebut, sudut-sudut dapat dihitung dengan persamaan trigonometri pada segitiga menggunakan aturan sinus dan segitiga siku-siku.



Gambar 4. 3 Jarak Deteksi Kamera

Dari hasil pengukuran panjang dan lebar citra yang didapatkan, nilai sudut $\alpha, \beta, \theta, \delta,$ dan γ serta nilai panjang A dan B pada gambar 4.3 dapat dihitung dengan cara berikut.

Mencari nilai panjang A.

$$A = \sqrt{135^2 + 160^2} \quad (4\text{-}1)$$

$$A = 20{,}93 \text{ mm}$$

Mencari nilai sudut θ.

$$\sin\theta = \frac{\sin 90^0 \text{ x } 13{,}5}{20{,}93} \quad (4\text{-}2)$$

$$\theta = sin^{-1}\left(\frac{1\,.\,13{,}5}{20{,}93}\right)$$

$$\theta = 40{,}166^0$$

Mencari nilai sudut α.

$$\alpha = 180^0 - 90^0 - \theta \quad (4\text{-}3)$$

$$\alpha = 180^0 - 90^0 - 40{,}166^0$$

$$\alpha = 49{,}83^0$$

Mencari nilai sudut δ.

$$\delta = 180^0 - \theta \quad (4\text{-}4)$$

$$\delta = 180^0 - 40{,}166^0$$

$$\delta = 139{,}83^0$$

Mencari nilai panjang B.

$$B = \sqrt{135^2 + 660^2} \quad (4\text{-}5)$$

$$B = 67{,}366\text{mm}$$

Mencari nilai sudut γ.

$$\frac{B}{\sin 90^0} = \frac{13{,}5}{\sin\gamma} \quad (4\text{-}6)$$

$$\frac{67{,}366}{\sin 90^0} = \frac{13{,}5}{\sin\gamma}$$

$$\sin\gamma = \frac{13{,}5}{67{,}366}$$

$$\gamma = sin^{-1}\left(\frac{13{,}5}{67{,}366}\right)$$

$$\gamma = 11{,}56^0$$

Mencari nilai sudut β.

$$\beta = 180^0 - \delta - \gamma \quad (4\text{-}7)$$

$$\beta = 180^0 - \delta - \gamma$$

$$\beta = 180^0 - 139{,}83^0 - 11{,}56^0$$

$$\beta = 28{,}6^0$$

Dapat diambil kesimpulan bahwa sudut yang didapatkan pada wilayah yang didapatkan oleh citra ada pada sudut $\beta, \gamma,$ dan δ. Nilai sudut yang didapatkan adalah $28{,}6^0$ untuk sudut β, $11{,}56^0$ untuk sudut γ, dan $139{,}83^0$ untuk sudut δ.

4.2 Pengujian Mode Otomatis

4.2.1 Waktu Pemrosesan *Edge Detection* pada Sistem Otomatis

Pengujian ini dilakukan untuk mengetahui waktu pemrosesan yang dilakukan untuk sistem untuk mengubah input suatu tepi menjadi perintah pergerakan arah *mobile robot* dalam 1 kali proses *edge detection* dalam program *looping* yang digunakan. Pengujian dilakukan dengan dua keadaan. Berikut merupakan keadaan pada pengujian ini dan hasil waktu pemrosesan disajikan pada tabel 4.3.

Keadaan 1: Saat tepi garis tak terdeteksi sama sekali oleh kamera *webcam*

Keadaan 2: Saat tepi garis terdeteksi

Tabel 4. 1 Waktu Pemrosesan *Edge Detection* pada Sistem Otomatis

Percobaan Ke-	Waktu Pemrosesan dalam 1 proses *edge detection*	
	Keadaan 1 (ms)	Keadaan 2 (ms)
1	78,4	68,6
2	82,5	71
3	82,2	71,5
4	78,7	71,7
5	76,3	71,2
6	77,2	69,5
7	77,4	71,5
8	81,3	70,8
9	80,5	70,6
10	81,7	69,3

Rata- Rata	Keadaan 1 (ms)	Keadaan 2 (ms)
	79,62	**70,57**

Hasil dari pengujian dalam satu kali *looping* yang dilakukan oleh sistem, saat kamera *webcam* tidak mendeteksi sama sekali garis tepi waktu proses yang dibutuhkan memiliki rata 79,62ms, sedangkan waktu pemrosesan saat sistem dapat mendeteksi tepi garis

dibutuhkan waktu dengan rata-rata 70,57ms dalam satu kali *looping* sistem dilakukan.

Sistem yang digunakan pada *LabVIEW* ini dapat menjadi dua bagian yang terpisah saat dijalankan, sehingga sistem tidak membebani proses yang dilakukan pada NI *My Rio*-1900. NI *My Rio*-1900 hanya menjalankan proses pengolahannya citranya saja dan pemberian sinyal pada aktuator. Sementara proses lainnya seperti proses manual *joystick* dapat dijalankan di *Personal Computer*.

4.2.2 Kecepatan Eksekusi *Mobile robot* secara Otomatis

Pengujian selanjutnya dilakukan menggunakan media *field* robot berbahan *komatex board* saja sebagai kondisi dengan *noise* yang sangat sedikit dibandingkan media *banner* sebagai *field*. Pengujian dilakukan menggunakan jalur lurus dengan belokan ke kanan dan jalur lurus dengan belokan ke kiri untuk mengetahui waktu tempuh yang dimiliki robot untuk gerakan berbelok. Sesuai dengan jalur gambar 3.8 untuk pergerakan *mobile robot* sepanjang 2,66 meter melewati belokan ke kanan dengan sudut 50^0, *mobile robot* mampu melewati jalur lurus lalu berbelok ke kanan dengan waktu pada rata-rata 18,17 detik. Sehingga kecepatan rata-rata yang didapatkan adalah 0,146meter/detik dengan nilai *error* rata-rata 13,22%. Data percobaan disajikan pada tabel 4.6. Nilai Simpangan atau *error* (%) yang disajikan pada tabel 4.6 dan 4.7 didapatkan dari rumus (4-8).

Tabel 4. 2 Kemampuan Waktu tempuh *Mobile robot* pada Jalur Lurus + Belok Kanan

Percobaan Ke-	Waktu (detik)	*Error* (%)
1	19,10	5,12
2	29,13	60,32
3	15,40	15,24
4	16,56	8,86
5	18,62	2,48
6	15,25	16,07
7	16,20	10,84
8	20,24	11,39
9	15,38	15,35
10	17,07	6,05
11	17,40	4,24
12	17,68	2,7

Sesuai dengan gambar 3.14 untuk pergerakan *mobile robot* sepanjang 2,66 meter melewati belokan ke kiri dengan sudut belokan dengan nilai 50^0, *mobile robot* mampu melewati jalur lurus lalu berbelok ke kiri dengan waktu pada rata-rata 18,84 detik. Sehingga kecepatan rata-ratanya senilai 0,141meter/detik dengan *error* rata-rata yang sebesar 8,36%. Data percobaan disajikan pada tabel 4.7.

Tabel 4. 3 Kemampuan Waktu tempuh *Mobile robot* pada Jalur Lurus + Belok Kiri

Percobaan Ke-	Waktu (detik)	*Error* (%)
1	15,73	16,51
2	18,91	0,37
3	25,31	34,34
4	18,03	4,3
5	16,73	11,2
6	20,5	8,81
7	19,86	5,41
8	18,99	0,8
9	17,49	7,17
10	18	4,46
11	17,58	6,69
12	18,9	0,32

Hasil kedua percobaan untuk berjalan lurus lalu berbelok ke kanan maupun kiri memiliki selisih waktu tempuh senilai 0,67 detik. Dengan selisih yang tidak begitu jauh ini *mobile robot* mampu belok kanan dan kiri dengan nilai *error* rata-rata 13,22% pada saat belok kanan dan 8,36% pada saat belok kiri. Rata-rata *error* ini didapatkan dari masing-masing perbandingan *error* antara nilai rata-rata keseluruhan data dengan nilai dari masing-masing percobaan yang dilakukan. Sehingga, tingkat keberhasilan untuk belokan dengan sudut 50^0 pada belok kanan sebesar 86,78% dan untuk belok kiri sebesar 91,64% dari 12 data percobaan.

4.2.3 Hasil Pengujian Proses Edge Detection menjadi Perintah Eksekusi pada *Mobile robot*

Pengujian ini dilakukan untuk mengetahui berapa kali *mobile robot* merubah arah posisi pergerakan dengan mengubah nilai PWM dari hasil *edge* yang terdeteksi pada satu kali siklus pada jalur yang digunakan sesuai dengan gambar 3.8. Pengujian dilakukan menggunakan jalur lurus dengan belokan ke kanan dan jalur lurus dengan belokan ke kiri. Data pengujian dapat dilihat pada tabel 4.8 untuk jalur lurus dengan belokan ke kanan, dan tabel 4.9 untuk jalur lurus dengan belokan ke kiri.

Nilai PWM pada motor *servo continous* berubah karena hasil dari olahan *gradien* yang didapatkan dari *edge detection*. Saat semakin banyak nilai PWM berubah (posisi arah pergerakan robot berubah), berarti semakin banyak robot melakukan *scanning* untuk mendeteksi edge karena perubahan pergerakan *mobile robot*.

Tabel 4. 4 Hasil Uji Pergerakan Otomatis *Mobile robot* pada Jalur Belok Kanan

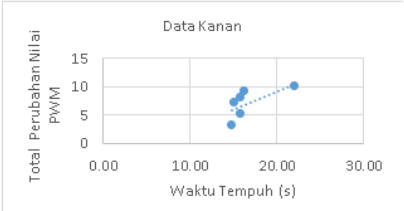
Percobaan Ke-	Waktu tempuh (detik)	Total Perubahan Nilai PWM
1	14,86	3
2	16,33	9
3	15,21	7
4	22,18	10
5	15,89	8
6	15,87	5

Tabel 4. 5 Hasil Uji Pergerakan Otomatis *Mobile robot* pada Jalur Belok Kiri

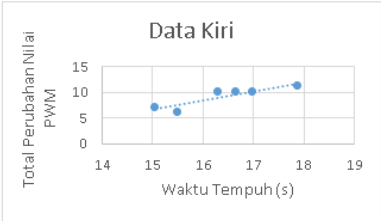
Percobaan Ke-	Waktu tempuh (detik)	Total Perubahan Nilai PWM
1	15,51	6
2	16,99	10
3	17,89	11

4	15,07	7
5	16,32	10
6	16,66	10

Dari kedua data pada tabel 4.8 dan 4.9 dapat diambil kesimpulan bahwa semakin sedikit waktu yang ditempuh untuk satu jalur yang dilalui, maka semakin sedikit Jumlah Perubahan nilai PWM untuk arah pergerakan *mobile robot*. Hal ini dapat dibuktikan dengan histogram yang disajikan pada gambar 4.4 dan 4.5.



Gambar 4. 4 Histogram Data Perubahan Posisi Jalur Belok Kanan



Gambar 4. 5 Histogram Data Perubahan Posisi Jalur Belok Kiri

4.3 Pengujian *Capture in Rate* (FPS)

Berdasarkan percobaan yang dilakukan pada *mobile robot* saat mendeteksi tepi garis, dilakukan pengambilan data *frame per second* saat *mobile robot* mendeteksi dalam keadaan diam dan dalam keadaan bergerak. Saat dalam keadaan diam rata-rata FPS terendah yang didapatkan adalah 29,77 FPS, dan tertinggi adalah 29,79. Sementara saat *mobile robot* mendeteksi dalam keadaan bergerak nilai FPS yang paling rendah adalah 29,476 FPS dan nilai tertinggi adalah 29,484 FPS. Data tersebut diperlihatkan pada tabel 4.9.

Tabel 4. 6 Rata-rata Data *Frame Per Second* Percobaan *Mobile robot*

Percobaan Ke-	Robot Diam (FPS)	Robot Bergerak (FPS)
1	29,79	29,483
2	29,77	29,477
3	29,78	29,484
4	29,77	29,476
5	29,78	29,478

5. KESIMPULAN

Berdasarkan pengujian yang dilakukan terhadap kendali *mobile robot* menggunakan pengolahan citra yang dibangun dan selama proses penyusunan karya tulis tugas akhir ini, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Pada pengujian otomatis, pengujian waktu pemrosesan didapatkan dalam satu kali *looping* yang dilakukan oleh sistem. Saat kamera *webcam* tidak mendeteksi sama sekali garis tepi, waktu proses yang dibutuhkan memiliki rata 79,62ms, sedangkan saat sistem dapat mendeteksi tepi garis dibutuhkan waktu dengan rata-rata 70,57ms dalam satu kali *looping* sistem dilakukan.
2. Pengujian lainnya pada mode otomatis, dilakukan pengujian untuk melihat perubahan yang terjadi pada nilai PWM motor *servo continous*. Perubahan nilai PWM pada motor *servo continous* terjadi karena hasil dari olahan *gradien* yang didapatkan dari *edge detection*. Saat semakin banyak posisi PWM berubah (posisi robot berubah), berarti semakin banyak robot melakukan *scanning* untuk mendeteksi *edge* karena perubahan pergerakan *mobile robot*.
3. Pengujian lainnya pada mode otomatis, dengan pergerakan *mobile robot* sesuai jalur sepanjang 2,66 meter dengan jalur lurus lalu berbelok ke kanan dengan nilai sudut 50^0, memiliki rata-rata waktu 18,17 detik dengan kecepatan rata-rata yang didapatkan adalah 0,146meter/detik dan *error* rata-rata yang terjadi memiliki nilai 13,22%. Sementara untuk pergerakan mobile robot melewati jalur lurus lalu berbelok ke kiri dengan nilai sudut 50^0, memiliki rata-rata waktu 18,84 detik. Selisih antara belokan ke kanan maupun belokan ke kiri senilai 0.67 detik dengan kecepatan rata-ratanya senilai 0,141meter/detik dan *error* rata-rata yang sebesar 8,36%.
4. Saat *mobile robot* dalam keadaan diam rata-rata FPS terendah yang didapatkan adalah 29,77 FPS, dan tertinggi adalah 29,79. Sementara saat *mobile robot* mendeteksi dalam keadaan bergerak nilai FPS yang paling rendah adalah 29,476 FPS dan nilai tertinggi adalah 29,484 FPS.

6. SARAN

Berdasarkan pengujian yang dilakukan terhadap kendali *mobile robot* menggunakan pengolahan citra yang dibangun dan selama proses penyusunan karya tulis tugas akhir ini, dapat ditarik saran sebagai berikut:

1. Pada penelitian selanjutnya diharapkan pengolahan citra untuk pergerakan *mobile robot* dengan penambahan metode lainnya yang ada pada NI *Vision Assistant* selain metode *edge detection*, sebagai parameter lain agar pemrograman tidak hanya mengandalkan nilai *gradien* saja karena saat *mobile robot* berada di posisi sangat melenceng dari jalur, perintah yang ada di program tidak sesuai dengan yang seharusnya.
2. Pada penelitian ini digunakan operator *Sobel* untuk mendapatkan tepi garis dari jalur karena berdasarkan referensi yang didapatkan, operator *Sobel* merupakan operator yang lebih tajam dibandingkan operator lain untuk mendeteksi tepi garis. Sementara dengan resolusi citra yang digunakan sebesar 544x288 0.81x32-bit RGB yang digunakan dalam *template kernel Sobel* 3x3, menyebabkan ada beberapa data citra yang hilang

karena dengan *kernel* 3x3 tidak dapat menampung semua data citra yang masuk. Sehingga disarankan untuk penelitian selanjutnya *kernel* yang digunakan dapat lebih besar atau resolusi citra diperkecil namun dalam ketentuan *edge* masih terdeteksi.

3. Media jalur yang digunakan *mobile robot* pada penelitian ini dengan *noise* yang lebih sedikit adalah media berupa *komatex board* berwarna putih dengan garis jalur berwarna hitam. Namun dalam penggunaannya, saat cahaya matahari masuk ke dalam ruangan, *komatex board* tidak dapat menyerap cahaya yang datang, sehingga pengambilan data citra oleh sensor kamera *webcam* terganggu karena pantulan cahaya dari *komatex board* yang digunakan. Disarankan untuk penelitian selanjutnya dapat menggunakan media *field mobile robot* yang memiliki sedikit *noise* dan dapat menyerap cahaya yang ada pada lingkungan yang digunakan.

DAFTAR PUSTAKA

[1] M. Rifa'i, Matematika dasar (Pra kalkulus), Ponorogo: Uwais Inspirasi Indonesia, 2019.

[2] P. N. Andono, T. Sutojo and Muljono, Pengolahan Citra Digital, Yogyakarta: CV Andi Offset, 2017.

[3] M. Z. A. Martin, "Desain dan implementasi multitasking sensor dengan raspberry PI 3 model B pada mobile robot line follower," (Karya Tulis). Diploma IV. Polman Bandung, Bandung, 2016

[4] D. Indra, "Pendeteksian tepi objek menggunakan metode *gradien*," *Jurnal Ilmiah ILKOM Vol.8 No.2 ISSN: 2087-1716* , pp. 69-75, 2016.

[5] B. Crnokic, S. Rezic dan S. Pehar, “Comparison of edge detection methods for obstacles detection in a mobile robot environment,” dalam *DOI: 10.2507/27th.daaam.proceedings.035*, Austria, 2016.

[6] K. D. Seryanto and dkk., "Pengendalian mobile robot vision menggunakan webcam pada objek arah panah berbasis raspberry PI," *Jurnal Arus Elektro Indonesia (JAEI) - Universitas Jember*, pp. 27-32, 2015.

[7] M. Karim dan dkk, “Robot line follower berbasis raspberry PI dengan sensor kamera.,” STMIK/AMIK MDP (Skripsi), Palembang, 2014.

[8] S. Islam and M. Ahmed, "A study on edge detection technique for natural image segmentation," *International Journal of Innovative Technology and Exploring Engineering (IJITEE). ISSN:2278-3075. Vol 2(3),* pp. 80-83, 2013.

[9] M. Yunus, "Perbandingan metode-metode edge detection untuk proses segmentasi citra digital," *Jurnal Teknologi Informasi Vol.3 No.2*, pp. 146-160, 2012.

[10] R. Szeliski, Computer vision, algorithms, an applications, London: Springer., 2011.

[11] R. Siegwart, I. R. Nourbakhsh and D. Scaramuzza, Introduction to autonomous mobile robots, London: Cambridge, Massachusetts., 2011.

[12] R. Muthukrishnan and M. Radha, "Edge detection techniques for image segmentation," *International Journal of Computer Science & Information Technology (IJCSIT)* , vol. 3, no. 6, pp. 259-267, 2011.

[13] M. Purnomo and A. Muntasa, Konsep Pengolahan Citra Digital dan Ekstraksi Fitur, Yogyakarta: Graha Ilmu, 2010.

[14] Universitas Gunadarma, Pengolahan citra: konsep dasar, Depok: Universitas Gunadarma, 2006.

[15] D. A. Forsyth and J. Ponce, Computer vision, California: Prentince Hall, 2002.