ROBOT BERODA PENGHINDAR MANUSIA MENGGUNAKAN *VECTOR FIELD HISTOGRAM* BERBASIS *POINT CLOUD*

PROPOSAL SKRIPSI

Oleh:

Devan Yusfa Sukmadya
210491100027

PROGRAM STUDI TEKNIK MEKATRONIKA

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS TRUNOJOYO MADURA

BANGKALAN

2024

ABSTRAK

Kemampuan robot untuk menghindari manusia sebagai rintangan dinamis menjadi tantangan penting dalam pengembangan robot otonom. Penelitian proposal ini berfokus pada implementasi robot beroda yang mampu menghindari manusia menggunakan *Vector Field Histogram* (VFH) berbasis data *input* dari *Point Cloud,* sensor kamera *kinect*. Data dari *point cloud* pada *kinect* digunakan untuk memetakan lingkungan tiga dimensi dan mendeteksi jarak antara posisi manusia pada kamera *kinect* secara *real-time*. Metode VFH dipilih karena kemampuannya dalam menghitung jalur navigasi optimal berdasarkan distribusi rintangan, termasuk manusia yang bergerak. Robot yang digunakan adalah robot beroda *Differential Drive Mobile Robot* (DDMR). Pengujian sistem dilakukan dalam skenario di mana manusia bergerak menghalangi lajur robot beroda pada lingkungan *real*, dengan tujuan untuk menilai efektivitas robot dalam menghindari tabrakan dan beradaptasi dengan perubahan lingkungan.

Kata Kunci**:** Robot beroda, *Vector Field Histogram*, *Point Cloud*, *Kinect*, Penghindaran manusia.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Kemampuan robot untuk menghindari manusia sebagai rintangan dinamis merupakan salah satu tantangan utama dalam pengembangan robot otonom. Dalam lingkungan dinamis seperti ruang publik atau fasilitas *indoor*, robot harus mampu beradaptasi dengan cepat terhadap perubahan posisi manusia dan objek lain yang bergerak.

Salah satu kawasan yang berkembang secara komprehensif adalah teknologi robotik robot penghindar halangan, yang merupakan contoh perangkat yang digunakan untuk meningkatkan keselamatan dan efektivitas pekerjaan manusia. Dalam banyak skenario, seperti layanan publik, kesehatan, dan industri, robot diharapkan mampu bergerak secara mandiri tanpa mengganggu aktivitas manusia [1].

Berbagai jenis sensor telah dikembangkan untuk mendukung kemampuan robot dalam mendeteksi dan merespons lingkungan. Sensor dari *kinect* merupakan salah satu kamera dengan sensor yang populer digunakan dalam robotika karena mampu menangkap data lingkungan secara tiga dimensi melalui teknologi *Point Cloud*. Dengan kemampuan ini, *Kinect* dapat digunakan untuk mendeteksi objek secara *real-time*, termasuk manusia dan rintangan dinamis lainnya. Namun, jika hanya memiliki data lingkungan saja tidak cukup, dibutuhkan untuk memproses data tersebut agar robot dapat mengambil keputusan yang tepat dalam menghindari rintangan dan merencanakan jalur yang aman [2].

Metode *Vector Field Histogram* (VFH) adalah salah satu pendekatan yang efektif untuk penghindaran rintangan pada robot otonom. VFH bekerja dengan memetakan lingkungan robot ke dalam *histogram polar* yang menggambarkan jarak dan arah rintangan robot. Berdasarkan data ini, Metode VFH kemudian menghitung jalur yang memungkinkan untuk dilalui dengan menghindari area-area berbahaya yang terdeteksi. Keunggulan metode ini adalah kemampuannya untuk bekerja dalam lingkungan dinamis dengan kecepatan yang disesuaikan, sehingga cocok diterapkan pada skenario di mana manusia atau objek lain dapat bergerak mendekati robot sewaktu-waktu [3].

Dalam proposal ini, kombinasi sensor kamera *kinect* dan metode VFH diharapkan dapat menciptakan sistem navigasi otonom yang andal untuk robot beroda. Dengan memanfaatkan data *point cloud* dari kamera *kinect*, robot akan mampu mendeteksi dan memahami lingkungan tiga dimensi di sekitar robot. Sementara itu, metode VFH akan memproses data tersebut untuk menentukan jalur yang aman dan menghindari rintangan dinamis seperti manusia yang bergerak. Penerapan teknologi ini diharapkan dapat berkontribusi pada pengembangan robot yang dapat digunakan di berbagai sektor, termasuk layanan publik, kesehatan, dan industri, di mana interaksi dengan manusia dan penghindaran rintangan dinamis sangat penting.

Tujuan proposal ini memanfaatkan sensor pada kamera *kinect* yang mampu menghasilkan data *Point Cloud* tiga dimensi, serta penambahan Metode *Vector Field Histogram* (VFH) sebagai metode untuk merancang robot beroda yang dapat menghindari manusia dalam lingkungan dinamis dan mengunakan kontrol motor PWM (*Pulse Width Modulation*).

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, beberapa rumusan masalah yang diidentifikasi dalam proposal ini adalah:

1. Bagaimana merancang robot beroda yang mampu menghindari manusia menggunakan metode VFH berbasis *point cloud* dari kamera *kinect*?
2. Bagaimana robot mendeteksi area lingkungan melalui *point cloud?*
3. Bagaimana menggerakkan posisi robot untuk menghindar ketika terdapat rintangan dinamis pada jalur robot?

1.3. Tujuan dan Manfaat

Adapun tujuan dan manfaat pada pelaksanaan proposal ini adalah berikut dibawah ini:

1. Dapat merancang sistem navigasi robot beroda yang mampu menghindari manusia menggunakan metode VFH berbasis *point cloud* dari sensor kamera *kinect*.
2. Merancang robot agar dapat mendeteksi adanya halangan melalui *point cloud*
3. Dapat merancang sistem yang memungkinkan robot mendeteksi dan menghindari halangan yang mendekat di jalur pergerakannya.

1.4. Batasan Masalah

Berikut merupakan batasan masalah pada proposal yang memiliki beberapa batasan yang perlu diperhatikan, yaitu:

1. Uji coba dilakukan dalam lingkungan tertutup dan terbatas dengan skenario *real*, dan 1 manusia sebagai rintangan dinamis utama.
2. Robot diuji cobakan pada ruang *indoor* secara *real-time* atau tanpa dimensi ruang minimal lebar 3,2 meter dan panjang 10 meter.
3. Area pengujian tidak memiliki halangan tambahan selain manusia yang menghalangi laju robot.
4. Target robot berupa *traffic cone* pada jarak 4,8m dari posisi robot beroda dijalankan, dengan derajat orientasi 90 derajat (lurus dengan orientasi robot *actual*).
5. Manusia akan berjalan berlawanan dengan arah robot setelah target robot dikenali dan mulai pergi ke titik target.

1.5. Metodologi Penelitian

Metodologi laporan proposal mempunyai lima tahapan, diantaranya tahap persiapan, tahap perancangan, tahap uji coba sistem, tahap analisis data, tahap evaluasi dan pembuatan laporan. Berikut penjelasannya:

1. Tahap Persiapan

   Pada tahap persiapan yang pertama menentukan topik proposal terlebih dahulu. Diantaranya adalah menentukan topik, membuat latar belakang, membuat judul, membuat proposal, melakukan studi pustaka
2. Tahap Perancangan

   Selanjutnya yakni tahapan yang dimana setelah proposal disetujui, maka dilakukan pembuatan alat yang akan diteliti.
3. Tahap Pengujian Sistem

   Pengambilan data serta pengujian alat dari sistem *hardware* maupun *software.* Diperlukan dokumentasi foto maupun video apabila melakukan uji coba alat.
4. Tahap Analisis

   Data yang diperoleh setelah pengujian alat akan dianalisis menggunakan metode yang diperlukan. Dari data yang sudah dianalisis maka akan diparafrase dalam

beberapa paragraf.

5. Tahap Pengerjaan Laporan Akhir

   Tahap pengerjaan akhir ini adalah seluruh data hasil pengujian sistem yang sudah dianalisis dimasukkan dalam laporan. Dari pengerjaan tersebut maka akan disimpulkan hasil dari pengujian alat.

1.6. Sistematika Penulisan

Pada proposal robot beroda penghindar manusia menggunakan *vector field histogram* berbasis *point cloud* terdapat sistematika penulisan antara lain:

1. **BAB I: Pendahuluan**

   Bab ini berisi tentang penjelasan latar belakang permasalahan, perumusan masalah, tujuan dari penelitian, manfaat dari penelitian, batasan masalah serta sistematika penulisan.

2. **BAB II: Tinjauan Pustaka**

   Bab ini berisi tentang penjelasan teori serta metode yang akan digunakan untuk membuat Metode serta konponen-komponen yang digunakan pada penelitian.

4. **BAB III: Perancangan Sistem**

   Bab ini berisi tentang penjelasan langkah-langkah yang dilakukan pada proses pembuatan sistem yang berupa metode dan desain dari alat tersebut.

5. **BAB IV: Hasil dan Pembahasan**

   Bab ini berisi tentang penjelasan dari pengujian sistem dari sistem *hardware* maupun *software*, serta analisis dari data yang diperoleh pada pengujian sistem.

5. **BAB V: Penutup**

   Bab ini berisi tentang penjelasan kesimpulan dan saran dari hasil penelitian yang sudah dilakukan.

1.7. Jadwal Pelaksanaan Penelitian

Bagian ini menjelaskan rencana waktu atau jadwal yang akan dilakukan selama pelaksanaan penelitian tentang "Robot Beroda Penghindar Manusia Menggunakan *Vector Field Histogram* Berbasis *Point Cloud*." Bagian ini penting karena memberikan gambaran tentang bagaimana penelitian ini akan dilakukan secara sistematis dan teratur. Terdapat penjelasan rinci tentang rencana waktu yang akan digunakan untuk melaksanakan seluruh rangkaian kegiatan penelitian.

Jadwal pelaksanaan penelitian ini juga membantu memberikan gambaran yang jelas kepada pembaca proposal mengenai bagaimana penelitian ini akan dilaksanakan dan berapa lama waktu yang diperlukan untuk menyelesaikannya. Tahapan-tahapan dalam penelitian ini, yang melibatkan "Robot Beroda Penghindar Manusia Menggunakan *Vector Field Histogram* Berbasis *Point Cloud*," dijadwalkan secara terperinci pada Tabel 1.1.

Tabel 1.1. Rencana Kegiatan Penelitian

No.	**Nama Kegiatan**	**Bulan**					
		I	**II**	**III**	**IV**	**V**	**VI**
1.	Studi literatur.						
2.	Desain robot beroda 3D *Differential Drive Mobile Robot* Pada Thinkercad						
3.	Pembuatan rangkaian robot beroda, *Differential Drive Mobile Robot*.						
4.	Perancangan logika *vector field histogram* pada simulasi python						
5.	Pembuatan program *point cloud* pada kamera *kinect*						
6.	pengujian robot menggunakan *vector field histogram* berbasis *point cloud*						
7.	Pengolahan dan analisis data.						
8.	Pembuatan laporan penelitian.						

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian terkait penggunaan *ultrasonic* dan *Vector Field Histogram* (VFH) dalam navigasi robot telah dilakukan oleh Saleh Gustanto dkk. Salah satu penelitian menunjukkan bahwa *ultrasonic*, dengan kemampuan proyeksi pola inframerah, dapat digunakan secara efektif untuk mendeteksi objek dalam ruangan dengan akurasi tinggi. Sensor ini memberikan informasi kedalaman yang memadai bahkan dalam kondisi pencahayaan yang bervariasi, sehingga sangat berguna dalam pengoperasian robot beroda di dalam ruangan yang kompleks. Berdasarkan hasil penelitian yang telah dilakukan, maka dapat disimpulkan bahwa, Kinerja algoritma *Vector Field Histogram* yang diterapkan sebagai pengendali perilaku menghindari rintangan pada robot *soccer* beroda [4]

2.2. *Kinect*

Kinect yang awalnya dirancang untuk konsol Xbox 360, adalah sebuah perangkat sensor gerak yang dikembangkan oleh Microsoft. Perangkat ini memungkinkan pengguna untuk berinteraksi dengan sistem tanpa menggunakan kontroler fisik, melainkan dengan gerakan tubuh, suara, dan pengenalan wajah. *Kinect* dilengkapi dengan berbagai sensor, termasuk kamera RGBD, sensor inframerah, dan mikrofon, yang bekerja secara terintegrasi untuk menangkap gerakan dan suara dari lingkungan. Teknologi ini memungkinkan *Kinect* untuk memetakan posisi pengguna dalam ruang tiga dimensi, membuatnya ideal untuk game dan aplikasi interaktif.

Salah satu fitur utama dari *kinect* adalah kemampuan untuk mendeteksi dan melacak kerangka (*skeleton tracking*) dari satu atau lebih pengguna secara *real-time*. Dengan sensor inframerah, *kinect* dapat mendeteksi kedalaman dan jarak pengguna dari perangkat, memungkinkan pelacakan yang akurat bahkan dalam kondisi pencahayaan yang rendah. Selain itu, *kinect* dapat mengenali gerakan tertentu, seperti melambaikan tangan atau melompat, yang kemudian dapat digunakan sebagai input untuk mengontrol game atau aplikasi lainnya. Fitur pengenalan suara juga menjadi elemen penting, di mana pengguna dapat memberikan perintah suara untuk mengontrol perangkat.

Meskipun awalnya dikembangkan untuk game, *kinect* telah digunakan di berbagai bidang lainnya, termasuk dalam dunia pendidikan, kesehatan, dan penelitian. Dalam aplikasi mekatronika, *kinect* digunakan untuk pemetaan lingkungan, pelacakan gerak, dan interaksi robot. Kemampuannya untuk menangkap gerakan dan posisi secara presisi membuatnya menjadi alat yang berguna dalam penelitian robotika dan *augmented reality*. Meskipun saat ini sudah ada versi yang lebih canggih, *kinect* tetap menjadi teknologi penting yang menandai awal dari era interaksi manusia-mesin yang lebih intuitif dan alami.

Selain itu, *kinect* memiliki peran penting dalam perkembangan teknologi interaksi tanpa kontak (*contactless interaction*), yang memungkinkan pengguna untuk berinteraksi dengan perangkat tanpa menyentuhnya. Teknologi ini menjadi landasan bagi berbagai inovasi di luar dunia hiburan. Misalnya, dalam bidang kesehatan, *Kinect* digunakan untuk rehabilitasi fisik, di mana pasien dapat melakukan latihan gerakan yang dipantau oleh sistem, memungkinkan dokter untuk memantau perkembangan pasien tanpa perlu kehadiran fisik. Fitur ini menawarkan cara baru dalam *telemedicine*, yang semakin relevan di era digital.

Dalam perkembangannya, *kinect* menjadi lebih dari sekadar alat hiburan. Teknologi ini menjadi dasar untuk penelitian lebih lanjut dalam pengenalan gerak, pemrosesan gambar, dan pengembangan antarmuka yang lebih alami dan intuitif. Perangkat ini membantu menjembatani kesenjangan antara dunia fisik dan virtual, memberikan kontribusi signifikan terhadap evolusi teknologi yang memadukan kecerdasan buatan, visi komputer, dan interaksi manusia-mesin. Meskipun kamera *kinect* tidak lagi diproduksi, warisannya tetap berlanjut dalam berbagai aplikasi teknologi modern [5]. Berikut merupakan Gambar 2.1 kamera *kinect*.

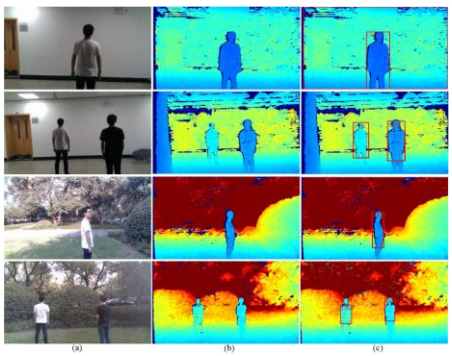
Gambar 2.1. Kamera *Kinect*

2.3. *Depth image prossesing*

Pemrosesan citra kedalaman (*depth image processing*) menggunakan *kinect* adalah salah satu point penting, terutama karena sensor inframerahnya mampu menghasilkan peta kedalaman dari lingkungan. Citra kedalaman yang dihasilkan adalah representasi visual di mana setiap piksel mewakili jarak suatu objek dari sensor Kinect. Teknologi ini memungkinkan pemetaan ruang 3D secara *real-time*, yang sangat berguna dalam berbagai aplikasi seperti pelacakan gerak, pemetaan lingkungan, dan interaksi robot.

Kinect bekerja dengan memancarkan sinar inframerah yang kemudian dipantulkan oleh objek-objek di sekitar kamera. Sensor inframerah menangkap pantulan ini dan menghitung jarak antara sensor dan objek berdasarkan waktu yang diperlukan sinar untuk kembali. Data kedalaman yang dihasilkan disusun dalam bentuk peta kedalaman (*depth map*), di mana setiap nilai piksel menunjukkan jarak dalam satuan milimeter dari objek ke sensor. Peta ini dapat diproses lebih lanjut menggunakan Metode khusus untuk mengekstraksi informasi yang lebih kompleks, seperti deteksi bentuk tubuh, pelacakan gerakan, atau identifikasi objek [6].

Dalam aplikasi pemrosesan citra kedalaman, *Kinect* sering digunakan untuk mengidentifikasi dan melacak objek secara presisi di lingkungan tiga dimensi. Misalnya, dalam pengembangan robotika, citra kedalaman memungkinkan robot untuk "melihat" lingkungan, mendeteksi rintangan, atau mengenali objek yang perlu diambil. Pada aplikasi interaksi manusia-mesin, peta kedalaman digunakan untuk pelacakan gerak tubuh atau tangan. *Kinect* SDK menyediakan akses ke data kedalaman, yang bisa diintegrasikan dengan berbagai bahasa pemrograman, termasuk Python, C++, dan C#. Berikut merupakan tampilan Gambar 2.2 *output depth prossesing* mengunakan *kinect* pada program *python.*



Gambar 2.2. *Output Depth Prossesing* mengunakan *Kinect* pada program *Python*

Pada hasi program pada Gambar 2.2. *Depth Prossesing* mengunakan *kinect* untuk mendeteksi wajah dan menghitung jarak antara sensor dengan wajah yang terdeteksi, berdasarkan pemrosesan citra kedalaman dan warna. Terdapat *stream* data yang diambil dari *Kinect*: *stream* warna (RGB) dan *stream* kedalaman (*Depth*). Data kedalaman digunakan untuk menghitung jarak objek, sedangkan data warna digunakan untuk mendeteksi wajah menggunakan detektor wajah *Haar Cascade*, Pada bagian utama program, citra kedalaman yang diperoleh dari sensor *kinect* disimpan sebagai array dua dimensi, di mana setiap piksel berisi nilai kedalaman dalam satuan milimeter. Untuk menghitung jarak dari wajah yang terdeteksi, program mencari titik tengah wajah menggunakan koordinat pusat persegi panjang yang menutupi wajah.

Perhitungan kedalaman (*Depth*) menggunakan Kinect didasarkan pada sensor inframerah yang mampu menangkap informasi jarak dari objek ke perangkat. Sensor ini bekerja dengan memancarkan pola inframerah yang dipantulkan oleh objek di serkitar, dan dari pantulan tersebut, *Kinect* menghitung jarak objek berdasarkan seberapa cepat sinyal tersebut kembali ke sensor. Perhitungan kedalaman terjadi dalam fungsi *calculate_distance*. Ketika wajah terdeteksi pada citra warna, koordinat titik tengah wajah (X_{center}, Y_{center}) dihitung dengan rumus pada persamaan 2.1.

$$X_{center} = x + \frac{w}{2} \ , \ Y_{center} = y + \frac{h}{2} \tag{2.1}$$

Di mana x dan y adalah koordinat sudut kiri atas dari kotak pembatas wajah, dan w serta h adalah lebar dan tinggi kotak tersebut. Setelah itu, nilai kedalaman pada titik pusat wajah diambil dari peta kedalaman (*depth map*) yang dihasilkan oleh Kinect. Nilai kedalaman yang diambil dari *Limit* adalah nilai 16-bit yang mewakili jarak dalam milimeter. Untuk mendapatkan jarak dalam meter, nilai kedalaman tersebut dibagi dengan 1000, sesuai pada persamaan 2.2.

$$Depth\_in\_meters = \frac{Depth\_value}{1000} \tag{2.2}$$

Misalnya, jika sensor memberikan nilai kedalaman sebesar 1500 (milimeter), maka jaraknya akan menjadi:

$$Depth\_in\_meters = \frac{1500}{1000} = 1.5 \text{ meter}$$

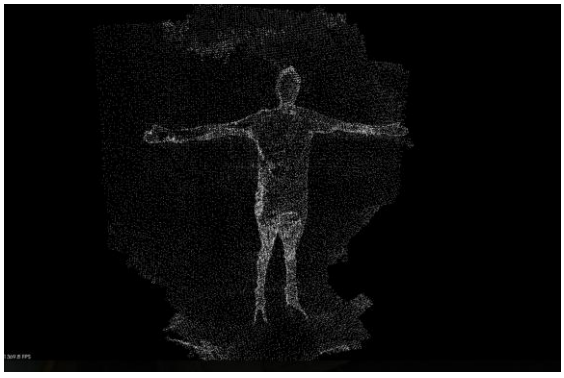
Proses tersebut mengonversi nilai kedalaman dari milimeter ke meter untuk kemudahan

interpretasi jarak. Nilai kedalaman ini kemudian digunakan untuk menampilkan jarak wajah dari sensor secara real-time di antarmuka grafis, serta dicetak ke layar dalam satuan milimeter.

2.4. *Depth Prossesing Point Cloud*

Dengan menggunakan sensor inframerah pada *kinect*, *kinect* mampu menghasilkan data kedalaman (*depth*) yang berguna untuk berbagai aplikasi, seperti pelacakan gerak, pemetaan 3D, dan interaksi manusia-mesin. Salah satu implementasi dari sensor ini adalah pembuatan *point cloud*, yaitu representasi ruang tiga dimensi (3D) yang terdiri dari titik-titik dengan koordinat tertentu dalam sumbu x, y, dan z. *Point cloud* memungkinkan visualisasi dan pemahaman objek serta permukaan dalam ruang 3D secara lebih rinci dan presisi [7].

Pemrosesan citra kedalaman dari sensor seperti *Kinect* dapat dilakukan dengan bantuan perangkat lunak seperti OpenNI dan PCL (*Point Cloud Library*) yang menyediakan antarmuka untuk menangkap data kedalaman dan warna secara simultan. Dengan memanfaatkan parameter intrinsik kamera, data kedalaman yang diperoleh dapat diproyeksikan ke dalam koordinat 3D untuk membangun *point cloud*. Program ini mendemonstrasikan cara mendeteksi kedalaman objek dari sensor *Kinect* dan menampilkan hasil *point cloud* pada citra RGB yang diambil secara bersamaan. Berikut merupakan Gambar 2.3 *output point cloud* pada *kinect* mengunakan *depth prossessing.*



Gambar 2.3. *Output Point Cloud* Pada *Kinect* mengunakan *Depth Prossessing*

Persamaan yang digunakan pada hasil *output* pada Gambar 2.3. *Output Point Cloud* Pada *Kinect* mengunakan *Depth Prossessing* dalam program untuk menghitung koordinat 3D dari setiap piksel dalam citra kedalaman adalah persamaan proyeksi kamera. Persamaan ini menghubungkan koordinat 2D piksel pada citra dengan posisi 3D

pada *real-time* berdasarkan rumus *depth prossesing*. Rumus ditunjukan pada persamaan 2.3.

$$x = \frac{(u - c_x) \,.\, z}{f_x}, \quad y = \frac{(v - c_y) \,.\, z}{f_y} \tag{2.3}$$

Pada persamaan 2.3, x dan y adalah koordinat dalam ruang 3D, sementara u dan v adalah koordinat piksel pada citra 2D. Variabel z mewakili nilai kedalaman (*depth*) pada piksel tertentu, yang menunjukkan jarak objek dari sensor dalam milimeter. Parameter f_x dan f_y adalah panjang fokus (*focal length*) dalam arah sumbu-x dan sumbu-y, masing-masing sebesar 525 mm dalam program tersebut, sedangkan c_x dan c_y adalah pusat optik kamera (*optical center*), yang berada di tengah citra dan dalam contoh ini bernilai 319,5 untuk sumbu-x dan 239,5 untuk sumbu-y.

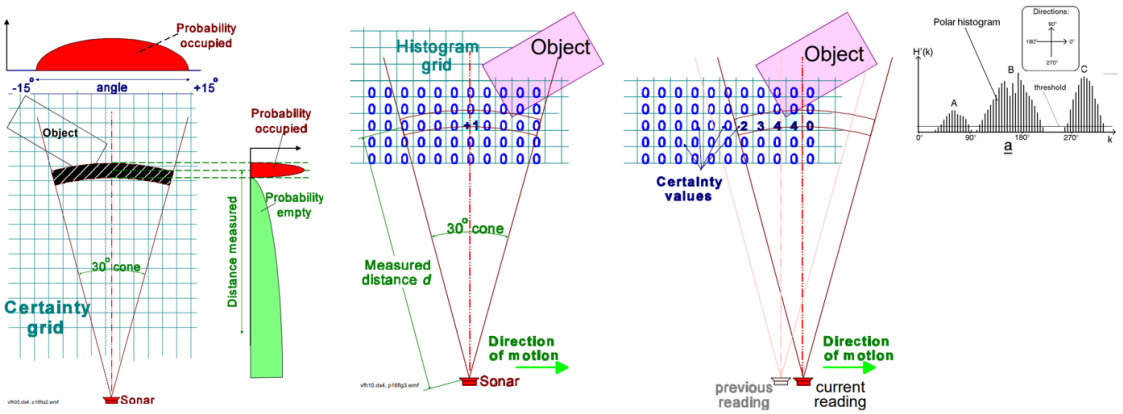
Bagaimana data kedalaman diubah menjadi koordinat 3D. Koordinat 2D pada citra (u, v) dikalikan dengan nilai kedalaman (z) yang menunjukkan jarak antara objek dan sensor, kemudian dibagi dengan panjang fokus untuk mengembalikan koordinat 3D yang sesuai. Akhirnya, x dan y dihasilkan sebagai posisi objek di ruang relatif terhadap sensor. Persamaan ini memungkinkan transformasi data kedalaman menjadi representasi spasial dalam bentuk *point cloud*, yang digunakan untuk memahami posisi objek dalam ruang 3D secara akurat.

2.5. Metode *Vector Field Histogram (VFH)*

Vector Field Histogram (VFH) merupakan metode penghindaran rintangan dalam robotika, terutama untuk sistem robot bergerak di lingkungan dinamis. VFH dapat menggunakan data dari sensor seperti *kinect* untuk membentuk representasi spasial lingkungan disekitar robot. Informasi tersebut kemudian diolah menjadi histogram arah (*polar histogram*) yang merepresentasikan kepadatan rintangan di setiap sektor di sekitar robot. Nilai histogram dihitung berdasarkan jarak dan sudut antara robot dan rintangan yang terdeteksi. Semakin tinggi nilai histogram di suatu sektor, semakin besar kemungkinan adanya rintangan, yang menunjukkan bahwa sektor tersebut berbahaya untuk dilalui [8].

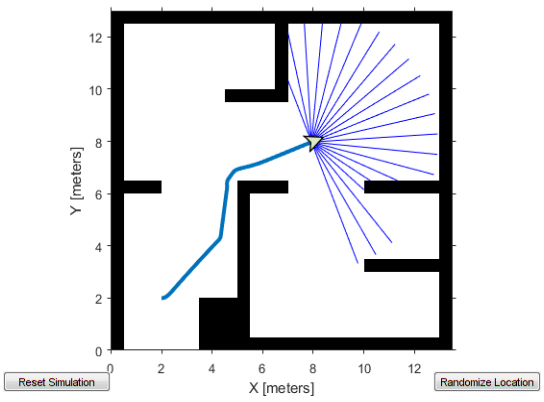
Setelah histogram terbentuk, metode VFH menentukan sektor dengan nilai histogram terendah sebagai jalur yang paling aman. Arah aman ini dipilih dengan mempertimbangkan target yang ingin dicapai robot, sehingga robot dapat tetap menuju

tujuan akhir sambil menghindari rintangan di jalurnya. Dengan pembaruan terus-menerus dari sensor *Kinect*, robot dapat bereaksi secara real-time terhadap perubahan lingkungan dan pergerakan rintangan, baik objek statis maupun dinamis seperti manusia. Berikut merupakan kinerja VFH yang ditunjukan pada Gambar 2.4.



Gambar 2.4. Kinerja VFH

Keunggulan utama VFH adalah kemampuannya bekerja dalam waktu nyata dan menangani data sensor yang kompleks seperti *Point Cloud* dari *Kinect*. Metode ini tidak memerlukan sensor tambahan seperti ultrasonik atau LiDAR, sehingga mengurangi kompleksitas sistem. Dengan hanya menggunakan data kedalaman dari *Kinect*, robot dapat menghindari rintangan dengan akurat sambil menjaga fleksibilitas desain, sehingga ruang untuk pengembangan tambahan pada robot atau penerapan dalam skala yang lebih kecil tetap tersedia. Berikut merupakan visualisasi VFH yang ditunjukan pada Gambar 2.5. Visualisasi metode *Vector Field Histogram*.

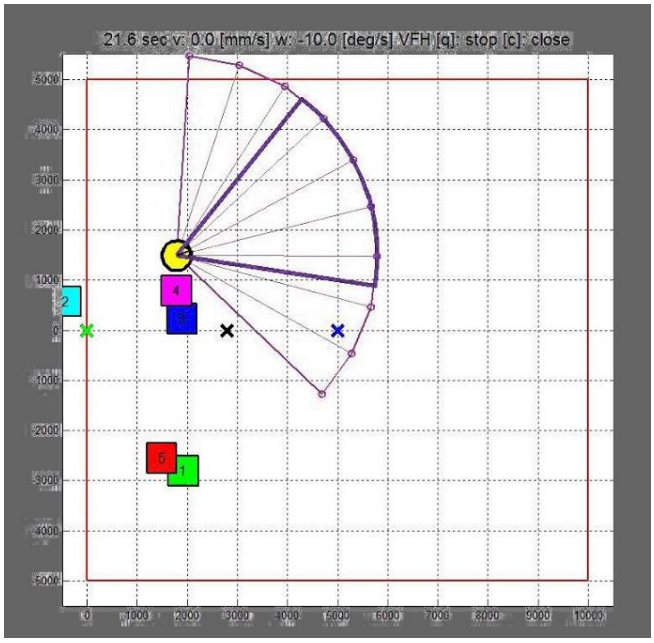


Gambar 2.5. Visualisasi Metode *Vector Field Histogram*

2.6. Metode VFH sebagai *Collison Avoiding*

Vector Field Histogram (VFH) bekerja dengan prinsip mengubah lingkungan robot menjadi grid yang terus diperbarui oleh sensor, seperti lidar atau kamera Kinect. Setiap sel mewakili keberadaan objek dengan nilai tertentu berdasarkan jarak dan orientasi objek terhadap robot. Dari data ini, VFH membangun *histogram polar*, yang merupakan representasi data berbentuk lingkaran dengan nilai yang lebih tinggi menunjukkan keberadaan rintangan pada arah tertentu. *Histogram* ini kemudian digunakan untuk menentukan jalur bebas rintangan yang optimal bagi robot.

Pada tahap berikutnya, VFH menganalisis *histogram polar* untuk mengidentifikasi sektor-sektor bebas hambatan dan memilih arah yang paling aman dan efisien untuk dilalui. Metode ini memperhitungkan faktor-faktor seperti orientasi robot, kecepatan, dan arah tujuan untuk memilih sudut orientasi terbaik yang mendekati jalur target tanpa menabrak rintangan. Dengan menggabungkan reaksi cepat terhadap lingkungan lokal dan kemampuan untuk tetap mengikuti jalur utama, VFH mampu memberikan solusi efektif untuk robot agar tetap bergerak menuju tujuan sambil secara dinamis menghindari rintangan [9].Berikut merupakan tampilan simulasi VFH sebagai *collison avoidance* pada *pure pursuit* yang ditunjukan pada Gambar 2.6.



Gambar 2.6. Simulasi VFH sebagai *Collision Avoiding* Pada *Pure Pursuit*

Logika dalam mengimplementasikan Metode Pure Pursuit untuk navigasi robot menuju target, dengan penambahan metode *Vector Field Histogram* (VFH) untuk

menghindari rintangan bergerak. Berikut penjelasan persamaan dan cara kerja metode pada simulasi Gambar 2.6, yaitu:

1. Navigasi dengan *Pure Pursuit*

   Pure Pursuit bertugas untuk mengarahkan robot ke target. Pada setiap iterasi, sudut *alpha* (α) dihitung menggunakan persamaan 2.4.

   $$\alpha = arctan2\left(d_y, d_x\right) - \theta \quad (2.4)$$

   Dimana:

   d_y dan d_x adalah selisih koordinat antara target dan posisi robot.

   θ adalah orientasi robot. Sudut alpha (α) digunakan untuk mengarahkan robot ke target berdasarkan selisih sudut antara arah target dan orientasi robot.

2. Penghindaran Rintangan dengan *Vector Field Histogram* (VFH)

   Metode VFH bertanggung jawab untuk mendeteksi rintangan di ser robot dan mengarahkan robot agar menghindarinya. Beberapa langkah penting dalam VFH meliputi:

   a. Pemindaian Sensor: Robot memproyeksikan beberapa balok sensor (dalam kode ini 40 balok) dalam bidang pandang *field_of_view* sebesar 90 derajat ($\frac{\pi}{2}$). Setiap balok memiliki sudut yang diukur dari orientasi robot (θ).

   b. Deteksi Rintangan: Setiap rintangan dalam lingkungan dihitung jaraknya dari robot menggunakan persamaan jarak *Euclidean* pada persamaan 2.5

      $$dist = \sqrt{(Obsticle_x - x)^2 + (Obsticle_y - y)^2 - r} \quad (2.5)$$

      Di mana r adalah radius rintangan. Jika rintangan berada dalam jangkauan sensor dan dalam bidang pandang, balok sensor yang terdekat ke arah rintangan disesuaikan untuk memperbarui jarak terdekat yang terdeteksi oleh balok tersebut.

   c. Pemilihan Sudut Aman: Setelah jarak untuk semua balok dihitung, sudut dari balok yang paling jauh dari rintangan dipilih sebagai sudut aman (*safe_angle*), yang merupakan arah aman untuk dilalui robot.

3. Perubahan Orientasi Robot

   Setelah sudut aman (*avoidance_angle*) dihitung oleh VFH, orientasi robot θ diperbarui menggunakan faktor perataan (*smoothing_factor*). Jika jarak terdekat dari rintangan lebih kecil dari ambang batas penghindaran (*avoidance_threshold*),

maka θ diperbarui sebagian ke arah sudut aman menggunakan persamaan 2.6.

$$\theta = (smoothing\_factor \times \theta + ((-smoothing\_factor) \times avoidance\_angle) \quad (2.6)$$

Jika tidak ada rintangan di dekat robot, θ diperbarui mengikuti Metode *Pure Pursuit* untuk mengarahkan robot menuju target yang ditunjukan pada persamaan 2.7.

$$\theta += \alpha \times dt \quad (2.7)$$

Pada persamaan tersebut, dt adalah waktu interval yang digunakan untuk memperbarui posisi dan orientasi robot.

4. Pergerakan Robot

Setelah orientasi diperbarui, posisi x dan y robot diperbarui menggunakan kecepatan v dan orientasi θ dengan persamaan gerak dasar yaitu persamaan 2.8 dan 2.9.

$$x += v \times cos\,(\theta) \times dt \quad (2.8)$$

$$y += v \times cos\,(\theta) \times dt \quad (2.9)$$

Dengan demikian, metode ini memungkinkan robot untuk secara dinamis menghindari rintangan yang bergerak sambil tetap bergerak menuju target yang ditentukan mengunakan Metode VFH (*Vector Field Histogram*) sebagai *Obstacle Avoiding*

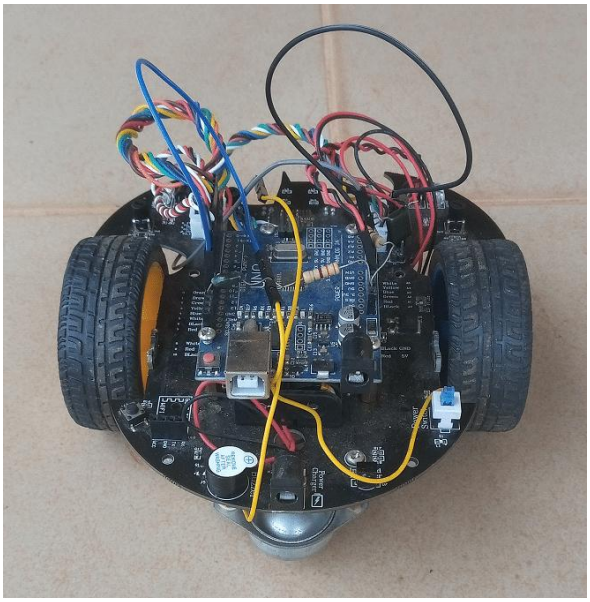
2.7. Robot Beroda *Differential Drive Mobile Robot*

Differential Drive Mobile Robot (DDMR) adalah jenis robot bergerak yang menggunakan dua roda yang digerakkan secara independen untuk navigasi. Kedua roda ini ditempatkan di sisi kiri dan kanan robot, sementara roda bebas atau pelari tambahan yang tidak digerakkan sering digunakan untuk menjaga keseimbangan. Sistem penggerak ini memungkinkan robot untuk bergerak dalam berbagai arah, termasuk berbelok di tempat dengan memutar kedua roda ke arah yang berlawanan. Karena kedua roda dapat berputar pada kecepatan yang berbeda, robot ini memiliki kemampuan manuver yang baik, membuatnya cocok untuk digunakan dalam berbagai aplikasi yang memerlukan mobilitas tinggi, seperti robotika industri, robot pembersih, dan robot penelitian.

Pada sistem *differential drive*, pergerakan robot bergantung pada kecepatan relatif antara roda kiri dan roda kanan. Jika kedua roda berputar pada kecepatan yang sama, robot akan bergerak lurus. Jika satu roda berputar lebih cepat dari yang lain, robot akan

berbelok ke arah roda yang bergerak lebih lambat. Untuk mengubah arah secara signifikan atau memutar di tempat, kedua roda dapat diputar ke arah yang berlawanan. Perbedaan kecepatan dan arah rotasi antara kedua roda menghasilkan perubahan lintasan atau orientasi robot, membuat kontrol navigasi DDMR cukup fleksibel dan presisi [9].

Salah satu keunggulan utama dari robot ini adalah kesederhanaan desainnya, yang tidak memerlukan mekanisme kemudi yang kompleks. Namun, tantangan muncul dalam hal pengendalian gerakan dan stabilitas lintasan, terutama pada permukaan yang tidak rata atau dalam skenario yang melibatkan banyak rintangan. Penggunaan Metode kontrol seperti Pure Pursuit dan teknik penghindaran rintangan, seperti *Vector Field Histogram* (VFH), sering diperlukan untuk membantu robot bergerak secara efisien di lingkungan yang dinamis atau kompleks. Berikut merupakan Gambar 2.7. robot beroda ddmr.



Gambar 2.7. Robot Beroda *Differential Drive Mobile Robot*

2.7.1. Kecepatan Linear (v)

Besaran yang menunjukkan seberapa cepat sebuah objek bergerak sepanjang lintasan lurus. Dalam konteks robotik, kecepatan linear mengacu pada kecepatan translasi pusat massa robot dalam arah tertentu, yang biasanya diukur dalam meter per detik (m/s). Kecepatan dihitung sebagai rata-rata kecepatan roda kiri dan kanan pada sistem penggerak diferensial. Kecepatan linear penting untuk menentukan seberapa cepat robot dapat bergerak maju atau mundur, dan merupakan komponen utama dalam navigasi robot untuk bergerak dari satu titik ke titik lain dengan efisien. Berikut adalah persamaan yang

digunakan kecepatan linear untuk *forward kinematics* yang ditujukan pada persamaan 2.10.

$$v = \frac{v_{left} + v_{right}}{2} \quad (2.10)$$

Cara untuk menghitung kecepatan linear ($\boldsymbol{v}$) sebuah robot dengan *forward kinematics*:

1. v adalah kecepatan linear dari robot, yaitu seberapa cepat robot bergerak maju atau mundur.
2. v_{left} adalah kecepatan roda kiri.
3. v_{right} adalah kecepatan roda kanan.

Persamaan ini mengambil rata-rata kecepatan dari kedua roda untuk menentukan kecepatan linear keseluruhan dari robot. Dengan kata lain, kecepatan angular robot adalah rata-rata dari kecepatan kedua roda.

2.7.2. Kecepatan Angular (ω)

Kecepatan angular merupakan ukuran seberapa cepat sebuah objek berputar atau mengubah orientasinya, dan dalam konteks robot berpenggerak diferensial, ini merujuk pada seberapa cepat robot dapat berbelok. Kecepatan angular (ω) dihitung dengan menggunakan perbedaan kecepatan antara roda kiri (v_{left}) dan roda kanan (v_{right}) serta jarak antara kedua roda tersebut (d) Besaran ini diukur dalam radian per detik (rad/s). Kecepatan angular sangat penting untuk navigasi robot, karena memungkinkan robot untuk berbelok dan menghindari hambatan, sehingga membuat robot lebih responsif dan adaptif terhadap perubahan lingkungan. Berikut adalah persamaan yang digunakan kecepatan angular untuk *forward kinematics* yang ditujukan pada persamaan 2.11.

$$\omega = \frac{v_{left} - v_{right}}{d} \quad (2.11)$$

Untuk menghitung kecepatan angular (ω) sebuah robot dengan sistem penggerak diferensial. Berikut adalah pengertian dari komponen-komponen dalam persamaan 2.11

1. ω adalah kecepatan angular, yang mengukur seberapa cepat robot berputar atau mengubah orientasinya, dinyatakan dalam radian per detik (rad/s).
2. v_{left} adalah kecepatan roda kiri.
3. v_{right} adalah kecepatan roda kanan.

4. d adalah jarak antara roda kiri dan roda kanan, yang dikenal juga sebagai *track width* atau *wheelbase*, dinyatakan dalam meter.

Persamaan ini mengukur seberapa cepat robot dapat berbelok berdasarkan perbedaan kecepatan antara kedua rodanya dan jarak di antara mereka. Kecepatan angular penting untuk mengontrol gerakan berbelok robot, memastikan robot dapat menavigasi dengan tepat dan efisien melalui berbagai lintasan dan lingkungan. Kecepatan angular (ω) menggambarkan tingkat perubahan orientasi robot. Jika kecepatan roda kanan (v_{right}) lebih besar dari kecepatan roda kiri (v_{left}), robot akan berputar ke kiri, menghasilkan kecepatan angular positif. Sebaliknya, jika (v_{left}) lebih besar dari (v_{right}), robot akan berputar ke kanan, menghasilkan kecepatan angular negatif.

2.7.3. *Forward Kinematics*

Forward kinematics dalam konteks robot penggerak diferensial merupakan proses perhitungan posisi dan orientasi (*pose*) robot berdasarkan kecepatan linear dan kecepatan angular dari kedua roda. Dengan menggunakan *forward kinematics*. [10] Robot dapat memprediksi posisi akhir robot setelah bergerak selama *interval* waktu tertentu, berdasarkan kecepatan roda kiri dan roda kanan. Merupakan langkah penting dalam pengendalian robot, karena memungkinkan untuk menghitung lintasan yang diambil robot dari posisi awal ke posisi akhir.

Pada robot berpenggerak diferensial, ada dua parameter utama yang digunakan untuk menghitung gerakan robot, yaitu kecepatan *linear* (v) dan kecepatan *angular* (ω). Kecepatan *linear* merupakan hasil rata-rata kecepatan translasi dari kedua roda, sedangkan kecepatan angular ditentukan oleh perbedaan kecepatan antara roda kiri dan roda kanan serta jarak antar roda (d). Untuk menghitung posisi robot (x, y) dan orientasi θ setelah bergerak selama waktu (t), persamaan *forward kinematics* pada persamaan 2.12.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = 2\pi r \begin{bmatrix} 1/2 & 1/2 \\ -1/d & 1/d \end{bmatrix} \begin{bmatrix} \theta_L \\ \theta_R \end{bmatrix} \tag{2.12}$$

Pada persamaan 2.12, kecepatan linear v dan kecepatan angular ω dinyatakan dalam bentuk matriks yang menghubungkan sudut roda kiri (θ_L) dan sudut roda kanan (θ_R) dengan radius roda r dan jarak antara roda d. Persamaan tersebut menunjukkan hubungan antara sudut rotasi roda kiri dan kanan terhadap kecepatan linear dan angular robot yaitu:

1. v Kecepatan linear robot.

2. ω Kecepatan angular (putar) robot.
3. r Radius roda.
4. d Jarak antara roda kiri dan kanan (*track width*).
5. θ_{L} Sudut rotasi roda kiri.
6. θ_{R} Sudut rotasi roda kanan.

2.7.4. *Inverse Kinematics*

Pada robot berpenggerak diferensial (*Differential Drive Mobile Robot*, DDMR), *inverse kinematics* (IK) merupakan proses untuk menentukan kecepatan sudut roda kiri dan kanan berdasarkan posisi dan orientasi target robot [10]. Berbeda dengan *forward kinematics,* yang memprediksi posisi robot berdasarkan kecepatan roda, *inverse kinematics* diperlukan ketika robot harus mencapai posisi tertentu pada lintasan. Dalam *inverse kinematics*, mengetahui posisi target (x, y) serta sudut orientasi θ yang ingin dicapai oleh robot, dan ingin menentukan kecepatan roda kiri dan roda kanan. Pada DDMR, perhitungan kecepatan sudut roda kiri (ω_L) dan kanan (ω_R) didasarkan pada kecepatan linier (v) dan kecepatan angular (ω), yang dapat diperoleh dari persamaan 2.13 dan persamaan 2.14.

$$v = \frac{r}{2}(\omega_R + \omega_L) \tag{2.13}$$

$$\omega = \frac{r}{d}(\omega_R + \omega_L) \tag{2.14}$$

Dengan persamaan 2.13 dan persamaan 2.14, dapat menentukan berapa kecepatan masing-masing roda yang diperlukan untuk mencapai kecepatan *linear v* dan kecepatan *angular* ω, yang menggerakkan robot ke posisi dan orientasi target. Di mana:

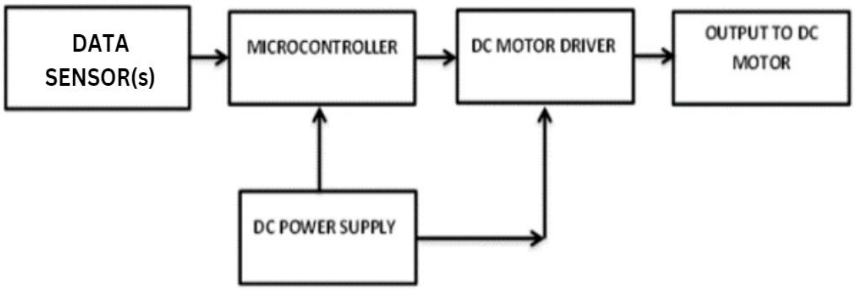
1. *v* adalah kecepatan linear robot,
2. ω adalah kecepatan angular (putaran) robot,
3. *r* adalah radius roda,
4. *d* adalah jarak antara roda kiri dan kanan (*track width*),
5. ω_L adalah kecepatan sudut roda kiri,
6. ω_R adalah kecepatan sudut roda kanan.

2.8. Kontrol Motor PWM (*Pulse Width Modulation*)

Kontrol motor merupakan aspek krusial dalam pengoperasian robot, terutama dalam memastikan pergerakan yang tepat dan responsif. Salah satu metode yang banyak

digunakan untuk mengontrol kecepatan dan torsi motor DC adalah teknik *Pulse Width Modulation* (PWM). Dengan teknik ini, lebar pulsa sinyal digital yang dikirimkan ke motor dapat diatur, memungkinkan pengendalian yang efisien terhadap daya yang diberikan. PWM menawarkan kelebihan dalam hal efisiensi energi, karena memungkinkan motor beroperasi pada kecepatan yang lebih rendah dengan konsumsi daya yang lebih sedikit dibandingkan metode kontrol lainnya. Selain itu, kontrol PWM juga memberikan respons yang lebih cepat terhadap perubahan sinyal, sehingga sangat cocok untuk aplikasi robotika yang memerlukan penyesuaian *real-time* dalam navigasi dan penghindaran rintangan [11].

Dalam konteks robot beroda yang dirancang untuk menghindari manusia dan rintangan dinamis, implementasi kontrol motor PWM menjadi sangat penting. Dengan memanfaatkan data yang diperoleh dari sensor, seperti *Kinect*, robot dapat mengambil keputusan yang lebih akurat dan responsif dalam menentukan arah dan kecepatan pergerakannya. Dengan demikian, pengendalian motor yang baik melalui teknik PWM akan berkontribusi besar terhadap kinerja keseluruhan robot dalam lingkungan yang kompleks. Berikut merupakan diagram blok PWM yang ditujukan pada Gambar 2.8.



Gambar 2.8. Diagram Block PWM (*Pulse Width Modulation*)

Setelah mengetahui kecepatan yang diinginkan, menghitung siklus tugas (*duty cycle*) PWM yang diperlukan. Siklus tugas ini menunjukkan berapa lama sinyal PWM harus "hidup" untuk mencapai kecepatan tertentu. Misalnya, jika robot bergerak dengan kecepatan 1 m/s dan motor maksimum dapat mencapai 2 m/s, dapat menghitung siklus tugas untuk masing-masing roda. Kemudian, menggunakan Arduino atau mikrokontroller, mengatur sinyal PWM untuk setiap roda berdasarkan perhitungan tersebut. Dengan cara ini, robot dapat mengontrol kecepatannya secara efektif dan merespons lingkungan sekitarnya dengan baik.

BAB III
PERANCANGAN SISTEM

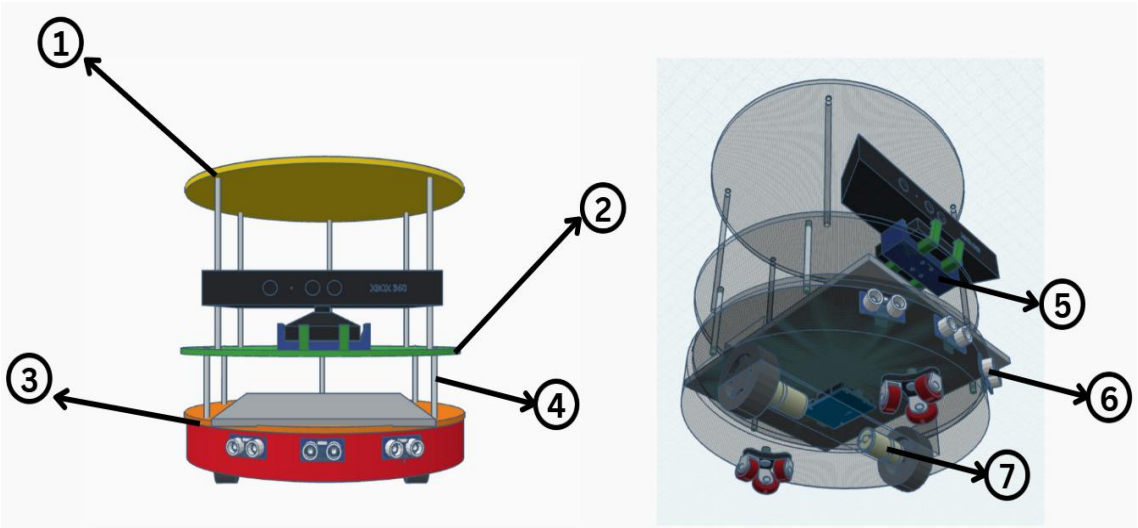
3.1. Metodelogi Penelitian

Pada bab ini membahas perihal metode penelitian yang akan digunakan untuk tugas akhir. Diantaranya melakukan studi literatur dengan menggunakan landasan teori serta jurnal yang ada, melakukan bimbingan bersama dosen pembimbing, merancang alat, menghubungkan dan memilah rangkaian masing-masing komponen sesuai kegunaannya, dan melakukan analisis terhadap alat yang akan dibuat.

3.2. Perancangan Desain Alat

Robot beroda dirancang untuk beroperasi di lingkungan *indoor*, memanfaatkan *Kinect* sebagai sensor utama untuk mendeteksi lingkungan melalui *point cloud*. Data kedalaman dari *Kinect* memungkinkan robot memetakan area sekitarnya secara 3D dan mendeteksi rintangan, baik yang statis maupun bergerak, termasuk manusia. Selain itu, informasi RGB digunakan untuk mengenali objek target yang relevan, memastikan robot bergerak ke arah tujuan yang benar.

Desain robot memungkinkan penambahan komponen lain untuk pengembangan lebih lanjut atau untuk membuat robot lebih kecil sesuai kebutuhan penelitian. Berikut merupakan desain pada robot beroda yang ditujukan pada Gambar 3.1.

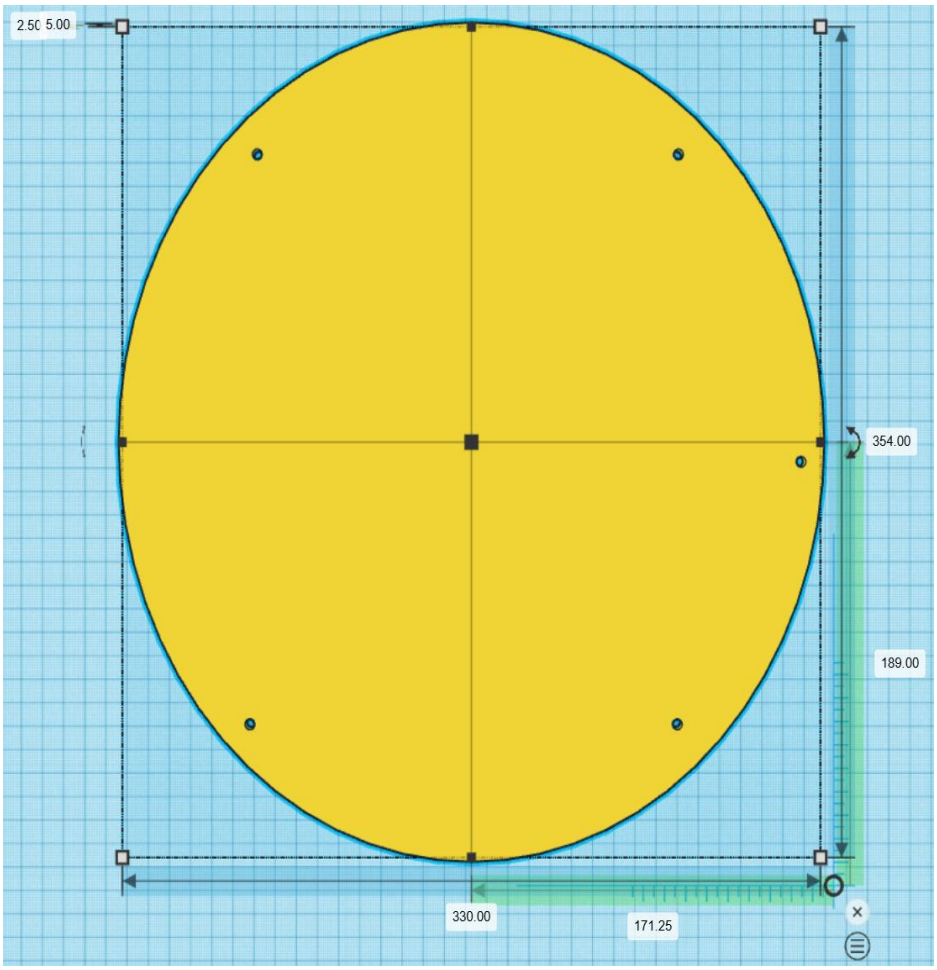


Gambar 3.1. Desain Robot Beroda Penghindar Manusia Menggunakan *Vector Field Histogram* Berbasis *Point Cloud*

Berdasarkan pada Gambar 3.1. berikut adalah penjelasan terkait komponen-komponen yang digunakan dalam proyek robot roda berbasis *Kinect* sebagai sensor utama untuk pemetaan berbasis *point cloud*:

1. Layer Atas

   Layer atas menggunakan material PVC (*Polyvinyl Chloride*) dengan ketebalan 5 mm. Material PVC dipilih karena sifatnya yang kokoh, tahan lama, dan mampu menahan beban komponen berat. Panjang dan lebar layer atas dengan panjang 354 mm dan lebar 330 mm sesuai dimensi lingkaran yang terlihat pada desain. Dengan ketebalan 5 mm PVC yang ditujukan pada Gambar 3.2.

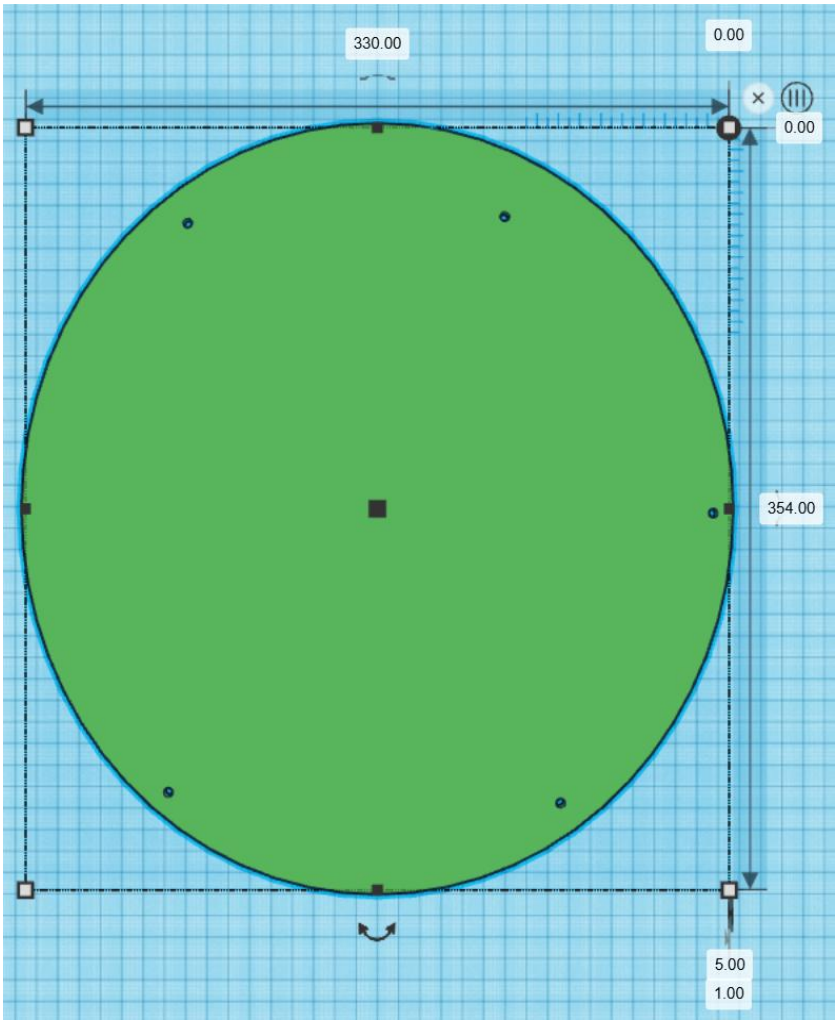


Gambar 3.2. Layer Atas Dengan Material PLA

2. Layer Tengah

   Lapisan layer tengah memiliki ketebalan 5 mm dan ukuran yang serupa dengan layer sebelumnya, yaitu panjang 354 mm dan lebar 330 mm. Material PVC dipilih karena kekuatannya yang lebih baik dibandingkan bahan lain, sehingga

cocok untuk menahan beban dan memberikan stabilitas tambahan pada struktur robot, pengunaan layer ini ditujukan sebagai menopang kamera *kinect*. Berikut merupakan layer tengah bermaterial PVC yang ditujukan pada Gambar 3.3.

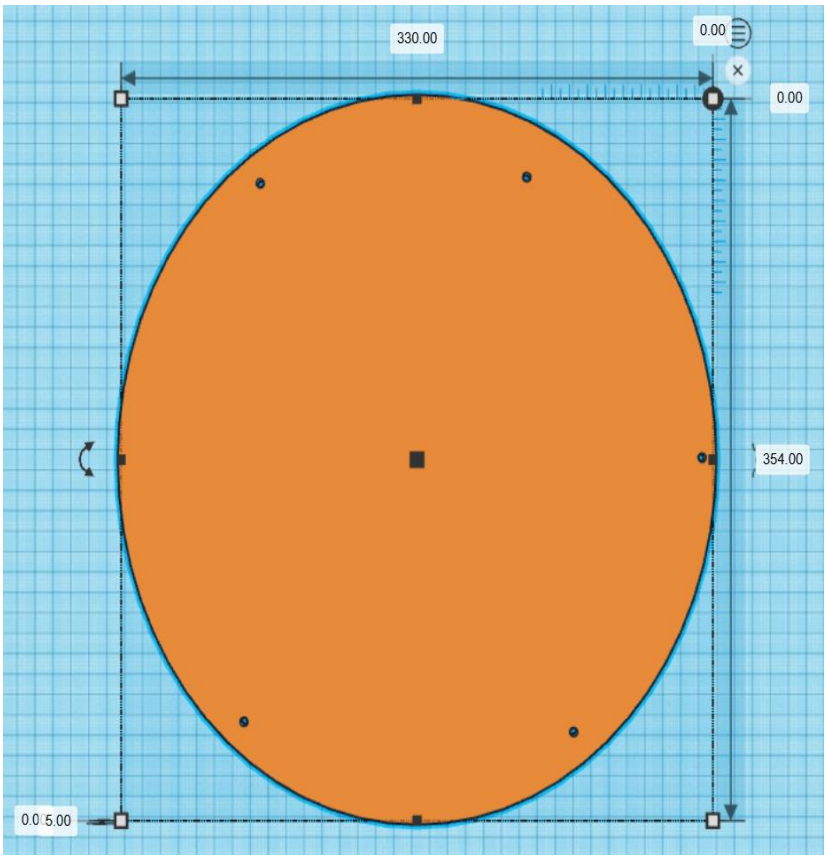


Gambar 3.3. Layer Tenggah Dengan Material PVC

3. Layer Bawah

Layer bawah pada desain ini berfungsi sebagai penyangga utama laptop sekaligus wadah untuk menempatkan berbagai komponen elektronik dan motor DC. Lapisan ini didesain menggunakan material PVC dengan ketebalan 5 mm, yang memberikan kekuatan struktural untuk menopang laptop secara aman. Pada bagian atas layer ini terdapat ruang datar yang dirancang khusus untuk menempatkan laptop, dengan dimensi yang memadai untuk memastikan stabilitas selama robot beroperasi. Material PVC yang digunakan tidak hanya kuat tetapi juga tahan terhadap tekanan dan beban, menjadikannya ideal untuk mendukung fungsi

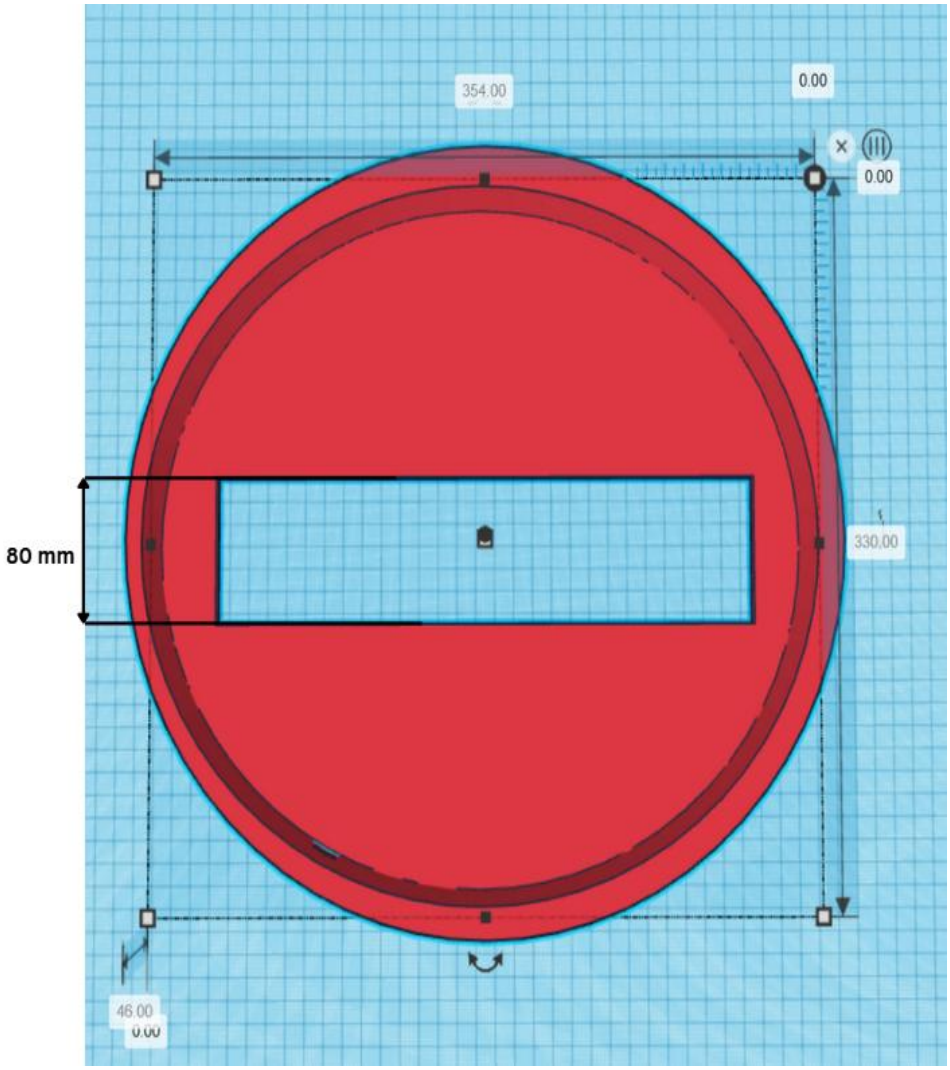
lapisan ini sebagai basis dari sistem robot. Berikut merupakan layer tengah bermaterial PVC yang ditujukan pada Gambar 3.4.



Gambar 3.4. Layer Bawah Dengan Material PVC

Pada bagian dalam layer bawah, dirancang sebuah ruang khusus yang berfungsi untuk menampung berbagai komponen elektronik penting, seperti *microcontroller*, *driver motor*, serta modul tambahan lainnya yang mendukung pengoperasian robot. Ruang ini dibuat dengan mempertimbangkan efisiensi tata letak dan aksesibilitas untuk memudahkan instalasi maupun perawatan komponen. Layer bawah ini tidak hanya berfungsi sebagai penyangga mekanis untuk layer atas yang menopang laptop, tetapi juga sebagai wadah internal yang menjaga agar seluruh komponen elektronik terlindungi dengan baik dari benturan atau kerusakan eksternal. Selain itu, desain layer bawah mengutamakan kombinasi kompak dan fungsionalitas yang memungkinkan semua komponen terorganisasi dengan rapi tanpa mengurangi kestabilan struktur. Lapisan ini juga memberikan ruang yang cukup untuk pemasangan motor DC dan mekanisme penggerak roda, termasuk

jalur keluar bagi poros roda agar dapat terhubung dengan baik. Dengan perancangan ini, layer bawah berkontribusi secara signifikan terhadap keseimbangan antara fungsi mekanis dan elektronik, memastikan stabilitas robot sekaligus mendukung performa keseluruhan sistem secara maksimal. Material PVC yang digunakan pada layer ini memberikan keunggulan dari segi daya tahan, ringan, dan kemudahan modifikasi, seperti yang ditunjukkan pada Gambar 3.5.

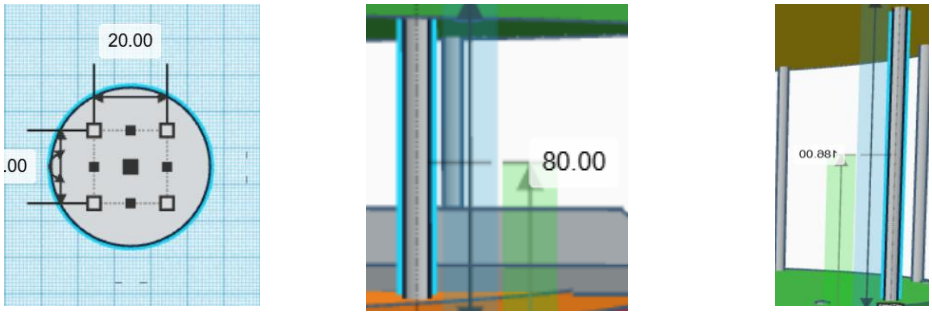


Gambar 3.5. Pangkon DC *Encoder* Dengan Material PLA

4. Penopang Layer

PLA dipilih karena kemudahan pencetakan dan bentuk kompleks yang dapat dibentuk dengan presisi. PLA cocok untuk bagian ini karena tidak membutuhkan ketahanan mekanik yang sangat tinggi namun tetap ringan dan mudah dibentuk. Berikut Penopang layer bermaterial PLA dengan jarak ketinggian

penopang layer bawah pada layer tengah dengan tinggi 80 mm dan layer tengah pada layer atas tinggi 180 mm yang tunjukan pada Gambar 3.6.

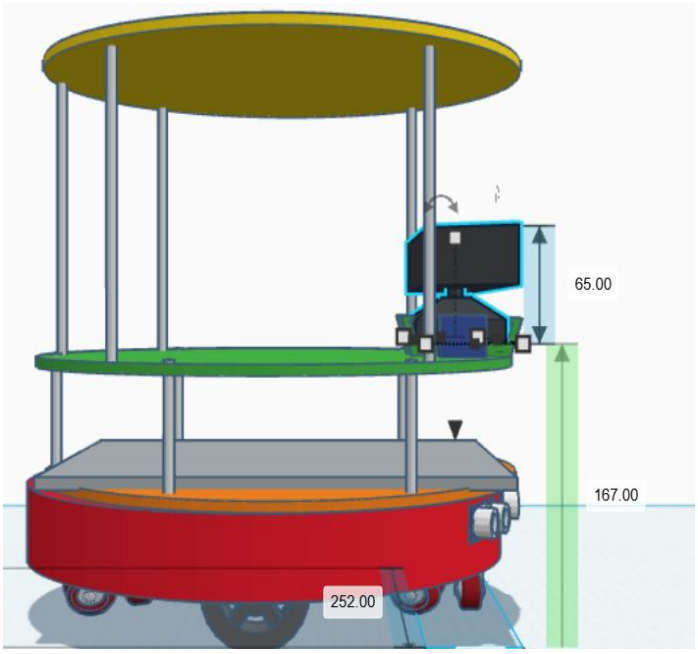


(a) (b) (c)

Gambar 3.6. (a)Penopang layer (b) Jarak Bawah Pada Tengah, dan (c) Jarak Tengah Pada Atas

5. Posisi *Kinect*

Posisi *kinect* terletak di tingkat tengah pada desain robot dengan jarak 167 mm, *kinect* digunakan untuk menghasilkan data *point cloud* dan pengenalan target. Penempatannya strategis untuk memperoleh cakupan area pandang yang optimal, baik untuk penghindaran. Jarak posisi kamera *kinect* robot yang ditampilkan pada Gambar 3.7.



Gambar 3.7. Jarak posisi kamera *kinect* robot

6. Sensor

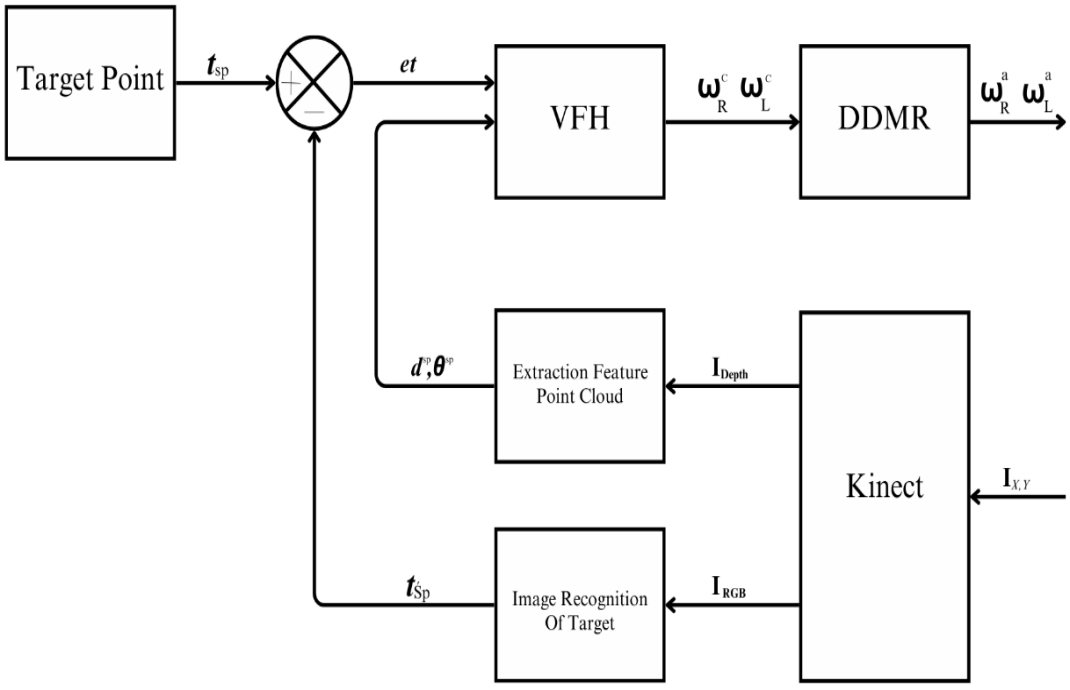
Sensor berfungsi sebagai *emergency* dalam mendeteksi lingkungan sekitar robot. Sensor yang digunakan berjenis *ultrasonic.* Dengan menggunakan sensor ini, robot dapat mengidentifikasi keberadaan rintangan atau manusia yang ada di jalurnya, sehingga dapat mengambil keputusan untuk menghindari atau mengubah arah jika kamera *kinect* tidak dapat mendeteksi rintangan dalam jarak dekat yang berada pada posisi layer ketiga robot.

7. Akuator

Aktuator berupa motor DC dengan encoder yang digunakan untuk mendukung sistem pengendalian pergerakan yang lebih presisi. Motor ini bekerja dengan mengonversi energi listrik dari sumber daya menjadi energi mekanik dalam bentuk rotasi roda, sehingga memungkinkan robot untuk bergerak maju, mundur, dan berbelok sesuai dengan perintah yang diberikan.

3.3. Diagram Blok Sistem

Berikut merupakan diagram blok pada penelitian robot beroda penghindar manusia menggunakan *vector field histogram* berbasis *point cloud* yang ditunjukan pada Gambar 3.8.



Gambar 3.8. Diagram Blok Sistem

Pada " Robot Beroda Penghindar Manusia Menggunakan *Vector Field Histogram* Berbasis *Point Cloud*," *input, prosses* dan *output*-nya dapat dijabarkan sebagai berikut:

1. *Input*

   Pada *input* mengunakan *target point* berupa *traffic cone* yang merupakan titik tujuan *finish* dimana robot beroda akan pergi pada titik target, informasi tersebut diterapkan sebagai target *set-point* (T_{sp}) yaitu koordinat atau posisi tujuan.

   Dengan dua data yang digunakan pada kamera *kinect* yang berupa data warna (I_{RGB}) sebagai penganalan target dan data kedalam (I_{Depth}) agar membentuk *point cloud.*

2. Proses

   Pada proses mengunakan *image recognition of target* dengan mengenali data pada (I_{RGB}) agar robot dapat pergi ke titik tujuan. *Extraction fiture point cloud* digunakan dari data (I_{Depth}) agar mengenali lingkungan dan *object* pada sekitar robot beroda. Informasi yang dihasilkan berupa jarak dan sudut *traffic cone* (d^{sp}, θ^{sp}) yang digunakan sebagai data *vector field histogram* (VFH). *Error tresshold* (*et*) merupakan sistem pembanding posisi *current* pada robot beroda dengan target *set-points* (T_{sp}), jika terdapat perbedaan, akan dihitung sebagai *error tresshold* (*et*) yang menjadi umpan balik untuk pergerakan robot beroda.

   Vector field histogram (VFH) metode menghindar rintangan bedasarkan informasi jarak dan sudut dari *point cloud.* VFH menghasilkan *command* roda kiri (ω_L^c) dan roda kanan (ω_R^c). DDMR merupakan model *differential* yang digunakan robot beroda serta mengkonversi metode VFH menjadi pergerakan *actual* dengan *output* (ω_L^a) dan (ω_R^a)

3. *Output*

   Gerak robot beroda dirancang untuk memastikan kemampuan adaptasi terhadap rintangan di jalur pergerakannya. Orientasi robot akan diatur berdasarkan command yang diproses oleh sistem, yang memperhitungkan berbagai parameter lingkungan. Apabila robot mendeteksi adanya rintangan pada jalur laju, misalnya keberadaan manusia dalam studi kasus ini, maka sistem akan secara otomatis menyesuaikan lintasan geraknya untuk menghindari rintangan tersebut. Penyesuaian ini dilakukan tanpa mengabaikan tujuan utama robot, yaitu

mencapai target set-point yang telah ditentukan. Sistem yang diterapkan tidak hanya memastikan efisiensi navigasi, tetapi juga mempertahankan aspek keselamatan dan keakuratan dalam lingkungan indoor.

3.4. Perancangan Elektronika

Pada bagian perancangan elektronika dalam proyek robot beroda penghindar manusia, mencakup beberapa aspek sebagai berikut:

1. Mikrokontroler

Mikrokontroler yang digunakan adalah Arduino R4 WiFi, yang berfungsi sebagai otak dari sistem robot. Arduino R4 WiFi dipilih karena kemampuannya dalam mengendalikan berbagai komponen elektronik serta kemampuan komunikasi nirkabelnya. Mikrokontroler ini akan mengontrol aktuator, seperti motor DC dengan *encoder*, yang memungkinkan robot untuk bergerak dengan presisi dan akurasi tinggi. Berikut ini merupakan Arduino R4 Wifi yang ditunjukan pada Gambar 3.10. dan spesifikasi pada Arduino R4 Wifi yang ditunjukan pada Tabel 3.1.

Gambar 3.9. Arduino R4 Wifi

Tabel 3.1. Spesifikasi Arduino R4 Wifi

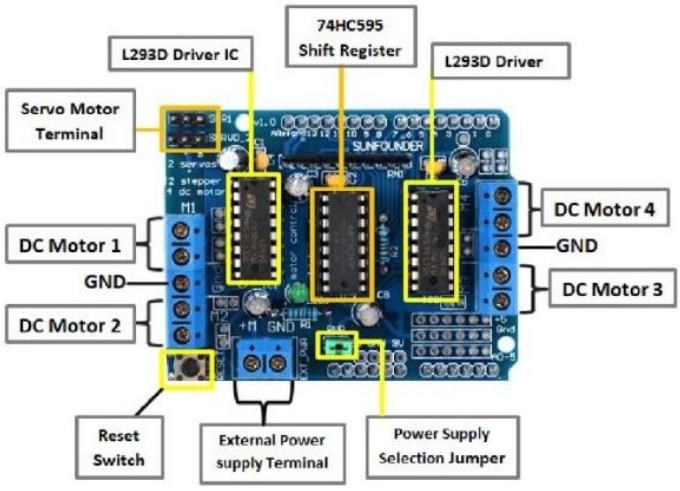
komponen	*Type*
Mikroprosesor	Renesas RA4M1 (Arm Cortex-M4) ESP32-S3
Memory	RA4M1: 256 KB Flash, 32KB SRAM ESP32-S3: 384 KB ROM, 512 KB SRAM
Tegangan operasi	5VDC (ESP32-S3 3.3VDC)
Rentang tegangan input	6-24VDC

komponen	*Type*
Arus DC per I/O Pin	8m
Clock speed	Main core: 48MHz ESP32-S3: up to 240MHz
Port pemograman	USB-C
Pin Digital	14 pin
Pin Analog	6 pin
PWM	6 pin
Antarmuka	UART, I2C, SPI, CAN Bus
Dimensi	68.85 x 53.34mm

2. Driver Motor Arduino *Shield* L293D

Driver motor yang digunakan dalam proyek ini adalah Arduino *Shield* L293D. Motor driver yang dirancang untuk mengendalikan dua motor DC secara bersamaan. Dengan menggunakan *shield* ini, Arduino R4 WiFi dapat mengatur arah, kecepatan, dan kontrol motor dengan sangat efektif. Hal ini memungkinkan robot bergerak dengan responsivitas yang tinggi, memberikan kontrol yang halus dan presisi dalam mengikuti jalur atau menghindari rintangan.

Selain itu, L293D juga dilengkapi dengan fitur proteksi terhadap suhu berlebih dan arus lebih yang dapat mencegah kerusakan pada driver motor. Tampilan dari driver motor Arduino *Shield* L293D pada Gambar 3.10



Gambar 3.10. Driver motor Arduino *Shield L293D*

3. Motor DC dengan *Encoder*

Motor DC PG36 dipilih untuk proyek ini karena kemampuannya dalam memberikan kontrol yang tepat dan responsif terhadap pergerakan robot. Motor ini dilengkapi dengan *encoder*, yang berfungsi untuk memberikan umpan balik mengenai posisi dan kecepatan rotasi motor. Berikut merupakan tamplan Motor DC PG36 yang ditunjukan pada Gambar 3.11.

Gambar 3.11. Motor DC PG36

Dengan spesifikasi yang mendukung, motor PG36 mampu memberikan torsi yang cukup untuk menggerakkan robot meskipun dalam kondisi beban yang bervariasi. Menunjukkan spesifikasi teknis dari motor DC PG36 pada Tabel 3.2

Tabel 3.2. Spesifikasi Motor DC *Encoder* PG36

Parameter	Daya
Daya	24V
Arus	0,3 A
Kecepatan	440 rpm
Torsi Gerak	11 kgfcm
Stall Torsi (Torsi Diam)	29 kgfcm
Panjang Gearbox	39.5mm

4. Baterai *Lithium Polymer* (LiPo) 3S dengan tegangan nominal 11.1V dan kapasitas 1800mAh

Baterai ini dipilih karena kemampuannya untuk memberikan daya yang tinggi dalam bentuk yang ringan dan kompak. Kapasitas 1800mAh memungkinkan penggunaan yang cukup lama tergantung pada kebutuhan daya dari motor DC dan komponen lainnya. Dengan rasio C (25-50C), baterai ini dapat memberikan arus maksimum hingga 90A (dari 1800mAh x 50C), yang cukup untuk mendukung dua motor DC berdaya tinggi. Berikut merupakan tampilan pada Gambar 3.12.

Gambar 3.12. Lipo Battery 3s 11.1v 1800mah 25-50c Onbo Power

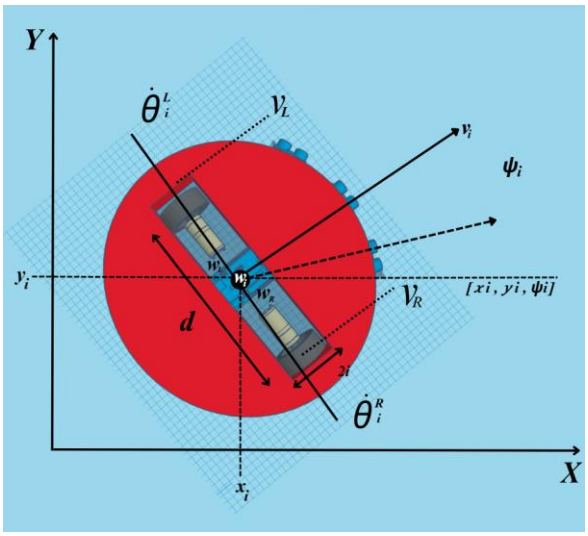
3.5. Perancangan Mekanikal

Pada perancangan mekanikal dari robot beroda yang digunakan dalam penelitian ini. Perancangan mekanikal mencakup struktur fisik robot, komponen yang digunakan, serta mekanisme pergerakan yang mendukung fungsi robot dalam melakukan navigasi dan penghindaran rintangan. Desain mekanikal dirancang untuk mendukung kestabilan, mobilitas, dan kemampuan *Kinect* dalam mendeteksi lingkungan sekitarnya. Sebagai berikut:

1. Model *Cartesian kinematika* robot beroda

   Dalam pemodelan kartesian, posisi robot dinyatakan dalam koordinat X dan Y, sedangkan orientasi ditentukan oleh sudut Ψ. Model memungkinkan robot untuk bergerak secara akurat dengan memperhitungkan kecepatan putar roda kiri

($\dot{\theta}_L$) dan kanan ($\dot{\theta}_R$). Berikut merupakan pemodelan kartesian dan kinematika yang digunakan, pada Gambar 3.13.



Gambar 3.13. Model Cartesian dan Kinematika Robot Beroda

Berikut keterangan persamaan yang digunakan pada model kinematika yang ditunjukan pada persamaan 3.1, persamaan 3.2, dan persamaaan 3.3.

$$Pi = [xi\ ,\ yi\ ,\ \psi i\ ] \tag{3.1.}$$

$$\dot{P}\ i = [\dot{x}i\ ,\ \dot{y}i\ ,\ \dot{\psi}i\ ] \tag{3.2.}$$

$$\dot{q}i = [\dot{\theta}_L^i\ ,\ \dot{\theta}_R^i] \tag{3.3.}$$

Di mana posisi robot *actual* robot *Pi* dilambangkan dengan xi dan yi dan orientasi dilambangkan dengan ψi, $\dot{q}i$ dilambangkan sebagai kecepatan roda robot Transisi konfigurasi robot dari base ke koordinat $\dot{P}i$ sebagai persamaan 3.4

$$\begin{bmatrix} \dot{x}i \\ \dot{y}i \\ \dot{\psi}i \end{bmatrix} = \begin{vmatrix} \cos\psi i & 0 \\ \sin\psi i & 0 \\ 1 & 0 \end{vmatrix} = \begin{bmatrix} vi \\ \omega i \end{bmatrix} \tag{3.4.}$$

Perubahan posisi mengunakan *Forward kinematic* dengan kecepatan linier (vi) dan kecepatan angular (ωi) robot berdasarkan kecepatan roda kiri ($\dot{\theta}_L^i$) dan kecepatan roda kanan ($\dot{\theta}_R^i$) robot beroda ddmr direpresentasikan persamaan 3.5

$$\begin{bmatrix} vi \\ \omega i \end{bmatrix} = \begin{vmatrix} 1/2 & 1/2 \\ 1/d & 1/-d \end{vmatrix} = \begin{bmatrix} \dot{\theta}_i^L \\ \dot{\theta}_i^R \end{bmatrix} \tag{3.5.}$$

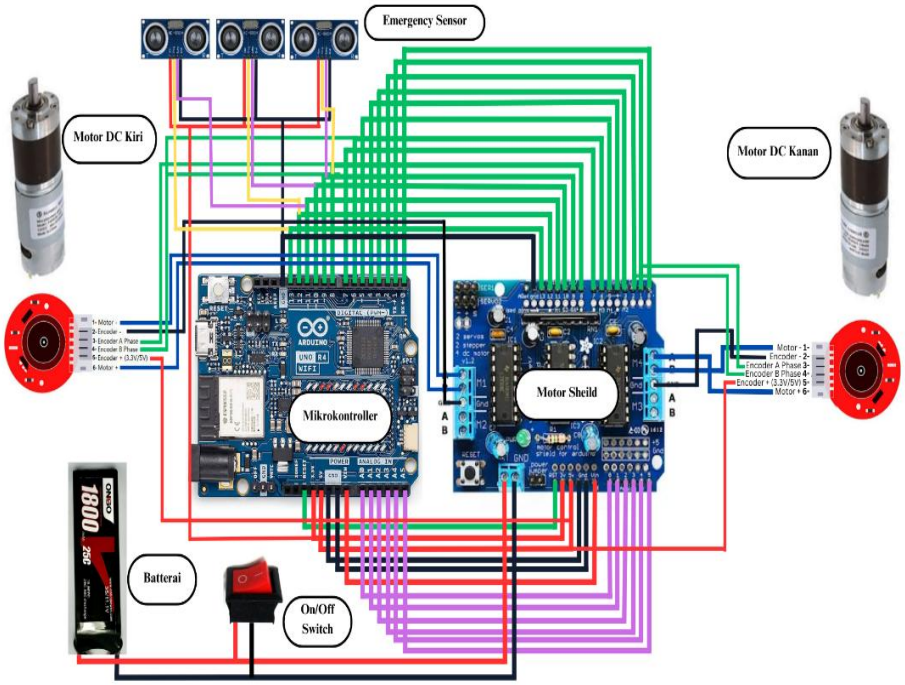
Perubahan posisi mengunakan *Inverse kinematic* dengan menggunakan kecepatan roda robot ($\dot{q}i$) pada kecepatan linear dan angular direpresentasikan sebagai

persamaan 3.6

$$\begin{bmatrix} \dot{\theta}_L^i \\ \dot{\theta}_R^i \end{bmatrix} = \frac{1}{i} \begin{vmatrix} 1 & d/2 \\ 1 & -d/2 \end{vmatrix} = \begin{bmatrix} vi \\ \omega i \end{bmatrix} \quad (3.6.)$$

2. Skema Rangkaian Elektronika

Skema rangkaian ini dirancang untuk memastikan bahwa semua komponen dapat bekerja dengan sinergi. Berikut adalah komponen utama yang digunakan dalam sistem elektronika robot yang ditunjukan pada Gambar 3.14.



Gambar 3.14. Skema Rangkaian Elektronika

Penjelasan berikut mencakup fungsi masing-masing pin dan penghubungannya pada arsitektur rangkaian elektronika. Sehingga memudahkan pemahaman tentang interaksi antar komponen dalam rangkaian:

a. Motor 1

Terhubung ke motor *driver shield* (Motor + dan Motor -) Terhubung ke terminal motor di *shield* pada label M1. (*Encoder* A Phase) dan (*Encoder* B Phase) Terhubung ke pin digital di Arduino untuk membaca data *encoder* untuk posisi dan kecepatan motor. (*Encoder* +:

Terhubung ke pin 5V di Arduino. (*Encoder* -) Terhubung ke *ground* (GND*).*

b. Motor 2

Terhubung dengan cara yang sama, dengan (Motor + dan Motor-) Terhubung ke terminal motor di *shield* pada label M2. (*Encoder* A Phase) dan (*Encoder* B Phase) Terhubung ke pin digital di Arduino. (*Encoder* +) Terhubung ke pin 5V di Arduino. (*Encoder* **-)** Terhubung ke *ground* (GND).

c. Koneksi Arduino ke Motor Driver Shield

Power Supply shield motor menerima daya dari baterai melalui pin VCC. Pin Kontrol Motor 1 dan Motor 2 dikontrol oleh PWM dan pin digital dari Arduino (pin 8 dan 7 untuk Motor 1) serta (pin 2 dan pin3) untuk Motor 2.

d. Baterai

Baterai *Lithium Polymer* (LiPo) 3S dengan tegangan nominal 11.1V dan kapasitas 1800mAh sumber daya utama untuk Arduino dan motor *driver shield* pada VCC dan GND.

e. *Kinect*

Terhubung ke laptop melalui USB, yang kemudian berfungsi sebagai pengolah data dan komunikasi dengan Arduino melalui jaringan *Serial Port*.

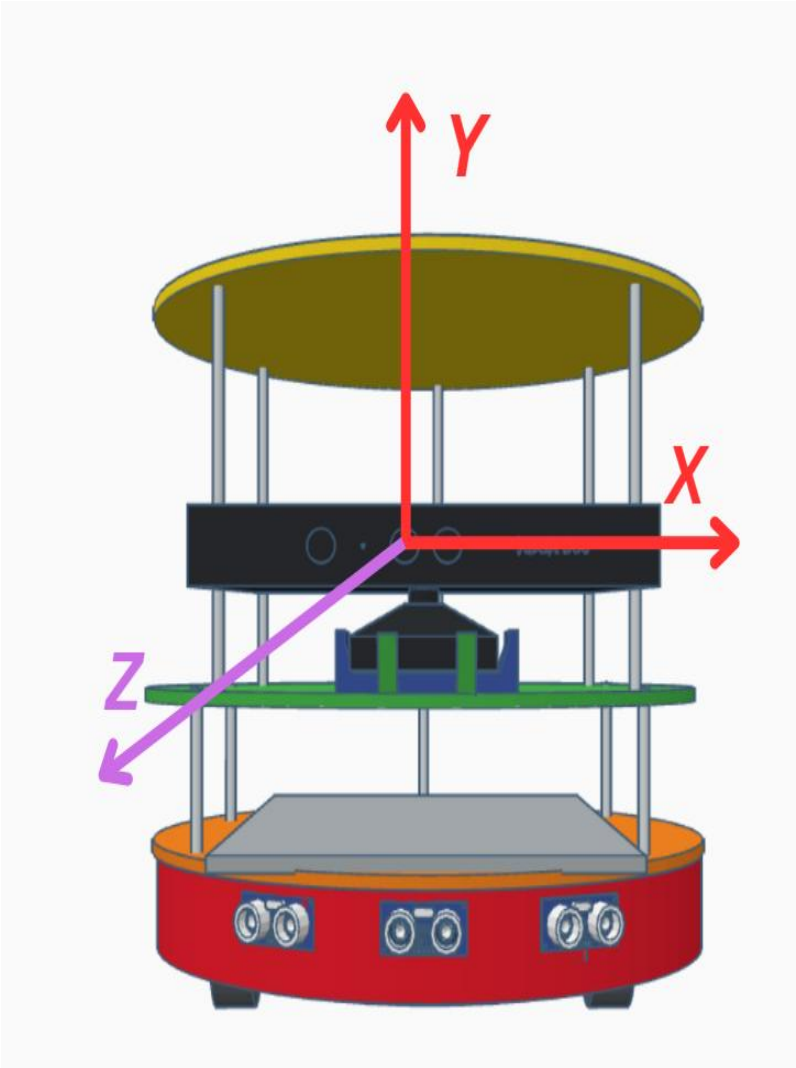
f. Sensor ultrasonik

3 buah sensor ultrasonik digunakan sebagai *emergency* sensor bila *obstacle* atau halangan manusia pada FoV kamera *kinect* terdeteksi tiba-tiba tanpa melalui proses metode VFH yang mungkin menganggu kinerja penghindaran, Dengan menonaktifkan sementara kecepatan kedua roda hingga semua sensor tidak mendeteksi pada jarak 0,10 millimeter dari oreantasi robot (90 derajat)

3. Model *Cartesian* pada kamera *Kinect*

Koordinat kamera dalam ruang 3D, yang terdiri dari tiga sumbu: X, Y, dan Z. Kamera *kinect* menangkap *point cloud* dengan mengukur jarak antara objek di

ruang nyata dan sensor, serta menggunakan data visual dan kedalaman untuk menghasilkan representasi 3D. Yang ditunjukan pada Gambar 3.15.



Gambar 3.15. Model *Cartesian* pada Kamera *Kinect*

Kamera *Kinect* dipasang pada robot, dengan transformasi koordinat dari tik penglihatan kamera ke titik penglihatan robot sebagai berikut:

a. Rotasi

Menggunakan matriks rotasi 3x3 untuk orientasi kamera terhadap robot.

b. Translasi

Menggunakan *vektor* translasi untuk menentukan *offset* posisi kamera dari pusat robot.

Setiap titik dalam *Point Cloud* yang dihasilkan oleh *Kinect* akan berada pada posisi tertentu dalam sistem koordinat 3D, dengan nilai x, y, dan z yang

menggambarkan posisi objek relatif terhadap sensor. Dengan representasi transformasi *cartesian.* Yang ditunjukan pada persamaan 3.7

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + T \tag{3.7}$$

Di mana:

R adalah matriks rotasi 3x3.

T adalah *vektor* translasi.

(x, y, z) adalah koordinat asli dalam sistem kamera.

(x′, y′, z′) adalah koordinat dalam sistem robot.

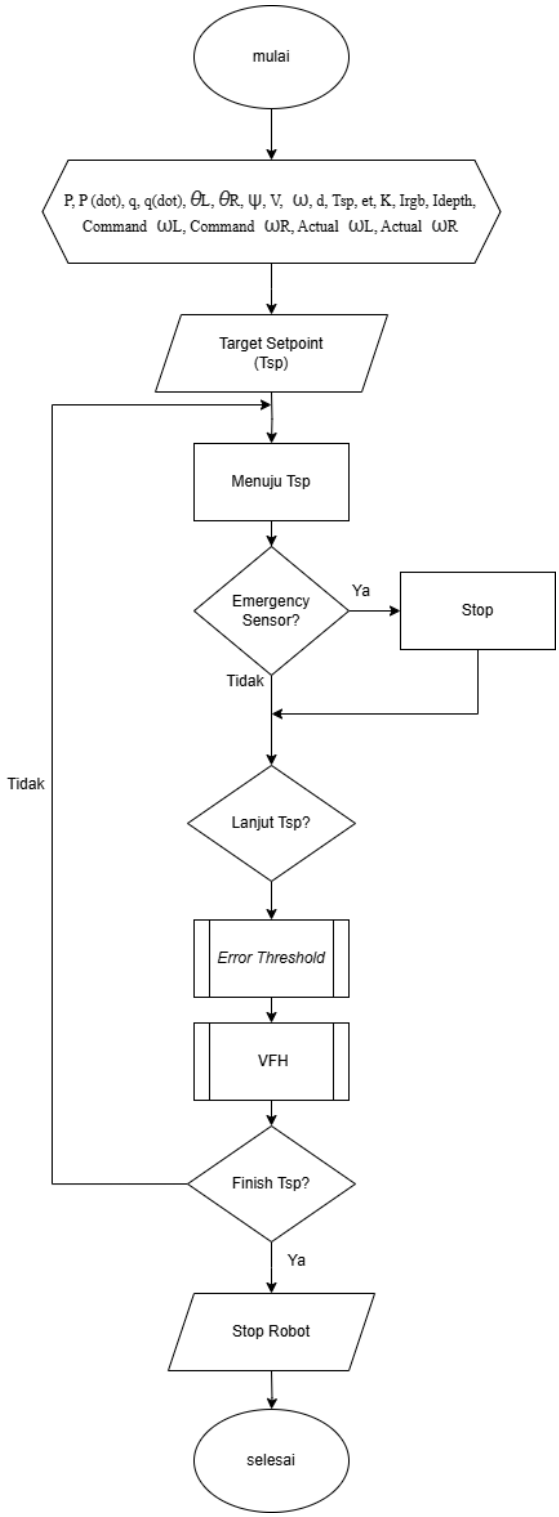
3.6. *Flowchart* Sistem

Skenario operasional sistem robot beroda untuk mencapai tujuan utama, yaitu robot pergi ke titik target *set-point* dimana robot dapat menghindari manusia sebagai rintangan dalam lingkungan *indoor*. Sistem robot beroda yang dirancang untuk bergerak menuju titik target *set-point* (Tsp) sambil memastikan navigasi yang aman di lingkungan *indoor*. Sistem ini memiliki tujuan utama untuk memungkinkan robot bergerak secara efisien menuju target dengan menghindari manusia dan rintangan lain yang terdeteksi di jalurnya. Pendekatan yang digunakan dalam sistem ini adalah *Vector Field Histogram* (VFH), sebuah metode penghindaran rintangan real-time yang efektif dalam menganalisis distribusi rintangan di sekitar robot dan menentukan jalur aman.

Proses dimulai dengan inisialisasi parameter-parameter sistem seperti posisi aktual robot (P), posisi target (Tsp), kecepatan roda kiri (ωL) dan roda kanan (ωR), serta nilai sensor kedalaman (I*rgb* dan I*depth*) yang berasal dari data *kinect*. Parameter ini berfungsi sebagai masukan utama dalam menentukan pergerakan robot. Setelah parameter diinisialisasi, sistem menentukan Target *Setpoint* (Tsp) yang menjadi tujuan pergerakan robot. Robot kemudian bergerak menuju set-point dengan terus memantau kondisi lingkungan menggunakan sensor darurat yang bertugas mendeteksi rintangan atau keadaan kritis yang memerlukan pemberhentian sistem.

Jika sensor darurat mendeteksi situasi berbahaya, sistem akan menghentikan operasi robot untuk menghindari risiko tabrakan. Namun, jika tidak ada gangguan yang terdeteksi, sistem akan melanjutkan proses pergerakan robot. Pada tahap ini, sistem mengevaluasi *Error Threshold* untuk memastikan bahwa pergerakan robot menuju target

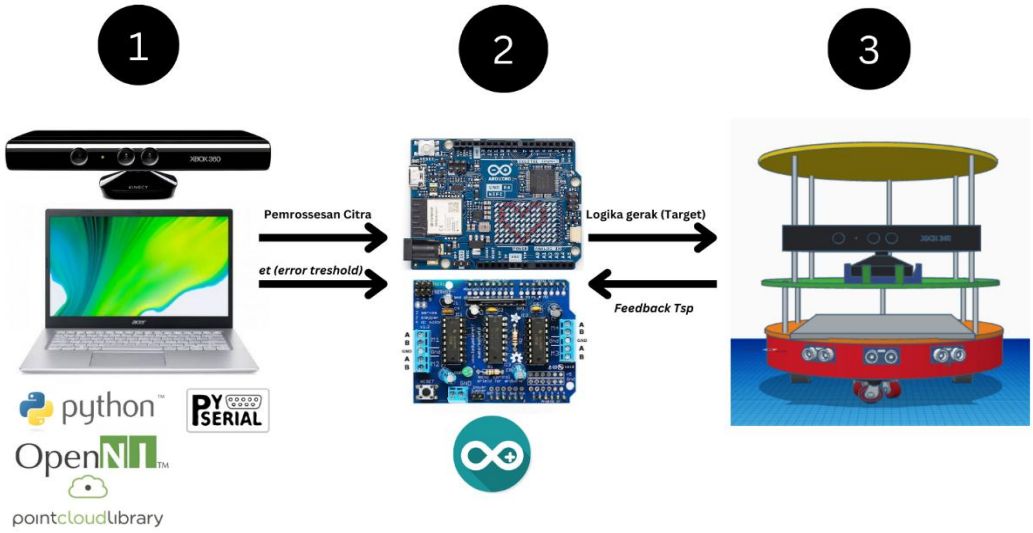
berjalan sesuai perhitungan. *Error Threshold* mengacu pada nilai toleransi kesalahan antara posisi target dan posisi aktual robot, di mana jika kesalahan melebihi ambang batas, sistem akan melakukan koreksi untuk mengarahkan robot kembali ke jalur yang benar. Berikut merupakan *flowchart* sistem yang di tunjukan pada Gambar 3.16.



Gambar 3.16. *Flowchart* Sistem

3.7. Perancangan Metode

Perancangan metode meliputi beberapa tahap, mulai dari pemrosesan data hingga kontrol motor yang mendukung navigasi robot. Setiap komponen perangkat lunak diintegrasikan untuk memastikan sinkronisasi antara pemrossesan citra, pengambilan keputusan, dan eksekusi gerakan robot secara *real-time*. Berikut merupakan visualisasi perancangan metode yang ditunjukan pada Gambar 3.17.



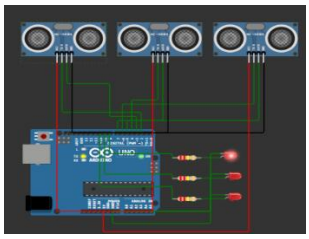
Gambar 3.17. Visualisasi Perancangan Metode

Pada Visualisasi Perancangan Metode. Menunjukkan metode perancangan implementasi robot beroda yang menggunakan sensor *kinect*. Berikut merupakan penjelasan yang digunakan dalam rancangan skenario

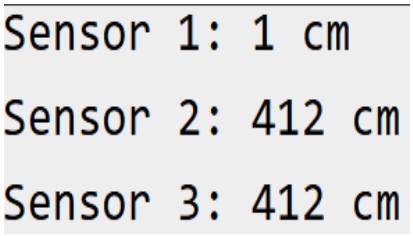
3.7.1. Pemrossesan Citra

Pada Gambar 3.16. yang menunjukan area (1). Sensor pada kamera *kinect* digunakan untuk menangkap data visual lingkungan, termasuk data kedalaman (I_{Depth}) dan warna (I_{RGB}). Data dari *Kinect* diproses pada komputer menggunakan Bahasa pemrograman Python dengan pustaka Opencv, OpenNI, PySerial, dan *Point Cloud Library* (PCL). Metode VFH digunakan dengan *input* data pada data kedalaman (I_{Depth}) yang diproses dengan *library Point Cloud Library* (PCL). Sehingga hambatan laju dapat dideteksi oleh robot beroda dan akan diproses dengan *polar histogram.* Proses tersebut menghasilkan informasi tentang lingkungan yang diperlukan sebagai navigasi. Data yang telah diproses, dikirim ke *serial port* untuk menentukan pergerakan robot.

Emergency sensor merupakan sistem yang dirancang untuk mendeteksi kondisi darurat yang dapat menghambat pergerakan atau membahayakan robot selama operasi. Sensor ini berfungsi agar robot dapat berhenti mengambil tindakan penghindaran dengan cepat untuk mencegah tabrakan mengunakan tiga buah sensor ultrasonic pada jarak 10 mm atau 1cm. yang ditunjukan pada Gambar 3.18.



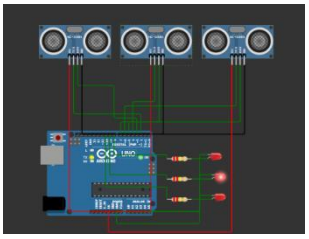
(a)



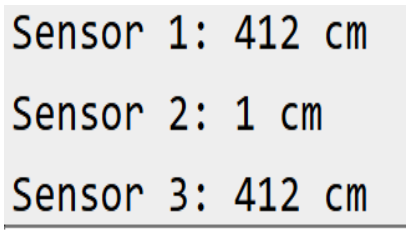
(b)

Gambar 3. 18. (a) *Emergency* Sensor 1 Terdeteksi Pada Jarak 1 cm (b) *Serial Monitor*

Penempatan *emergency* sensor 2 pada posisi 90 derajat terhadap arah pergerakan robot, yang terletak di layer bawah robot. Sensor ini berfungsi untuk mendeteksi objek atau hambatan yang berada di sisi tenggah, yang ditunjukan pada Gambar 3.19.



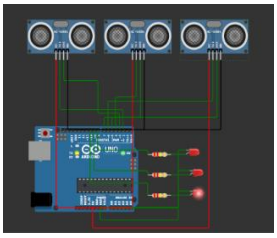
(a)



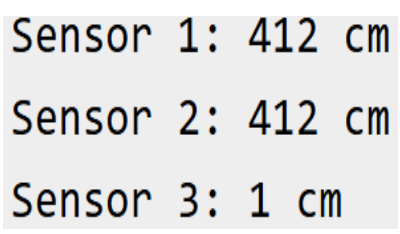
(b)

Gambar 3.19. (a) *Emergency* Sensor *2* Terdeteksi Pada Jarak 1 cm (b) Serial Monitor

Penempatan *emergency* sensor 3 pada posisi 135 derajat terhadap arah pergerakan robot, yang terletak di layer bawah robot. Sensor ini berfungsi untuk mendeteksi objek atau hambatan yang berada di sisi kanan, yang ditunjukan pada Gambar 3.20.



(a)



(b)

Gambar 3.20. (a) *Emergency* Sensor 3 Terdeteksi Pada Jarak 1 cm (b) Serial Monitor

Setelah *traffic cone* dikenali menggunakan *Haar Cascade*, posisi *cone* dapat diproyeksikan ke dalam koordinat 2D dimensi menggunakan data warna (I_{RGB}) dari *kinect*. Proses dimulai dengan menentukan titik pusat dari *bounding box Haar Cascade* dalam citra 2D, lalu mengonversi titik tersebut ke koordinat 3D. Berikut ini merupakan penerapan *Haar Cascade yang* telah dilatih pada Gambar 3.21.

(a) (b)

Gambar 3. 21. (a) *Output* Antarmuka Penerapan *Haar Cascade* Yang Telah Dilatih (b) Jarak Posisi Kamera *Kinect* Secara *Real*

Dengan data ini, jarak *Euclidean* antara robot dan *cone* dapat dihitung untuk mengetahui seberapa jauh *cone* berada. Berikut merupakan perhitungan yang digunakan untuk menentukan jarak posisi *target set-point* yang ditunjukan pada persamaan 3.8.

$$X_{Tsp} = x + \frac{w}{2} \ , Y_{Tsp} = y + \frac{h}{2} \ , Z_{Tsp} \tag{3.8}$$

Di mana:

x dan y adalah koordinat dari *bounding box Haar Cascade* yang terdeteksi.

w dan h adalah lebar dan tinggi *bounding box Haar Cascade*.

Nilai Z_{Tsp} diambil dari data kedalaman (I_{Depth}) yang diukur oleh sensor *kinect* pada titik (X_{Tsp}, Y_{Tsp}). Data kedalaman ini memberikan informasi jarak *target set-point* dari kamera dalam satuan meter.

3.7.2. *Error Treshold*

Error threshold berfungsi untuk menentukan kapan robot perlu melakukan koreksi atau tidak. Hal ini bertujuan agar robot tidak bereaksi terhadap perbedaan kecil yang tidak signifikan, seperti *noise* dari sensor atau fluktuasi kecil yang tidak memengaruhi jalur pergerakan secara keseluruhan. Rumus yang digunakan untuk

menghitung nilai *error* dan membandingkannya dengan nilai *threshold* adalah pada persamaan 3.9

$$E = T_{sp} - P \tag{3.9}$$

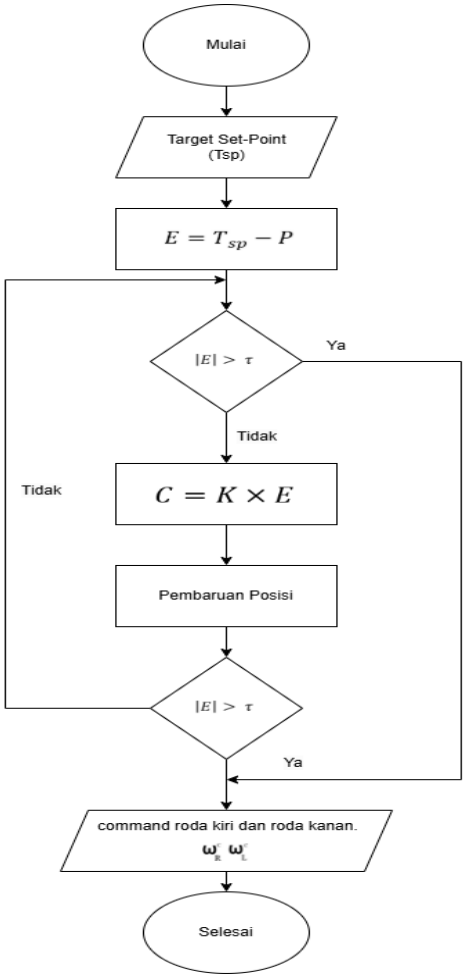
Dengan *E* sebagai *error*, T_{sp} sebagai target *set-point* (posisi target), dan P sebagai posisi aktual robot yang ditunjukan pada persamaan 3.10.

$$|E| > \tau \tag{3.10}$$

Dimana τ adalah nilai *threshold*. Jika $|E| > \tau$, koreksi pada arah atau kecepatan dilakukan. Dan jika $|E| \leq \tau$, robot tetap bergerak tanpa koreksi tambahan yang ditunjukan pada persamaan 3.11.

$$C = K \times E \tag{3.11}$$

Dengan *C* sebagai nilai koreksi dan *K* sebagai konstanta dan *E* sebagai nilai *error* antara posisi target dengan posisi aktual. Berikut merupakan *flowchart* pada *error threshold* yang ditunjukan pada Gambar 3.22.



Gambar 3.22. *Flowchart* pada *Error Threshold*

3.7.3. Implementasi Metode *Vector Field Histogram* Sebagai Penghindar, Mengunakan Data *Point Cloud*

Metode *Vector Field Histog*ram (VFH) digunakan untuk navigasi dan penghindaran rintangan pada robot mobile. Dalam implementasi ini, VFH diterapkan untuk mengidentifikasi jalur bebas rintangan berdasarkan data *point cloud* yang diperoleh dari sensor *kinect*. Yang nantinya digunakan dalam analisis *histogram*. Berikut ini merupakan skenario *multi-thread processing,* yaitu:

1. Data *Point Cloud*

   Setiap titik dalam *point cloud* memiliki koordinat (x, y,z) di ruang 3D. Untuk mendeteksi visual jalur serta *obstacle*, dihitung dalam sistem koordinat kamera menggunakan persamaan 3.12

   $$X = \frac{(u-c_x).z}{f_x},\ Y = \frac{(u-c_y).z}{f_y} \text{ dan } Z = (u-v)z \quad (3.12)$$

   Dimana:

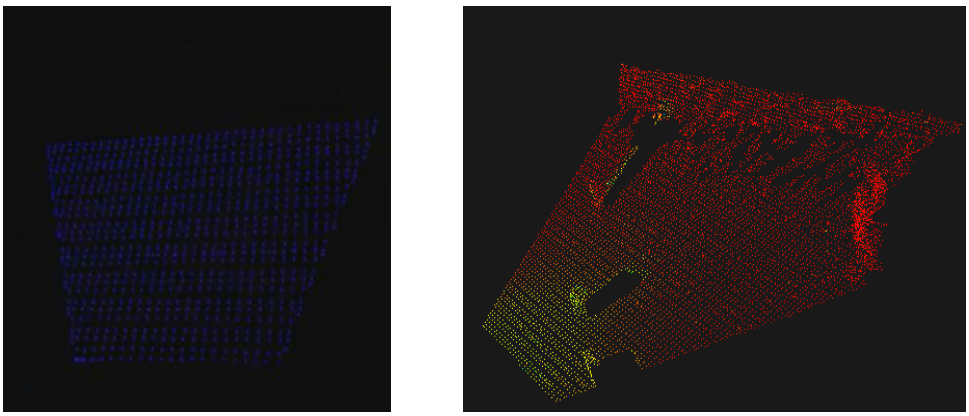
   u, v: Koordinat piksel pada gambar 2D.

   Z (u, v): Kedalaman piksel (dari *Depth Image*).

   c_x, c_y: Koordinat tengah kamera (*Principal point*).

   f_x, f_y: Panjang fokus kamera dalam piksel.

   Pada *sector low* menandakan bahwa jalur pada robot didefinisikan sebagai lintasan aman dan *sector high* akan didefinisikan bahwa terdapat halangan pada jalur robot berikut merupakan tampilan *sector low* dan *sector high* yang ditunjukan pada Gambar 3.23.



(a) (b)

Gambar 3.23. (a) *Raw Data Point Cloud Sector Low* (b) *Raw Data Point Cloud Sector High*

Threshold dalam proses navigasi robot, penting untuk mengklasifikasikan lingkungan di sekitarnya berdasarkan informasi dari sensor. Salah satu metode yang digunakan adalah dengan membedakan antara *sector low* dan *sector high*, yang ditentukan berdasarkan ambang batas tertentu. Klasifikasi ini membantu dalam mengidentifikasi jalur yang aman dan area yang mengandung rintangan yang perlu dihindari. Untuk menentukan apakah suatu titik berada di *sector low* atau *Sector High*. Berikut merupakan persamaan sektor *low* dan *high* yang ditunjukan pada persamaan 3.13 dan persamaan 3.14

y_i = ≤ 0.01 m pada z_i = ≤ 0.18 m. dan $|x_i - x_{robot}|$ (3.13)

Sector Low

y_i = > 0.01 m pada z_i = > 0.18 m. dan $|x_i - x_{robot}|$ (3.14)

Sector High

Setelah mengklasifikasikan *Sector High* berdasarkan kriteria tersebut, dapat menghitung kepadatan rintangan. Semakin tinggi dalam *Sector High*, Setelah sektor dalam *histogram polar* diklasifikasikan menjadi *Sector Low* dan *Sector High*, langkah selanjutnya adalah menganalisis kepadatan rintangan dalam *Sector High*. Kepadatan rintangan ini penting untuk menentukan jalur terbaik yang dapat diambil oleh robot. Dengan memahami seberapa padat rintangan dalam suatu sektor, robot dapat menghindari jalur yang memiliki banyak hambatan dan memilih jalur dengan kepadatan rintangan yang lebih rendah untuk navigasi yang lebih efisien.

Kepadatan rintangan dalam *Sector High* dihitung berdasarkan jumlah titik dalam sektor yang memenuhi kriteria tertentu. Semakin banyak titik yang memenuhi syarat dalam sektor tersebut, semakin tinggi nilai kepadatan rintangannya. Semakin padat rintangannya. Kepadatan (K_i) dihitung dengan persamaan 3.15

$$K_i = \sum_{i \epsilon\, sector} [y_i > 0.01,\ |x_i - x_{robot}|,\ z_i > 0.18\, m] \quad (3.15)$$

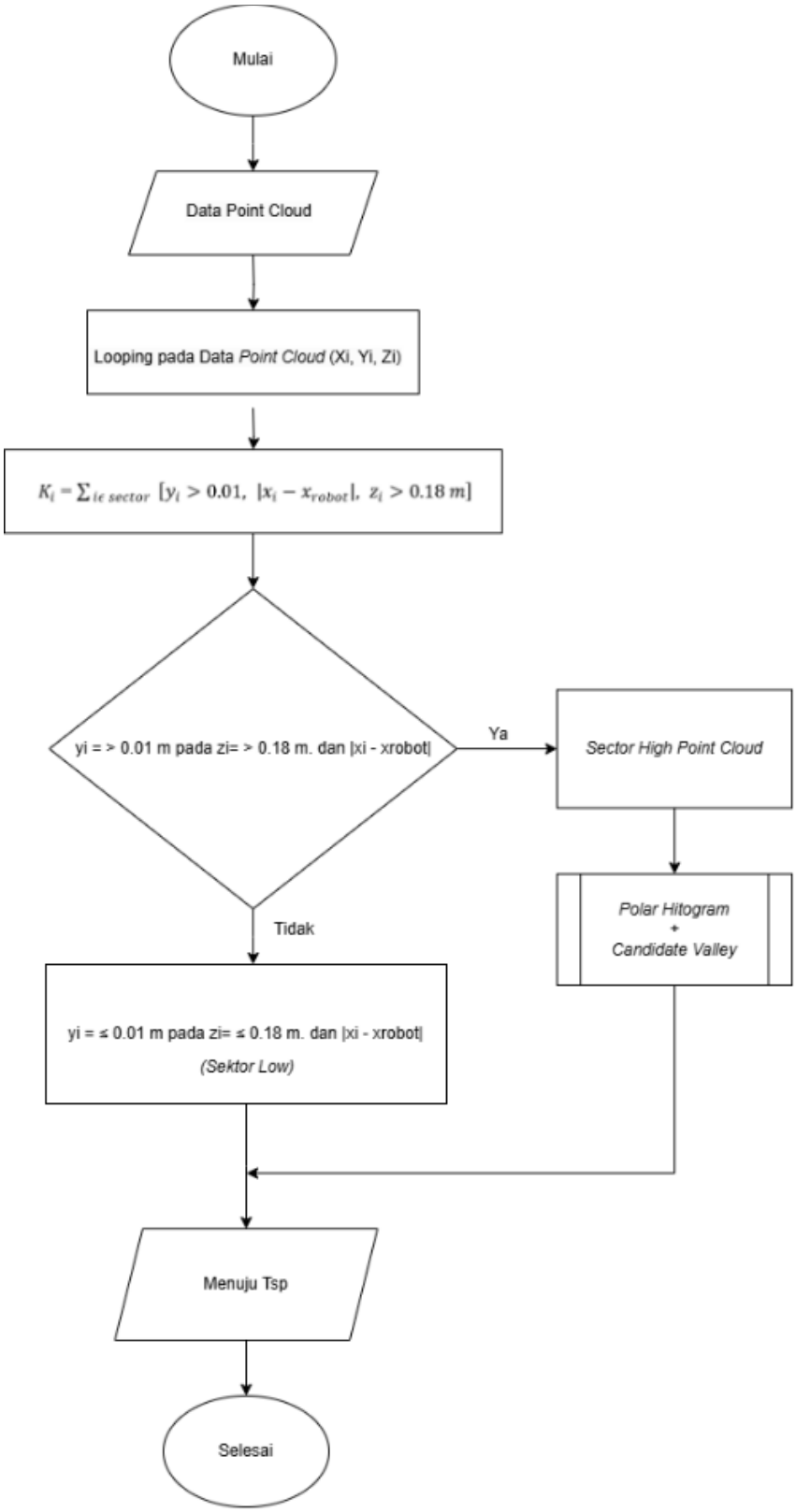
Dimana:

K_i adalah kepadatan

$\sum$ adalah penjumlahan nilai

i∈ *sector* adalah *Threshold sector high*

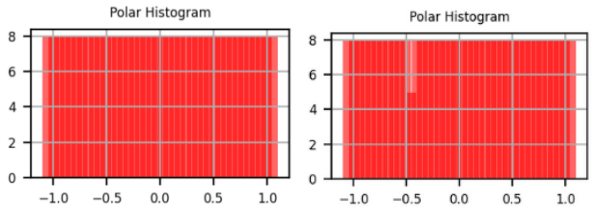
Berikut merupakan tampilan *flowchart* VFH yang ditunjukan pada Gambar 3.24.



Gambar 3.24. *flowchart VFH*

2. Deteksi *Polar Histogram* Pada *Vector Field Histogram*

Langkah perhitungan dalam implementasi VFH yang digunakan dalam sistem penghindaran rintangan hanya melibatkan dari *sector high point cloud* terhadap posisi robot, serta penghitungan penghindar kepadatan rintangan pada *polar histogram*. koordinat *polar* pada robot dibagi menjadi beberapa sektor. Robot menggunakan sistem *polar* dengan pembagian 0° sampai 180°, maka setiap sektor akan memiliki ukuran sudut tertentu. Pada *polar histogram*, setiap sektor menggambarkan interval sudut di sekitar robot. Jika sektor dibagi dalam satuan radian dan nilai yang diberikan adalah [−1.0, −0.5, 0.0, 0.5, 0.1], berikut merupakan visualisasi *polar histogram* pada Gambar 3.35.



(a) (b)

Gambar 3. 25. (a) Visualisasi *Polar Histogram* (b) *Pollar Histogram* Dengan *Obstacle* Pada *Sector* [-0.5]

Dengan pengaturan *polar histogram* pada robot akan dibagi dalam interval sudut 0° hingga 180°, di mana sektor tengah (arah utama robot) adalah 90°. Jadi, dengan pembagian sektor yang lebih besar. Pembagian sektor ini akan lebih menggambarkan distribusi penghalang atau rintangan dalam arah depan robot. Secara lebih rinci, berikut adalah cara pembagian sektor berdasarkan rentang 0° hingga 180° yang ditunjukan pada Tabel 3.3.

Tabel 3.3. Visualisasi Pembagian Sector

Sudut θ	Keterangan
0°	Area [-1.0] Menandakan arah kiri pojok
45°	Area [-0.5] Menandakan arah kiri tenggah
90°	Area [0.0] Menandakan pusat orantasi tenggah
135°	Area [0.5] Menandakan arah kanan tenggah
180°	Area [1.0] Menandakan arah kanan pojok

Dengan sistem yang digunakan, pembagian sektor berdasarkan sudut θ akan mencerminkan posisi titik-titik dalam *sector high point cloud* yang terdeteksi di sekitar robot. Berikut adalah pembagian sektor berdasarkan sudut θ yang lebih rinci dimana:

a. 0° (Area [-1.0]): Adalah area yang berada di sebelah kiri jauh (pojok kiri). Dalam sistem polar histogram, sektor ini akan mencakup area di kiri jauh robot.

b. 45° (Area [-0.5]): Adalah area kiri tengah, yang akan mencakup area kiri tengah robot.

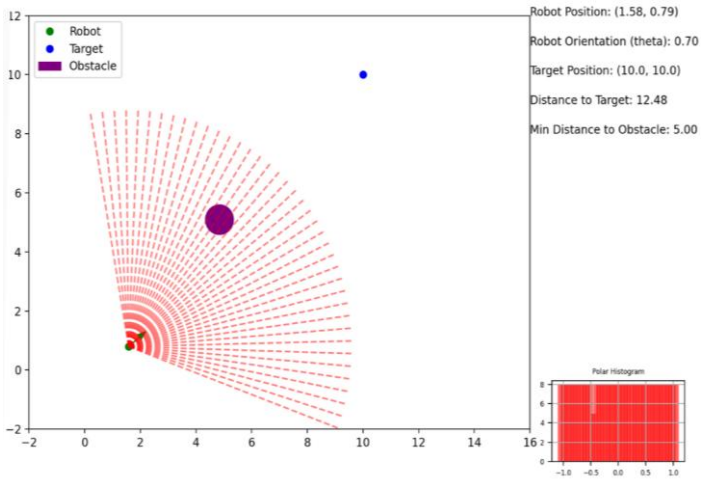
c. 90° (Area [0.0]): Adalah pusat orientasi robot (arah depan). Area ini menjadi titik referensi untuk arah utama robot.

d. 135° (Area [0.5]): Adalah area kanan tengah, yang mencakup area kanan tengah robot.

e. 180° (Area [1.0]): Adalah area yang berada di sebelah kanan jauh (pojok kanan). Area ini mencakup area kanan pojok robot.

3. *Candidate Valley* (Menentukan Arah Pergerakan Penghindar)

Candidate Valley merujuk pada area atau area-area yang diidentifikasi sebagai jalur potensial yang bebas dari hambatan atau rintangan, sehingga memungkinkan robot untuk bergerak menuju target dengan aman. Pada skenario halangan pada area [-0.5] *Candidate Valley* ditentukan dengan visualisasi yang ditunjukan pada Gambar 3.26.

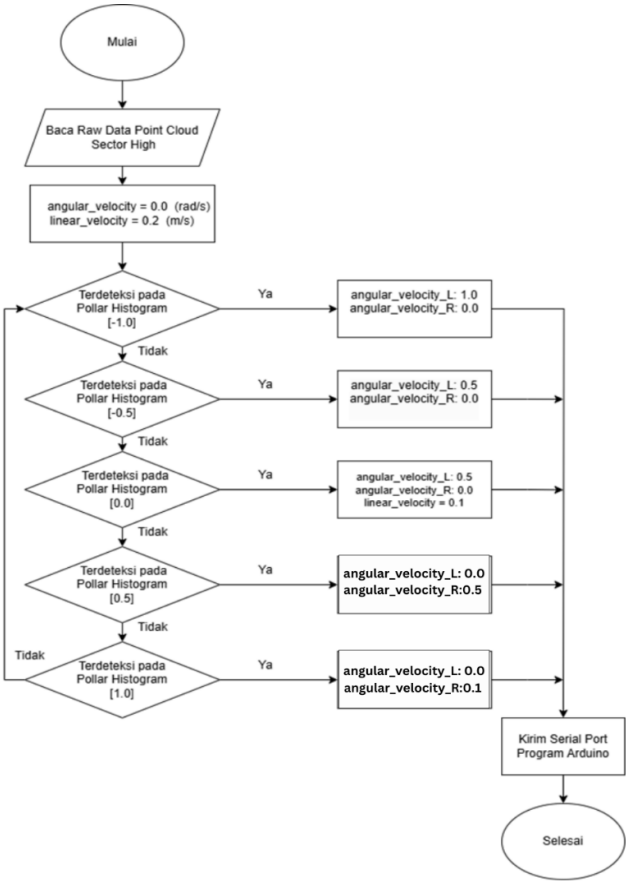


Gambar 3.26. Skenario Halangan Pada Sektor [-0.5]

Untuk mencari *Candidate Valley*, Menemukan sektor yang memiliki kepadatan rendah di antara dua area dengan kepadatan tinggi. Sebagai contoh, menganalisis area [-0.5] pada sudut 45°, yang memiliki kepadatan tinggi:

a. Area [-0.5] pada sudut 45° memiliki kepadatan, menunjukkan adanya rintangan.
b. Area [0.0°] pada sudut 90° memiliki kepadatan rendah, yang aman.
c. Area [0.5] pada sudut 135° juga memiliki kepadatan rendah, yang aman.
d. Jadi, dapat melihat bahwa area [0.0] dan [0.5] memiliki kepadatan rendah

Implementasi metode *Vector Field Histogram* (VFH) sebagai strategi penghindaran rintangan menggunakan data *point cloud*. Dengan metode ini, *point cloud* yang dihasilkan oleh sensor *kinect*. Proses ini melibatkan perhitungan jarak dan posisi titik terhadap orientasi robot untuk menentukan apakah titik tersebut menjadi bagian dari sektor aman atau terhalang. Berikut merupakan *flowchart polar histogram + candidate valley* digunakan yang ditunjukan pada Gambar 3.27.



Gambar 3.27. *Flowchart Polar Histogram + Candidate Valley*

BAB IV
HASIL PENELITIAN

4.1. Spesifikasi Sistem

Berikut dibawah ini merupakan spesifikasi sistem yang digunakan pada proposal robot beroda penghindar manusia menggunakan *vector field histogram* berbasis *point cloud,* pada tabel 4.1 dibawah ini.

Tabel 4.1. Spesifikasi Sistem (Laptop)

Nama	Keterangan
Prossesor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz (2.42 GHz)
Memory	8GB LPDDR4X
Graphic card	Intel® Iris Xe *Graphics*
Storage	512GB SSD
Wifi	Intel® Wireless Wi-Fi 6 AX201 802.11a/b/g/n/ac/ax
Dimension	302.5 (W) x 233.88 (D) x 15.95 (H) mm
Weight	1.19 kg
Display	13.5" IPS display with 2K Screen 3:2 ratio
OS	Windows 11 Home, Pre-Installed Office Home and Student 2019
Port	USB Type-C / ThunderboltTM 4 port USB 3.2 Gen 1 port with power-off charging

Pada kamera yang menggunakan Kinect v1 berikut merupakan spesifikasi yang digunakan pada Tabel 4.2.

Tabel 4.2. Spesifikasi sistem (Kinect v1)

Nama	Keterangan
Color camera	640x480 pixels @ 30 fps

Nama	Keterangan
Depth camera	320 × 240 pixels @ 30 fps (*structured infrared light*)
Depth Range	0.8 m – 4.0 m (*optimal* 1.2–3.5 m)
IR Emitter	830 nm *infrared laser projector*
Field of view	57° *horizontal*, 43° *vertical*
Frame rate (depth and color stream)	30 *frames per second* (FPS)
Audio format	16-kHz, 24-bit mono *pulse code modulation* (PCM)
USB	2.0 *connectivity*.
Power Requirement	12V DC, 1.08A (~13W *typical consumption*)
Power Adapter	12V *external power supply* (for PC or older Xbox 360 models)
Width	282 mm (11.1 in)
Height	76 mm (2.99 in)
Dimention	70 mm (2.76 in)
Weight	0.6 kg
SDK	OpenNI, Primesense, Kinect SDK

Berikut merupakan spesifikasi *microcontroller* arduino uno R4 wifi yang ditunjukan pada Tabel 4.3.

Tabel 4.3. Spesifikasi Arduino Uno R4 Wifi

Nama	Keterangan
Microcontroller	Renesas RA4M1 ARM Cortex-M4F 48 MHz
Flash Memory	256 KB
Operating Voltage	5V
Input Voltage (VIN)	6–24V
USB *Connector*	USB-C

Nama	Keterangan
Pin I/O *Digital*	14 pin (12 mendukung PWM)
Pin PWM	6 pin *dedicated*, sisanya soft-PWM
Pin *Analog Input*	6 (12-bit)
Analog Output (DAC)	1 kanal DAC 12-bit
Komunikasi Serial	UART, I²C, SPI
WiFi	ESP32-S3 (802.11 b/g/n)
Bluetooth	BLE & Bluetooth 5 (via ESP32-S3)
Board Power Consumption	± 154 mA saat WiFi

Berdasarkan spesifikasi yang dilampirkan pada tabel 4.1, 4.2, dan 4.3, sistem yang digunakan dalam proposal robot beroda penghindar manusia menggunakan metode *Vector Field Histogram* berbasis *point cloud* didukung oleh perangkat keras yang cukup mumpuni. Laptop yang digunakan memiliki prosesor Intel Core i5 generasi ke-11, memori 8GB, serta grafis Intel Iris Xe yang mampu menangani proses komputasi *point cloud* dan pengolahan sensor secara real time. Penyimpanan SSD 512GB, konektivitas Wi-Fi 6, serta layar 2K turut menunjang kelancaran pemrosesan data dan pengembangan perangkat lunak. Pada sisi sensor, Kinect v1 menyediakan kamera RGB 640×480 piksel dan *depth* sensor 320×240 piksel pada 30 FPS, dengan jangkauan kedalaman 0.8–4 meter, sehingga mampu menghasilkan data *point cloud* yang diperlukan untuk metode VFH. Sensor ini dilengkapi IR *emitter*, *field of view* yang cukup lebar, serta dukungan SDK seperti OpenNI dan Kinect SDK yang mempermudah integrasi sistem. Sementara itu, mikrokontroler Arduino Uno R4 WiFi berperan sebagai pengendali aktuator robot, dilengkapi mikrokontroler Renesas ARM Cortex-M4F, memori 256 KB, dukungan komunikasi UART/I2C/SPI, serta modul Wi-Fi dan Bluetooth melalui ESP32-S3. Spesifikasi ini menunjukkan bahwa seluruh perangkat yang digunakan sudah sesuai dan saling mendukung untuk menjalankan proses akuisisi data, pemetaan lingkungan, perhitungan *histogram vektor*, serta kontrol motor secara real time dalam sistem robot

penghindar manusia.

4.2. Implementasi Sistem

Navigasi robot beroda dilakukan berdasarkan rancangan yang telah dijelaskan pada bab sebelumnya. Sistem direalisasikan melalui integrasi perangkat keras dan perangkat lunak untuk mendukung fungsi penghindaran rintangan secara otonom menggunakan metode *Vector Field Histogram* (VFH). Informasi lingkungan diperoleh dari kamera *Kinect* dalam bentuk citra kedalaman yang selanjutnya diolah menjadi data *point cloud* sebagai masukan utama algoritma VFH. Hasil pemrosesan tersebut digunakan untuk menentukan arah gerak robot, kemudian dikirimkan ke mikrokontroler Arduino melalui komunikasi serial untuk mengendalikan aktuator motor. Implementasi ini memungkinkan robot bergerak secara responsif dan stabil dalam menghindari rintangan pada lingkungan pengujian. Berikut merupakan tampilan robot yang ditampilkan pada Gambar 4.1

Gambar 4.2. Fisik Robot Penelitian

4.3. Hasil Pembacaan *Point Cloud*

Point cloud diperoleh dari sensor Kinect v1 yang mampu menangkap informasi kedalaman (*depth*) setiap titik dalam bidang pandang kamera. Dengan memanfaatkan data ini, robot dapat memperoleh pemetaan lingkungan secara real-time, termasuk deteksi keberadaan objek atau rintangan di jalur pergerakannya. Pembacaan *point cloud* menjadi tahap awal yang penting sebelum dilakukan pengolahan lebih lanjut, seperti segmentasi sektor, perhitungan jarak, dan klasifikasi tingkat ketinggian (*high*/*low*) untuk penghindaran hambatan. Berikut merupakan hasil uji coba yang dilakukan sesuai dengan metodelogi penelitian yang di rencanakan.

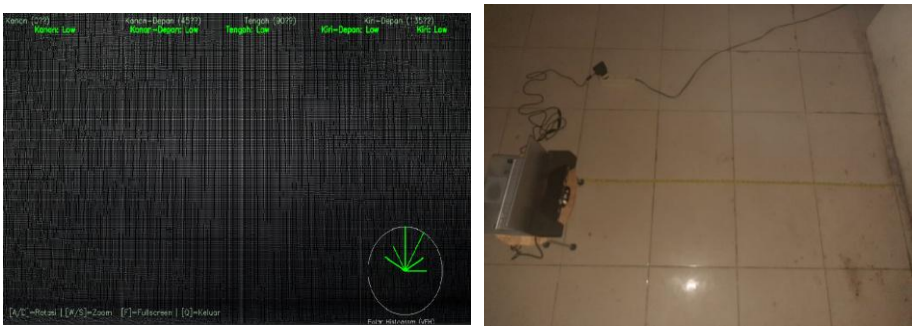
```
openni2.initialize()
dev = openni2.Device.open_any()
color_stream = dev.create_color_stream()
depth_stream = dev.create_depth_stream()
color_stream.start()
depth_stream.start()
```

Program dimulai dengan inisialisasi *Library* OpenNI2 untuk mengaktifkan komunikasi dengan sensor Kinect. Selanjutnya, perangkat Kinect dibuka menggunakan perintah “Device.open_any()” sehingga program dapat mengakses kamera yang tersedia. Setelah perangkat siap, dibuat dua aliran data atau stream, yaitu “color_stream” untuk menangkap frame warna (RGB) dan “depth_stream” untuk menangkap frame kedalaman (*depth*). Kedua stream ini kemudian dihidupkan menggunakan metode “start()”, sehingga program siap untuk membaca frame secara *real-time* dari sensor Kinect.

```
fx, fy = 525.0, 525.0
cx, cy = 319.5, 239.5
x = (c[valid] - cx) * z / fx
y = (r[valid] - cy) * z / fy
```

Program menetapkan parameter intrinsik kamera yang digunakan untuk konversi koordinat piksel pada gambar depth menjadi koordinat tiga dimensi di dunia nyata. Nilai “fx” dan “fy” mewakili panjang fokus (*focal length***)** kamera dalam arah horizontal dan vertikal, sedangkan “cx” dan “cy” menentukan pusat optik (*optical center*), yaitu titik

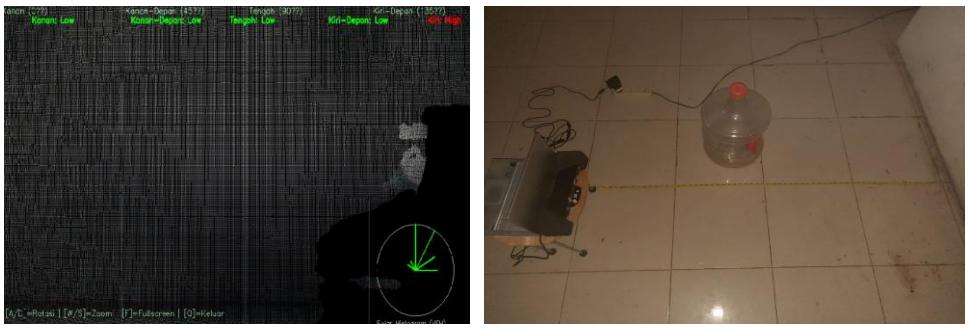
referensi di bidang gambar yang sesuai dengan sumbu optik kamera. Parameter ini penting untuk menghitung posisi X, Y, dan Z dari setiap titik pada point cloud menggunakan rumus proyeksi kamera, sehingga setiap titik *depth* dapat diubah menjadi koordinat 3D yang akurat. Berikut merupakan hasil uji tanpa *obstacle* yang di tunjukan pada Gambar 4.2



(a) (b)

Gambar 4.3. Hasil *Point Cloud* Tanpa halangan (*Sector Low*)

Pada tahap uji coba ini dilakukan pengamatan terhadap kemampuan sistem dalam mendeteksi keberadaan objek pada sektor kiri bidang pandang kamera. Dengan memanfaatkan data jarak minimum pada setiap sektor, sistem seharusnya memberikan status “High” pada sektor kiri apabila objek berada dalam jarak yang lebih dekat dari ambang batas yang telah ditentukan. Berikut merupakan pengujian yang diberi pada *sector* kiri pada Gambar 4.3

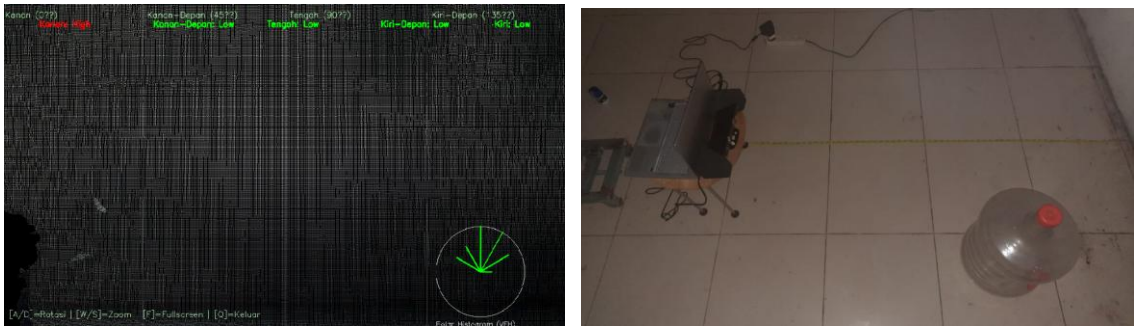


(a) (b)

Gambar 4.4. Hasil *Point Cloud* halangan kiri (*Sector High*)

Pada tahap uji coba ini dilakukan pengamatan terhadap kemampuan sistem dalam mendeteksi keberadaan objek pada sektor kanan bidang pandang kamera. Dengan memanfaatkan data jarak minimum pada setiap sektor, sistem seharusnya memberikan

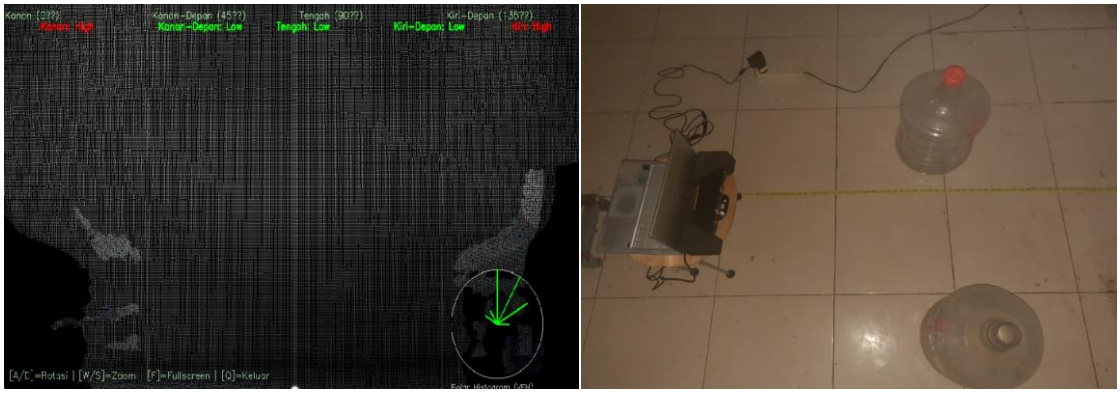
status "*High*" pada sektor kanan apabila objek berada dalam jarak yang lebih dekat dari ambang batas yang telah ditentukan. Berikut merupakan pengujian yang diberi pada *sector* kanan pada Gambar 4.4



(a) (b)

Gambar 4.5. Hasil *Point Cloud* halangan kanan (*Sector High*)

Pada tahap uji coba ini dilakukan pengamatan terhadap kemampuan sistem dalam mendeteksi keberadaan objek pada sektor kedua sisi bidang pandang kamera. Dengan memanfaatkan data jarak minimum pada setiap sektor, sistem seharusnya memberikan status "*High*" pada sektor kanan dan kiri apabila objek berada dalam jarak yang lebih dekat dari ambang batas yang telah ditentukan. Berikut merupakan pengujian yang diberi pada *sector* kanan dan kiri pada Gambar 4.5



(a) (b)

Gambar 4.6. Hasil *Point Cloud* halangan kanan-kiri (*Sector High*)

4.4. Hasil Perhitungan *Vector Field Histogram*

Hasil perhitungan *Vector Field Histogram* (VFH) diperoleh dari pengolahan data *point cloud* yang dihasilkan oleh kamera *Kinect* sebagai representasi kondisi lingkungan di sekitar robot. Data p*oint cloud* tersebut diproyeksikan ke dalam bidang polar dan dibagi

ke dalam beberapa sektor sudut untuk merepresentasikan kepadatan rintangan pada setiap arah. Nilai histogram yang dihasilkan digunakan sebagai dasar dalam menentukan sektor aman yang dapat dilalui oleh robot.

```
z = depth[valid]
x = (c[valid] - cx) * z / fx
y = (r[valid] - cy) * z / fy
```

Potongan kode di atas digunakan untuk mengonversi data citra kedalaman (*depth image*) menjadi koordinat tiga dimensi (*point cloud*) berdasarkan model kamera *pinhole*. Nilai z merepresentasikan jarak titik terhadap kamera dalam satuan meter yang diperoleh dari citra kedalaman *Kinect*. Koordinat x dan y dihitung dengan memproyeksikan posisi piksel pada bidang citra ke ruang tiga dimensi menggunakan parameter intrinsik kamera, yaitu panjang fokus (fx, fy) dan titik pusat kamera (cx, cy). Proses ini menghasilkan representasi spasial lingkungan dalam bentuk *point cloud* yang selanjutnya digunakan sebagai masukan dalam perhitungan *Vector Field Histogram* (VFH).

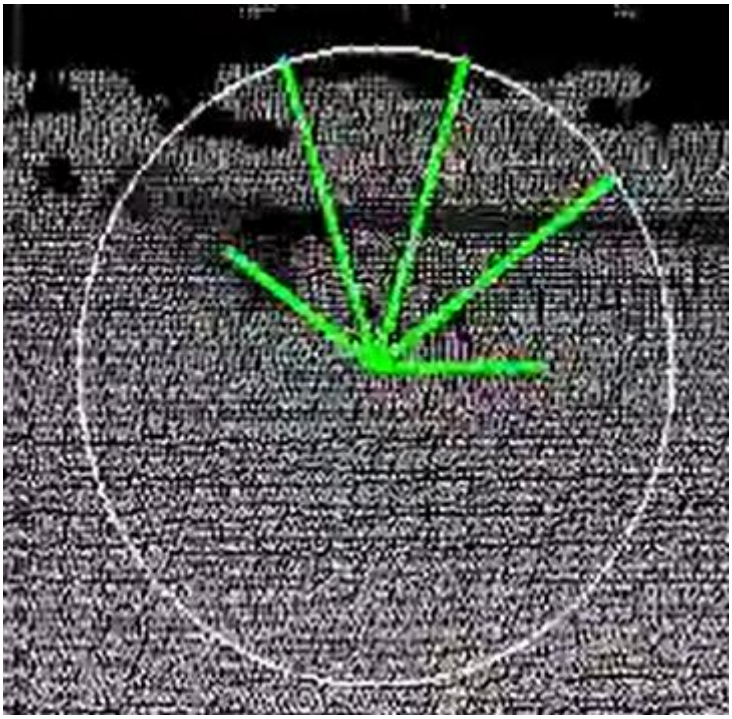
```
theta_point = np.arctan2(x_rot, z_rot)
theta_point = theta_point % np.pi
```

Untuk menghitung sudut polar (θ) dari setiap titik *point cloud* terhadap arah depan robot sebagai bagian dari proses pembentukan *Vector Field Histogram* (VFH). Fungsi "arctan2(x_rot, z_rot)" digunakan untuk memperoleh sudut antara posisi titik rintangan dan sumbu depan robot dengan mempertimbangkan tanda koordinat, sehingga sudut yang dihasilkan lebih stabil dibandingkan fungsi arctan biasa. Nilai sudut tersebut kemudian dinormalisasi ke dalam rentang 0 hingga π radian (0°–180°) menggunakan operasi modulo, yang merepresentasikan bidang pandang depan robot. Sudut polar ini selanjutnya digunakan untuk mengelompokkan titik-titik *point cloud* ke dalam sektor sudut pada histogram VFH.

4.4.1. Pembentukan *Polar Histogram*

Untuk merepresentasikan distribusi rintangan di sekitar robot berdasarkan arah sudutnya. *Polar histogram* dibentuk dari data *point cloud* yang diperoleh dari kamera Kinect setelah dilakukan perhitungan koordinat tiga dimensi dan sudut polar setiap titik

terhadap arah depan robot. Sudut polar tersebut dihitung menggunakan fungsi arctan2 sehingga setiap titik rintangan memiliki informasi arah relatif terhadap robot. Berikut merupakan Tampilan dari *Pollar Histogram* yang ditunjukan pada Gambar 4.6.



Gambar 4.7. *Pollar Histogram*

Representasi grafis dari distribusi rintangan di sekitar robot berdasarkan arah relatif terhadap sumbu depan. Pada sistem ini, data lingkungan diperoleh dari kamera Kinect dalam bentuk citra kedalaman (*depth image*), yang kemudian dikonversi menjadi koordinat tiga dimensi (x, y, z) untuk membentuk *point cloud*. Setiap titik pada *point cloud* mewakili posisi objek di ruang tiga dimensi relatif terhadap robot.

```
num_sectors = 5
histogram = np.zeros(num_sectors)
sector_status = {}
theta_point = np.arctan2(x_rot, z_rot)
theta_point = theta_point % np.pi

for i in range(num_sectors):
    theta_min = i * (np.pi / num_sectors)
    theta_max = (i + 1) * (np.pi / num_sectors)
```

```
    mask_sector = (theta_point >= theta_min) & (theta_point < theta_max)

    if np.any(mask_sector):
        min_distance = np.min(z[mask_sector])
        histogram[i] = min_distance if min_distance < 2 else 2
        sector_status[sector_labels[i]] = "High" if min_distance < threshold_distance else "Low"
    else:
        histogram[i] = 2
        sector_status[sector_labels[i]] = "Low"
```

Pada program diatas, pembentukan *polar histogram* dimulai dengan mendefinisikan jumlah sektor sebanyak lima sektor, yang merepresentasikan arah "kiri, kiri-depan, depan, kanan-depan, dan kanan" di sekitar robot. Selanjutnya dibuat array histogram untuk menyimpan nilai jarak minimum rintangan pada setiap sektor, serta dictionary sector_status untuk menyimpan klasifikasi sektor apakah terhalang ("High") atau aman ("Low"). Setiap titik pada *point cloud* kemudian dihitung sudut polarnya relatif terhadap sumbu depan robot menggunakan fungsi arctan2(x_rot, z_rot), dan sudut tersebut dinormalisasi ke rentang 0 hingga π radian agar hanya mencakup bidang pandang depan robot.

```
def draw_polar_histogram(canvas, histogram, max_val=2.0):
    center = (1080, 600)
    radius = 100
    cv2.circle(canvas, center, radius, (255, 255, 255), 1)
    sectors = len(histogram)
    for i, val in enumerate(histogram):
        theta = math.radians(i * (180 / sectors))
        r = radius * (val / max_val)
        x = int(center[0] + r * math.cos(theta))
        y = int(center[1] - r * math.sin(theta))
```

```
        cv2.line(canvas, center, (x, y), (0, 255, 0), 2)
    cv2.putText(canvas, "Polar Histogram (VFH)", (center[0]-
100, center[1]+radius+20),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255),
1)
```

Fungsi "draw_polar_histogram" bertugas untuk memvisualisasikan *pollar histogram* hasil perhitungan VFH pada kanvas gambar. Pertama, ditentukan titik pusat visualisasi dan jari-jari lingkaran sebagai acuan histogram. Setiap sektor pada histogram divisualisasikan sebagai garis radial dari pusat ke arah tertentu berdasarkan sudut sektor, dengan panjang garis proporsional terhadap nilai histogram pada sektor tersebut. Sudut setiap sektor dihitung dalam satuan radian, sedangkan panjang garis dihitung dengan membandingkan nilai histogram sektor dengan nilai maksimum yang telah ditentukan (max_val) sehingga semua garis dapat ditampilkan dalam skala yang seragam. Warna garis ditentukan hijau untuk menunjukkan distribusi rintangan pada setiap sektor. Selain itu, fungsi ini juga menambahkan teks "Polar Histogram (VFH)" sebagai label pada visualisasi untuk memudahkan pembaca memahami konteks gambar. Dengan visualisasi ini, pembaca dapat secara langsung melihat kepadatan rintangan di setiap sektor dan mengamati bagaimana nilai *pollar histogram* merepresentasikan kondisi lingkungan di sekitar robot.

4.4.2. Penentuan *Candidate Valley*

Setelah *pollar histogram* terbentuk, langkah berikutnya dalam *Vector Field Histogram* (VFH) adalah menentukan *candidate valley*, yaitu sektor-sektor aman yang dapat dilalui robot tanpa menabrak rintangan. *Candidate valley* diperoleh dengan menganalisis *pollar histogram* yang telah diklasifikasikan ke dalam sektor "High" (terhalang) dan "Low" (aman). Sektor-sektor bertanda "Low" yang saling berurutan membentuk sebuah *valley* atau celah bebas, yang menjadi kandidat arah gerak robot.

Proses ini penting karena robot harus memilih arah dengan ruang yang cukup untuk bergerak aman. Setiap *candidate valley* dievaluasi berdasarkan lebar sudutnya; *valley* yang terlalu sempit tidak dianggap aman dan diabaikan. Dari sekumpulan *candidate valley*, kemudian menentukan *valley* terbaik dengan mempertimbangkan

orientasi robot saat ini dan posisi relatif target, sehingga robot dapat melanjutkan pergerakan dengan risiko tabrakan yang minimal. Dengan cara ini, penentuan *candidate valley* menjadi dasar pengambilan keputusan arah pada VFH, yang memungkinkan robot menavigasi lingkungan dinamis secara real-time sambil menghindari rintangan.

```
sector_status[sector_labels[i]] = "High" if min_distance < threshold_distance else "Low"
```

Setelah nilai jarak minimum rintangan pada setiap sektor dihitung, setiap sektor diklasifikasikan menjadi dua kategori berdasarkan ambang jarak yang telah ditentukan (threshold_distance). Jika jarak minimum pada suatu sektor lebih kecil dari threshold, sektor tersebut dianggap memiliki rintangan yang dekat dan diberi status "*High*", menandakan bahwa arah tersebut terhalang dan kurang aman untuk dilalui robot. Sebaliknya, jika jarak minimum lebih besar dari threshold atau tidak ada rintangan yang terdeteksi, sektor dikategorikan sebagai "*Low*", menandakan bahwa sektor tersebut aman dan dapat dilalui robot. Klasifikasi ini disimpan dalam dictionary sector_status dengan label sektor sebagai kunci, sehingga setiap sektor memiliki informasi apakah terhalang atau aman. Status ini menjadi dasar bagi algoritma navigasi robot dalam menentukan arah pergerakan berikutnya, sehingga robot dapat menghindari rintangan secara *real-time*.

```
if sector_status.get("Kiri", "Low") == "High" or sector_status.get("Kiri-Depan", "Low") == "High":
    turn_direction = "RIGHT"
elif sector_status.get("Kanan", "Low") == "High" or sector_status.get("Kanan-Depan", "Low") == "High":
    turn_direction = "LEFT"
elif sector_status.get("Tengah", "Low") == "High":
    # Depan terhalang, pilih sisi yang lebih free
    if sector_status.get("Kiri-Depan", "Low") == "Low":
        turn_direction = "LEFT"
    else:
        turn_direction = "RIGHT"
```

vfh_time_based() yang bertugas menentukan arah belok robot berdasarkan

kondisi sektor yang telah diklasifikasikan dalam sector_status. Jika sektor kiri atau kiri-depan memiliki status "*High*", menandakan adanya rintangan dekat di sisi kiri, maka robot diarahkan untuk berbelok ke kanan. Sebaliknya, jika sektor kanan atau kanan-depan terhalang, robot diarahkan berbelok ke kiri. Jika sektor depan terhalang, kode akan mengevaluasi sisi yang lebih aman dengan memeriksa sektor kiri-depan; apabila sisi kiri-depan aman ("*Low*"), robot akan berbelok ke kiri, sedangkan jika tidak, robot berbelok ke kanan. Berikut merupakan contoh keluaran yang di tunjukan pada *candidate valey* pada Gambar 4.7.

(a) (b)

Gambar 4.8. (a) Tampilan *Obsticle* terdeteksi *Candidate Valley* (b) Tampilan *Real-time*

Berikut ini merupakan pengeksekusian yang telah diberikan oleh *candidate valley* yang mengarahkan pada kontrol DC Motor yang di tujukan pada Gambar 4.8.

Gambar 4.9. (a) Tampilan Setelah *Candidate Valley* Dieksekusi (b) Tampilan *Real-time*

4.4.3. *Output* Kecepatan Robot

Setelah arah gerak robot ditentukan berdasarkan polar histogram dan status sektor, tahap berikutnya adalah mengubah perintah arah menjadi *output* kecepatan roda robot. Pada sistem ini, perintah arah (FORWARD, LEFT, RIGHT) dikirim ke Arduino melalui

komunikasi serial, yang kemudian dikonversi menjadi sinyal kontrol untuk motor DC. Perintah “FORWARD” akan membuat kedua roda bergerak maju dengan kecepatan yang sama, sementara perintah “LEFT” dan “RIGHT” menyebabkan roda berlawanan bergerak berbeda untuk menghasilkan gerakan berbelok.

Pengiriman perintah ini dilakukan secara berulang setiap frame, namun hanya dikirim jika ada perubahan perintah dari frame sebelumnya, sehingga mengurangi beban komunikasi serial. Dengan cara ini, robot dapat merespons kondisi lingkungan secara *real-time*, menyesuaikan kecepatan dan arah geraknya sesuai dengan distribusi rintangan yang terdeteksi, sehingga pergerakan robot tetap aman dan stabil selama navigasi.

```
#define RPWM1 6
#define LPWM1 7
#define R_EN1 8
#define L_EN1 9

#define RPWM2 10
#define LPWM2 11
#define R_EN2 12
#define L_EN2 13

const int PWM_KIRI  = 59;
const int PWM_KANAN = 255;

void setup() {
  Serial.begin(9600);

  pinMode(RPWM1, OUTPUT);
  pinMode(LPWM1, OUTPUT);
  pinMode(R_EN1, OUTPUT);
  pinMode(L_EN1, OUTPUT);

  pinMode(RPWM2, OUTPUT);
```

```
  pinMode(LPWM2, OUTPUT);
  pinMode(R_EN2, OUTPUT);
  pinMode(L_EN2, OUTPUT);

  digitalWrite(R_EN1, HIGH);
  digitalWrite(L_EN1, HIGH);
  digitalWrite(R_EN2, HIGH);
  digitalWrite(L_EN2, HIGH);

  stopMotor();
  Serial.println("READY");
}

void loop() {
  if (Serial.available()) {
    String cmd = Serial.readStringUntil('\n');
    cmd.trim();

    Serial.println(cmd);

    if (cmd == "FORWARD") {
      forward();
    }
    else if (cmd == "LEFT") {
      left();
    }
    else if (cmd == "RIGHT") {
      right();
    }
    else if (cmd == "STOP") {
      stopMotor();
```

```
    }
  }
}

void forward() {
  analogWrite(RPWM1, PWM_KIRI);
  analogWrite(LPWM1, 0);

  analogWrite(RPWM2, 0);
  analogWrite(LPWM2, PWM_KANAN);
}

void left() {
  analogWrite(RPWM1, 0);
  analogWrite(LPWM1, 0);

  analogWrite(RPWM2, 0);
  analogWrite(LPWM2, PWM_KANAN);
}

void right() {
  analogWrite(RPWM1, PWM_KIRI);
  analogWrite(LPWM1, 0);

  analogWrite(RPWM2, 0);
  analogWrite(LPWM2, 0);
}

void stopMotor() {
  analogWrite(RPWM1, 0);
  analogWrite(LPWM1, 0);
```

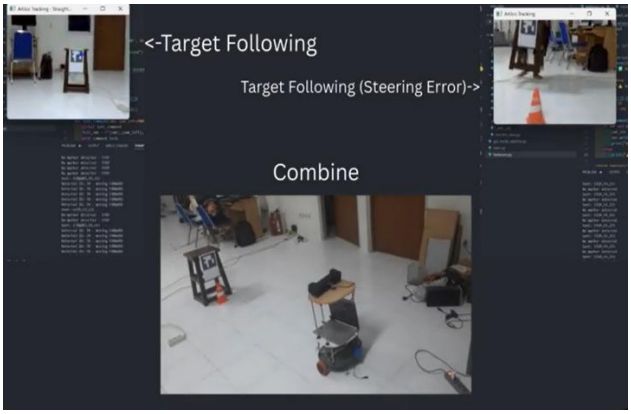
```
  analogWrite(RPWM2, 0);
  analogWrite(LPWM2, 0);
}
```

Program Arduino bertugas menerima perintah gerak dari *Python* melalui komunikasi serial dan mengubahnya menjadi sinyal kendali untuk motor DC pada robot. Pada bagian awal, setiap pin yang terhubung ke motor driver diatur sebagai *output* dan diaktifkan, termasuk pin PWM untuk mengatur kecepatan motor serta pin enable untuk mengaktifkan motor. Nilai PWM kiri dan kanan ditentukan oleh konstanta PWM_KIRI dan PWM_KANAN untuk mengatur kecepatan masing-masing motor sesuai kebutuhan.

4.5. Pengujian Robot Di Lingkungan Nyata

Pengujian ini bertujuan untuk memastikan bahwa seluruh komponen perangkat keras dan perangkat lunak yang telah diintegrasikan dapat bekerja dengan baik dalam kondisi operasional sebenarnya. Lingkungan nyata memiliki karakteristik yang lebih kompleks dibandingkan dengan simulasi, seperti adanya variasi pencahayaan, keberagaman objek, serta ketidakpastian posisi dan pergerakan manusia, sehingga pengujian ini menjadi indikator utama keberhasilan sistem.

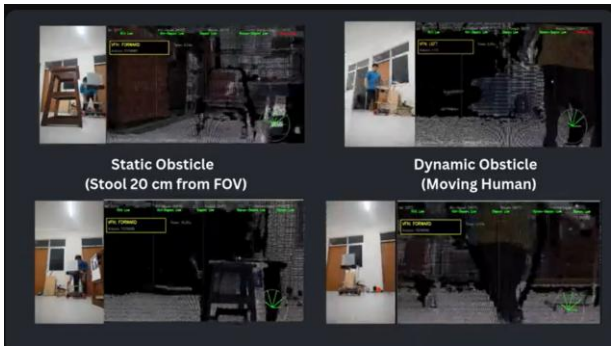
Pada tahap ini, robot diuji untuk menjalankan fungsi navigasi dan penghindaran manusia secara mandiri menggunakan data *point cloud* yang diperoleh dari sensor Kinect. Data tersebut diolah menggunakan metode *Vector Field Histogram* (VFH) untuk menentukan arah gerak yang aman bagi robot. Berikut ini merupakan Tampilan Percobaan *Target Following* yang di tunjukan pada Gambar 4.9.



(a) (b)

Gambar 4.10. (a) Tampilan percobaan *Target Following* (b) QR Link Video Percobaan

Selain itu, pengujian di lingkungan nyata dilakukan untuk mengevaluasi kinerja metode *Vector Field Histogram* (VFH) berbasis *point cloud* dalam mendukung navigasi robot. Pengujian ini juga mengamati kemampuan robot dalam menghindari tabrakan, kelancaran perubahan arah saat menghadapi rintangan, dan kestabilan gerakan selama proses navigasi berlangsung. Berikut merupakan tampilan percobaan VFH *Point Cloud* yang di tujukan padan Gambar 4.10.



(a) (b)

Gambar 4.11. (a) Tampilan percobaan VFH *Point Cloud* (b) QR Link Video Percobaan

Hasil pengujian menunjukkan bahwa metode VFH berbasis *point cloud* mampu menghasilkan keputusan arah gerak yang sesuai dengan kondisi lingkungan di sekitarnya. Robot dapat merespons keberadaan manusia dengan melakukan manuver penghindaran secara *real-time*, serta mempertahankan konsistensi gerakan sesuai dengan perintah navigasi yang dihasilkan oleh sistem. Dengan demikian, hasil pengujian ini memberikan gambaran bahwa metode VFH yang diterapkan telah berfungsi dengan baik. Berikut ini merupakan hasil percobaan dengan combinasi sesuai dengan batasan masalah dan rumusan masalah yang ditujukan pada Gambar 4.11.

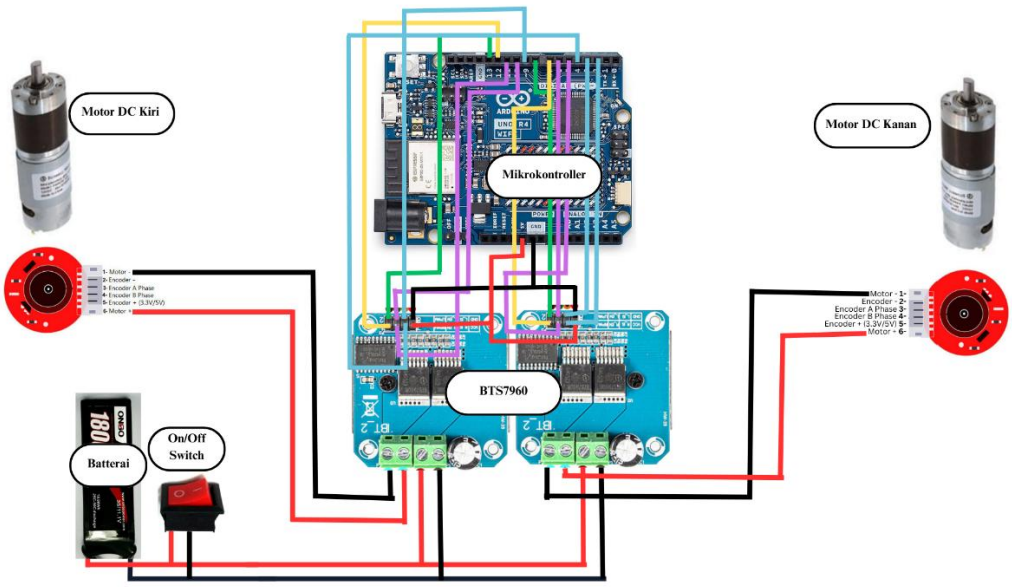
(a) (b)

Gambar 4.12. (a) Tampilan percobaan Hasil Penelitiaan (b) QR Link Video Percobaan

4.6. Evaluasi Koreksi Penelitian

Pada tahap evaluasi penelitian, dilakukan beberapa koreksi dan penyesuaian terhadap sistem yang telah dirancang sebelumnya. Koreksi pertama berkaitan dengan metode kontrol motor, di mana pada tahap perencanaan awal digunakan pendekatan *dynamic PWM*. Namun, berdasarkan hasil pengujian dan kalibrasi motor DC, ditemukan bahwa nilai PWM yang sama pada kedua motor tidak menghasilkan kecepatan putar yang seimbang. Oleh karena itu, pendekatan kontrol diubah menjadi *static PWM by command* dengan nilai PWM kiri sebesar 59 dan PWM kanan sebesar 255 agar robot dapat bergerak lurus secara kinematik. Perubahan ini dilakukan untuk meningkatkan kestabilan dan konsistensi pergerakan robot selama proses navigasi.

Koreksi selanjutnya meliputi perubahan pada komponen perangkat keras sistem. Pada tahap perencanaan awal, sistem penggerak motor dirancang menggunakan *motor shield*, namun pada tahap implementasi diganti dengan modul driver motor BTS7960. Pergantian ini dilakukan karena motor DC yang digunakan membutuhkan suplai tegangan sebesar 24 volt, sehingga BTS7960 dinilai lebih sesuai untuk menangani kebutuhan arus dan tegangan motor. Selain itu, fitur *emergency stop* yang direncanakan sebelumnya harus dieliminasi karena keterbatasan jumlah pin input pada mikrokontroler Arduino yang digunakan, sehingga dilakukan prioritas fungsi pada sistem navigasi dan kendali utama robot. Berikut ini merupakan skematik setelah hasil uji yang ditunjukan pada Gambar 4.13.



Gambar 4.13. Skema Rangkaian Elektronika Koreksi

Perubahan Selanjutnya berkaitan dengan metode deteksi target pada pengujian *traffic cone*. Pada perencanaan awal, deteksi objek dirancang menggunakan metode YOLO dan *Haar Cascade*, namun hasil pengujian menunjukkan bahwa kedua metode tersebut hanya mampu berjalan pada kisaran 5 frame per *second* (fps), yang tidak memenuhi kebutuhan sistem *real-time* dengan target minimal 30 fps. Oleh karena itu, metode deteksi diganti menggunakan ArUco marker, yang berdasarkan hasil pengujian mampu berjalan stabil pada kisaran 20–30 fps. Metode ini dinilai lebih sesuai untuk kebutuhan sistem *augmented reality* dan integrasi dengan sistem navigasi robot, sehingga dipilih sebagai solusi yang lebih efektif dalam penelitian ini.

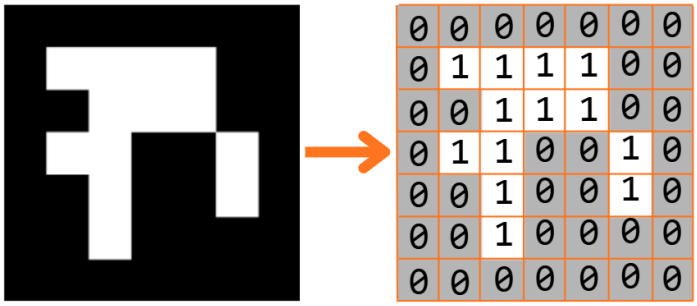
Rendahnya nilai *frame per second* (fps) pada metode *Haar Cascade* dan YOLO dalam penelitian ini disebabkan oleh tingginya beban komputasi yang diperlukan pada tahap pemrosesan citra. Metode *Haar Cascade* melakukan ekstraksi fitur secara berlapis pada setiap *frame*, sehingga membutuhkan waktu pemrosesan yang relatif besar ketika dijalankan pada resolusi citra yang cukup tinggi. Sementara itu, metode YOLO berbasis *deep learning* memerlukan proses inferensi jaringan saraf konvolusional yang kompleks, yang sangat bergantung pada kemampuan perangkat keras komputasi. Pada sistem yang digunakan dalam penelitian ini, pemrosesan dilakukan secara *CPU-based* tanpa akselerasi GPU, sehingga proses inferensi YOLO tidak dapat berjalan secara optimal dan menyebabkan penurunan fps secara signifikan.

```
aruco_dict = aruco.getPredefinedDictionary(aruco.DICT_5X5_1000)
aruco_params = aruco.DetectorParameters()

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
corners, ids, _ = aruco.detectMarkers(gray, aruco_dict,
parameters=aruco_params)
```

ArUco mudah dieksekusi dan mampu mempertahankan kinerja stabil hingga 30 frame per *second* (fps) karena prinsip deteksinya berbasis *pola biner diskrit*, bukan pada ekstraksi fitur kompleks atau pembelajaran mesin. Setiap marker ArUco direpresentasikan dalam bentuk *grid biner*, di mana setiap sel hanya memiliki dua kemungkinan nilai, yaitu hitam (0) atau putih (1). Proses identifikasi marker dilakukan dengan cara membagi area kandidat ke dalam sejumlah sel tetap, kemudian membaca

nilai biner dari masing-masing sel. Mekanisme ini membuat proses pengenalan marker bersifat deterministik dan memiliki kompleksitas komputasi yang rendah. Berikut merupakan Tampilan contoh pola biner diskrit pada salah satu contoh id ArUco yang ditunjukan pada Gambar 4.14.



Gambar 4.14. Pola Biner Diskrit ArUco

Selain itu, pembacaan pola biner pada ArUco dilakukan setelah proses penyederhanaan citra melalui *thresholding*, sehingga variasi pencahayaan dapat diminimalkan dan informasi yang diproses menjadi sangat sederhana. Berbeda dengan metode berbasis fitur atau *deep learning* yang harus memproses nilai piksel kontinu dan melakukan komputasi berlapis, ArUco hanya memerlukan operasi logika dasar untuk menentukan kesesuaian pola biner dengan kamus marker yang telah didefinisikan sebelumnya. Hal ini memungkinkan proses deteksi dan identifikasi marker dilakukan secara cepat pada setiap *frame*. Berikut merupakan perbandingan hasil penelitian yang ditunjukan pada Tabel 4.4.

Tabel 4.4. Perbandingan Metode Pose Hasil Penelitian

Metode Deteksi	Prinsip Deteksi	Komputasi	FPS (CPU *Bassed*)
Haar Cascade	Ekstraksi Fitur *Haar* + *Sliding window*	Menengah	5–10 fps
YoLo	*Deep Learning* (CNN)	Sangat tinggi	3–7 fps
ArUco *Marker*	Pola biner (*fiducial marker*)	Rendah	20–30 fps

BAB V
ANALISIS AKHIR

5.1. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem robot beroda penghindar manusia menggunakan *Vector Field Histogram* (VFH) berbasis *point cloud*, maka dapat disimpulkan beberapa hal sebagai berikut:

1. Sistem robot beroda penghindar manusia berhasil dikembangkan dengan memanfaatkan sensor Kinect sebagai sumber data kedalaman untuk membentuk *point cloud* lingkungan, serta algoritma *Vector Field Histogram* (VFH) sebagai metode navigasi dalam menentukan arah gerak yang aman. Sistem navigasi dijalankan pada platform berbasis Python, sedangkan eksekusi gerak robot dikendalikan oleh mikrokontroler Arduino melalui komunikasi serial.
2. Hasil pengujian menunjukkan bahwa algoritma VFH berbasis *point cloud* mampu mendeteksi keberadaan manusia dan objek penghalang secara efektif, serta menghasilkan keputusan arah gerak yang sesuai dengan kondisi lingkungan di sekitarnya. Robot mampu melakukan manuver penghindaran secara *real-time* dengan pergerakan yang relatif stabil dan responsif terhadap perubahan lingkungan.
3. Penerapan metode kontrol motor dengan nilai PWM statis hasil kalibrasi, yaitu PWM kiri sebesar 59 dan PWM kanan sebesar 255, terbukti mampu menjaga keselarasan pergerakan robot agar dapat bergerak lurus secara kinematik. Pendekatan ini mengatasi perbedaan karakteristik motor DC yang tidak dapat dicompensasi secara optimal menggunakan nilai PWM yang sama.
4. Penggunaan metode deteksi berbasis ArUco marker sebagai penanda target menunjukkan kinerja yang lebih stabil dibandingkan metode *Haar Cascade* dan YoLo pada sistem berbasis CPU. ArUco marker mampu berjalan pada kisaran 20–30 *frame per-second* (fps), sehingga sesuai untuk mendukung sistem navigasi robot yang membutuhkan respons *real-time* tanpa membebani proses pengolahan *point cloud* dan metode VFH.
5. Integrasi sensor Kinect dalam membentuk representasi lingkungan berbasis *point cloud* dengan metode *Vector Field Histogram* (VFH) terbukti mampu cukup

akurat untuk keperluan navigasi robot. Pengolahan *point cloud* menjadi *histogram polar* memungkinkan sistem untuk menyederhanakan data lingkungan yang kompleks menjadi keputusan arah gerak yang efisien, sehingga robot dapat melakukan penghindaran manusia dan rintangan secara adaptif di lingkungan nyata.

5.2. Saran

Berdasarkan hasil penelitian yang telah dilakukan, beberapa saran untuk pengembangan sistem pada penelitian selanjutnya dapat dirumuskan sebagai berikut:

1. Penelitian selanjutnya disarankan untuk mengintegrasikan sistem navigasi robot dengan metode *Simultaneous Localization and Mapping* (SLAM) agar robot tidak hanya mampu menghindari manusia dan rintangan secara reaktif, tetapi juga dapat membangun peta lingkungan serta mengetahui posisinya secara mandiri. Integrasi SLAM diharapkan dapat meningkatkan kemampuan navigasi robot dalam lingkungan yang lebih luas dan tidak terstruktur.
2. Pengembangan sistem dapat diarahkan pada penggunaan *Robot Operating System* (ROS) sebagai kerangka kerja utama, sehingga arsitektur sistem menjadi lebih modular, terstandarisasi, dan mudah dikembangkan. Penggunaan ROS memungkinkan integrasi berbagai paket navigasi, pemetaan, serta komunikasi antar *node* yang lebih efisien, serta mempermudah pengujian dan pengembangan sistem secara berkelanjutan.
3. Untuk meningkatkan tingkat kemandirian (*autonomous capability*) robot, penelitian selanjutnya dapat mengembangkan sistem pengambilan keputusan yang tidak hanya berbasis aturan (*rule-based*), tetapi juga berbasis pembelajaran. Penerapan metode *Reinforcement Learning* (RL) memungkinkan robot untuk mempelajari strategi navigasi dan penghindaran rintangan yang optimal berdasarkan pengalaman interaksi dengan lingkungan.
4. Selain itu, sistem kontrol gerak robot dapat dikembangkan menggunakan konsep *action primitives*, di mana setiap perilaku dasar seperti maju, belok, berhenti, dan menghindar direpresentasikan sebagai aksi terstruktur. Pendekatan ini dapat mempermudah integrasi antara modul perencanaan tingkat tinggi, sistem pembelajaran, dan kontrol motor.

DAFTAR PUSTAKA

[1] Ummah. Amirul, 2023 "Rancang bangun robot penghindar halangan dengan metode pid" jurnal teknik mesin, industri, elektro dan informatika (jtmei) p-issn: 2963-8208, hal 212-222.

[2] Bilesan. Alireza, 2021 "Improved 3D Human Motion Capture Using Kinect Skeleton and Depth Sensor" Jurnal Mechanical & Mechatronic Penerbit: Department of Mechanical Engineering, National Defense Academy of Japan 1-10-20 Hashirimizu, Yokosuka, Kanagawa, 239-8686, Japan.

[3] Gustanto, Saleh, 2019 "Kinerja Metode Vector Field Histogram Dalam Perilaku Menghindari Rintangan Pada Robot Sepak Bola Beroda " Jurnal Teknik Elektro. Penerbit: Universitas Islam “45” Bekasi Jl. Cut Meutia No. 83. Bekasi Timur, 021-883-444-36

[4] Hengm, Guang. 2019 "Analysis of performance between kinect v1 dan kinect v2 for various facial part movement" IEEE International Conference on Consumer Electronics (ICCE). DOI: 10.1109/ICSEngT.2019.8906419

[5] Baoqi, Yao. 2020 "Human Motion Recognition by Three-view Kinect Sensors in Virtual Basketball Training" *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1318-1334. DOI: 10.1109/TENCON50793.2020.9293892

[6] Jiaqiang, Zhou. 2021" Sparse Point Cloud Generation Based on Turntable 2D Lidar and Point Cloud Assembly in Augmented Reality Environment" IEEE *International Instrumentation and Measurement Technology Conference* (I2MTC) , DOI: 10.1109/I2MTC50364.2021.9459981

[7] Zong, C., Du, Q., Chen, J., Shan, Y., Wu, Y., & Sha, Z. 2024. "A Design of Three-Dimensional Spatial Path Planning Algorithm Based on Vector Field Histogram*." *Sensors*, 24(17), 5647.

[8] Mohammed, K., Aliedani, A., & Al-Ibadi, A. 2024. “Adaptive Vector Field Histogram Plus (VFH+) Algorithm using Fuzzy Logic in Motion Planning for Quadcopter” *Journal of Robotics and Control (JRC)*, 5(2)

[9] Ahmad Taher Azar. 2023. **Mobile Robot: Motion Control and Path Planning (Studies in Computational Intelligence, 1090)** Springer Press

[10] Muhammad Fuad, 2020 “Collision Avoidance of Multi Modal Moving Objects

for Mobile Robot Using Hybrid Velocity Obstacles" INASS: 13(3):407421: DOI:10.22266/ijies2020.0630.37

[11] Yusuf. Samson D, 2020 "Construction of A Pulse Width Modulation, (Pwm) Dc Motor Controller" SSRG International Journal of Electrical and Electronics Engineering (SSRG - IJEEE) - Volume 7 Issue 4