



TUGAS AKHIR - EE 184801

**PENENTUAN POSISI TENDANGAN KE GAWANG
MENGUNAKAN PREDIKSI AREA KOSONG DENGAN
CITRA KEDALAMAN PADA ROBOT SEPAK BOLA
BERODA**

Dzulfikar Ahmad Samhan
NRP 0711164000016

Dosen Pembimbing
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Ir. Tasripan, MT.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2019



ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - EE184801

**PENENTUAN POSISI TENDANGAN KE GAWANG
MENGUNAKAN PREDIKSI AREA KOSONG DENGAN
CITRA KEDALAMAN PADA ROBOT SEPAK BOLA
BERODA**

**Dzulfikar Ahmad Samhan
NRP 0711164000016**

**Dosen Pembimbing
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Ir. Tasripan, MT.**

**DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020**



FINAL PROJECT - EE184801

**SHOOTING AIM DETERMINATION USING EMPTY-AREA
PREDICTION WITH DEPTH IMAGE IN WHEELED
SOCCER ROBOT**

**Dzulfikar Ahmad Samhan
NRP 0711164000016**

**Supervisor
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Ir. Tasripan, MT.**

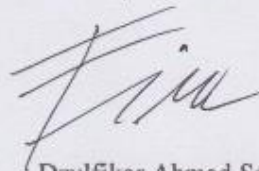
**ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electrical and Intelligent Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020**

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Penentuan Posisi Tendangan ke Gawang Menggunakan Prediksi Area Kosong dengan Citra Kedalaman pada Robot Sepak Bola Beroda**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 27 Mei 2020



Dzulfikar Ahmad Samhan
NRP. 0711 16 4000 0016

**PENENTUAN POSISI TENDANGAN KE GAWANG
MENGUNAKAN PREDIKSI AREA KOSONG
DENGAN CITRA KEDALAMAN PADA ROBOT
SEPAK BOLA BERODA**

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I



Dr. Ir. Hendra Kusuma, M.Eng.Sc.
NIP. 19640902 198903 1 003

**SURABAYA
JULI, 2020**

**PENENTUAN POSISI TENDANGAN KE GAWANG
MENGUNAKAN PREDIKSI AREA KOSONG
DENGAN CITRA KEDALAMAN PADA ROBOT
SEPAK BOLA BERODA**

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing II



Ir. Tasripan, MT.
NIP. 19620418 199003 1 004

**SURABAYA
JULI, 2020**

PENENTUAN POSISI TENDANGAN KE GAWANG MENGGUNAKAN PREDIKSI AREA KOSONG DENGAN CITRA KEDALAMAN PADA ROBOT SEPAK BOLA BERODA

Nama : Dzulfikar Ahmad Samhan
Pembimbing I : Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Pembimbing II : Ir. Tasripan, MT.

ABSTRAK

Pada pertandingan robot sepak bola beroda, sistem otomatisasi robot perlu dirancang dengan kemampuan mengambil keputusan berdasarkan kondisi lingkungannya. Penelitian ini bertujuan untuk memberikan kemampuan menentukan arah tendangan pada robot IRIS (Tim sepak bola beroda ITS) dengan memprediksi area kosong pada gawang.

Kemampuan ini, didapatkan dari pengolahan citra kedalaman *depth camera*. Dengan pemenuhan spesifikasi *Depth camera* antara lain yang *robust*/tahan atas cahaya dan juga memiliki frame rate per detik (FPS) diatas 30. Karena itu pada sistem tugas akhir ini akan digunakan *depth camera* Intel Realsense D435i yang memenuhi spesifikasi tersebut.

Penelitian ini menghasilkan otomatisasi sistem yang dapat melakukan perhitungan koordinat area kosong pada gawang di semua titik pengujian. Perhitungan koordinat ini didapat dengan cara melakukan eliminasi pada area gawang yang tertutup oleh objek penjaga gawang, sehingga akan didapat area gawang yang tidak dijaga. Keberhasilan sistem untuk mendeteksi area kosong mencapai 100% dengan rata-rata error yang didapat dari koordinat yang terdeteksi sebesar 1.3%, berdasarkan pengujian pada 60 titik uji. Penelitian ini diharapkan mampu meningkatkan kemampuan robot IRIS dalam pertandingan

Kata Kunci :depth image, MSL, stereo vision, robot sepak bola beroda

Halaman ini sengaja dikosongkan

DETERMINING SHOOTING POSITION USING FREE SPACE PREDICTION WITH DEPTH IMAGE IN WHEELED SOCCER ROBOT

Name : Dzulfikar Ahmad Samhan
Supervisor : Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Co-Supervisor : Ir. Tasripan, MT.

ABSTRACT

In a wheeled soccer robot competition, all robot is designed to be able to move around without being controlled by human. Therefore, the ability of decision making is a must. This research focused in implementing the ability to improve the shooting aim of IRIS robot (wheeled soccer robot team of ITS) by predicting unguarded area of the goal.

This aiming ability will be enhanced with depth image and RGB image that are obtained from depth camera. To be able to process the image smoothly, this research will use Intel Realsense D435i due to the fact that this camera can capture frame with high fps (greater than 30 FPS), and can withstand different light intensity.

As the result, this system has a good reliability to calculating unguarded space of goal with 100% detecting accuracy and 1.3% error of detected coordinate in 60 test point. This calculation is based on the process of eliminating the guarded goal area. This result conclude that the system can determine shooting coordinate for every test point

Keywords: Depth Image, MSL, Stereo vision

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Penulis mengucapkan Syukur atas berkat Rahmat Tuhan Yang Maha Esa sehingga penulis mampu untuk menyelesaikan penelitian tugas akhir ini yang berjudul Penentuan Posisi Tendangan ke Gawang Menggunakan Prediksi Area Kosong dengan Citra Kedalaman pada Robot Sepak Bola Beroda.

Penelitian dan penulisan Tugas Akhir ini merupakan persyaratan untuk dapat menyelesaikan pendidikan program Strata-Satu di Departemen Teknik Elektro, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini didasarkan pada teori dan praktik yang telah didapat melalui perkuliahan, pengalaman penulis sebagai salah satu anggota tim robot, serta berbagai literatur penunjang lainnya

Atas banyaknya pihak-pihak yang telah membantu penulis untuk menyelesaikan penelitian ini, penulis ingin mengucapkan banyak terimakasih khususnya kepada:

1. Bapak dan Ibu penulis yang selalu menemani penulis dalam pengerjaan Tugas Akhir ini.
2. Dr. Ir. Hendra Kusuma, M.Eng.Sc. Sebagai dosen pembimbing 1 atas segala bimbingan, arahan dan petunjuk yang diberikan dalam proses pengerjaan Tugas Akhir ini.
3. Ir. Tasripan, MT. Sebagai dosen pembimbing 2 atas segala bimbingan, arahan dan petunjuk yang diberikan dalam proses pengerjaan Tugas Akhir ini.
4. Dedet Candra Riawan, ST., M.Eng., Ph.D. selaku Kepala Departemen Teknik Elektro ITS Surabaya.
5. Dimas Anton Asfani S.T., M.T., Ph.D. selaku Sekretaris Departemen 1 Teknik Elektro ITS Surabaya.
6. Rizky Prasetya Ade Nugroho S.T. atas masukan dan saran yang diberikan selama pengerjaan Tugas Akhir ini
7. Raden Roro Widya Ningtyas Soeprajitno S.A. selaku teman terbaik penulis yang selalu menemani penulis dalam pengerjaan Tugas Akhir ini
8. Seluruh dosen dan tenaga kependidikan departemen Teknik Elektro
9. Seluruh staff dan karyawan departemen Teknik Elektro
10. Teman-teman tim IRIS yang telah banyak membantu penulis untuk merancang mekanik, elektronik dan program

11. Teman-teman laboratorium B-202 yang telah banyak membantu proses pengerjaan tugas akhir ini.

Penulis sangat menyadari bahwa Tugas akhir ini masih memiliki banyak kekurangan. Untuk itu besar harapan penulis atas kritik dan saran yang membangun. Semoga hasil dari penelitian ini dapat meningkatkan kualitas dari tim Robot ITS

Surabaya, 12 Juli 2020

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

ABSTRAK	i
ABSTRACT	iii
KATA PENGANTAR	v
DAFTAR ISI	viii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiv
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Perumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan	2
1.5 Metodologi	2
1.6 Sistematika Penulisan.....	3
1.7 Relevansi dan Manfaat.....	4
BAB 2 TINJAUAN PUSTAKA DAN TEORI PENUNJANG	5
2.1 Tim IRIS.....	5
2.1.1 Hardware tim IRIS	5
2.1.2 <i>Software</i> robot IRIS.....	6
2.2 Intel RealSense D435i.....	7
2.2.1 <i>RGB Image</i> dan <i>Depth Image</i>	8
2.2.2 <i>Inertia Measurement Unit</i> dan Sistem Koordinat.....	9
2.3 <i>Mini-PC</i> Intel NUC 6i7 KYK.....	10
2.4 STM32F4.....	11
2.5 <i>OpenCV</i>	12
2.6 <i>Librealsense</i> dan <i>Librealsense 2</i>	13
2.7 <i>Rotary Encoder</i>	13
2.8 Sistem Koordinat dan Pengenalan Gawang IRIS	14
2.9 Sistem Penentuan Sudut Gawang IRIS	15

BAB 3 PERANCANGAN SISTEM.....	17
3.1 Perancangan Sistem	17
3.1.1 Algoritma Penentu Koordinat.....	18
3.1.1.1 Blok <i>Depth camera</i>	19
3.1.1.2 Blok <i>Vision</i>	22
3.1.1.3 Blok Konverter.....	26
3.1.2 Algoritma Gerak.....	28
3.2 Kontrol Pergerakan	32
3.2.1 Sensor IMU.....	34
3.2.2 Kinematika dan Motor.....	35
3.2.3 Rotary Encoder	35
3.3 Desain mekanik robot	37
3.4 Desain elektronik robot.....	39
BAB 4 PENGUJIAN	43
4.1 Uji Penentuan Koordinat.....	44
4.2 Pengujian Jarak Pada 2 Titik	47
4.3 Uji Deteksi Gawang	50
4.4 Pengujian Jumlah Goal	52
BAB 5 KESIMPULAN DAN SARAN.....	56
5.1 Kesimpulan.....	56
5.2 Saran.....	56
DAFTAR PUSTAKA	57
LAMPIRAN	59
BIODATA PENULIS.....	86

Halaman ini sengaja dikosongakan

DAFTAR GAMBAR

Gambar 2.1 Logo Tim IRIS	5
Gambar 2.2 Ilustrasi <i>Hardware</i> Robot IRIS	6
Gambar 2.3 Logo ROS.....	6
Gambar 2.4 Struktur Program IRIS	Error! Bookmark not defined.
Gambar 2.5 Intel RealSense D435i.....	8
Gambar 2.6 Perbandingan <i>RGB image</i> dan <i>Depth Image</i>	9
Gambar 2.7 Sistem Koordinat Intel Realsense D435i	10
Gambar 2.8 Intel NUC 6i7KYK.....	11
Gambar 2.9 STM32F4	12
Gambar 2.10 Logo OpenCV.....	13
Gambar 2.11 Gambaran Koordinat lapangan IRIS Sisi Kanan	14
Gambar 2.12 Gambaran Koordinat lapangan IRIS Sisi Kiri.....	14
Gambar 2.13 Gambaran proses penentuan sudut tembakan ke gawang	15
Gambar 3.1 Blok Diagram Sistem.....	18
Gambar 3.2 Blok Diagram Algoritma Penentu Koordinat.....	19
Gambar 3.3 Tampilan Realsense-Viewer Untuk Mengatur Parameter kamera	20
Gambar 3.4 Grafik Perubahan FPS Tiap Satuan Waktu	21
Gambar 3.5 Perbandingan <i>Field of View</i> Sebelum proses	22
Gambar 3.6 Perbandingan <i>Field of View</i> Setelah proses	22
Gambar 3.7 Block Diagram proses vision	23
Gambar 3.8 <i>RGB Image</i>	23
Gambar 3.9 Pemilihan Objek berwarna putih	24
Gambar 3.10 Pemilihan objek dengan jarak relatif dari robot ke gawang	24
Gambar 3.11 Hasil Akhir Gawang	25
Gambar 3.12 Deteksi Area Kosong pada Gawang	26
Gambar 3.13 Gambaran Deteksi Gawang.....	27
Gambar 3.14 Contoh Penggambaran Posisi Area Kosong yang Didapat Oleh sistem	28
Gambar 3.15 Flow Chart Algoritma Tendang.....	29
Gambar 3.16 Interpretasi yang Tidak Sesuai (Gambar Gawang terpotong)	30
Gambar 3.17 Interpretasi Posisi gawang di Depan Gawang	31
Gambar 3.18 Interpretasi Posisi gawang di Kanan Gawang	31
Gambar 3.19 Interpretasi Posisi gawang di Kiri Gawang	32
Gambar 3.20 Blok Diagram Kontrol Robot	33
Gambar 3.21 Konfigurasi posisi motor pada base robot.....	35

Gambar 3.22 Trackbar Sebagai Input Posisi	36
Gambar 3.23 Interpretasi Posisi Robot (Titik Hitam) Berdasarkan Trackbar	37
Gambar 3.24 Desain Robot	38
Gambar 3.25 Posisi <i>Depth Camera</i> pada Robot.....	38
Gambar 3.26 Realisasi Robot	39
Gambar 3.27 Diagram Komunikasi Sistem Elektronik	39
Gambar 3.28 Skema PinOut STM32F4 Discovery	41
Gambar 3.29 Blok Diagram Elektronik Pengganti.....	41
Gambar 4.1 Xiaomi DUKA LS-P.....	43
Gambar 4.2 Pengujian Koordinat.....	44
Gambar 4.3 Pergeseran 20 cm ke Kanan	45
Gambar 4.4 Pergeseran 20 cm ke Kiri	45
Gambar 4.5 Pergeseran Jarak 100 cm.....	45
Gambar 4.6 Pengujian 2 Titik	48
Gambar 4.7 Contoh Uji Deteksi Gawang 1.....	51
Gambar 4.8 Contoh Uji Deteksi Gawang 2.....	51
Gambar 4.9 Contoh Uji Deteksi Gawang 3.....	51

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Tabel 3.1 Parameter PID	33
Tabel 4.1 Hasil Pengukuran Jarak	46
Tabel 4.2 Hasil Pengujian Jarak 2 Titik	49
Tabel 4.3 Pengujian Goal dan Koordinat deteksi	53

Halaman ini sengaja dikosongkan

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi telah mewarnai sejarah manusia. Teknologi yang dimiliki oleh suatu negara dapat mempengaruhi pandangan politik, social, dan ekonomi negara lain[1]. Salah satu bidang teknologi yang cukup berpengaruh adalah bidang robotika. Robot telah menjadi bagian yang tidak terpisahkan dari kehidupan manusia. Banyak pekerjaan yang dulunya hanya bisa dilakukan oleh manusia, kini telah digantikan oleh robot[2]. Perkembangan yang pesat ini didasari dari semakin berkembangnya teknologi kecerdasan buatan sehingga robot dapat melakukan berbagai pekerjaan kompleks.

Perkembangan ini telah banyak mendapat dukungan dari pemerintah, salah satunya dengan mengadakan banyak kompetisi yang mengambil tema robotika. Salah satu perlombaan nasional yang banyak diminati adalah Kontes Robot Indonesia (KRI). Cabang perlombaan Robot sepakbola beroda di KRI telah menjadi salah satu cabang perlombaan favorit. Meskipun baru diadakan pada tahun 2017, cabang lomba ini telah menarik lebih dari 20 tim dari berbagai universitas di Indonesia, dimana masing-masing tim diharapkan untuk mendesain sistem robot yang dapat bertanding dengan baik. Tentu hal ini merupakan tantangan yang cukup berat.

Salah satu tantangan yang dihadapi dalam mendesain sistem robot ini adalah kemampuan untuk mencetak goal. Kemampuan ini erat kaitannya dengan pengenalan gawang dan penjaga gawang. Dengan pengenalan gawang dan penjaga gawang, robot dapat memperhitungkan area gawang yang tidak dijaga.

Penelitian ini bertujuan untuk memberikan robot kemampuan untuk memperhitungkan sudut sasaran tendangan berdasarkan area kosong gawang yang tidak dijaga. Penelitian ini akan menggunakan *depth camera* Intel Realsense D435i yang memiliki kemampuan untuk membedakan jarak dari setiap objek yang ada di hadapannya [3]–[6]. Fokus pada penelitian ini hanya pada bagian pendeteksian area kosong pada gawang saja, sedangkan sistem pergerakan akan seluruhnya menggunakan sistem milik tim IRIS

1.2 Perumusan Masalah

Permasalahan pada penelitian ini adalah sebagai berikut:

1. Bagaimana cara mendapatkan citra kedalaman untuk melakukan deteksi gawang dan penjaga gawang
2. Bagaimana cara melakukan *noise filtering* untuk mengurangi *noise* pada gambar yang didapat
3. Bagaimana cara sistem menentukan sudut target tendangan berdasarkan posisi gawang dan penjaga gawang

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut:

1. Robot penjaga gawang, lingkungan pengujian dan robot yang digunakan mengacu pada *rule book* KRI nasional 2019 [7].
2. Jarak pendeteksian hanya pada batasan 2 hingga 6 meter
3. Objek penghalang tendangan hanya robot penjaga gawang

1.4 Tujuan

Tujuan dari pembuatan penelitian ini adalah sebagai berikut:

1. Mengambil gambar dengan tingkat akurasi tinggi dan *noise* seminimal mungkin
2. Membuat algoritma untuk mendeteksi area kosong pada gawang

1.5 Metodologi

Metodologi yang digunakan pada penelitian tugas akhir ini adalah sebagai berikut:

1. Studi Literatur

Pada tahap ini, akan dikumpulkan literatur dari penelitian-penelitian yang telah diterbitkan sebelumnya, untuk menentukan metode yang tepat untuk digunakan

2. Perancangan Sistem

Sistem yang akan dibuat adalah suatu sistem baru yang belum pernah diterapkan sebelumnya. Untuk itu, perlu dilakukan penyesuaian dari segi mekanik dan program pada robot. Tahap ini adalah tahapan pembuatan dan penyesuaian sistem pada robot

3. Pengujian dan Penyempurnaan Sistem

Untuk dapat mengetahui hasil dari sistem yang telah dibuat, perlu dilakukan suatu pengujian yang terukur. Bentuk pengujian pada penelitian ini dibagi menjadi beberapa tahap sebagai berikut :

- a. pengujian deteksi jarak pada *depth camera* untuk mengetahui ketelitian *depth camera*
- b. Pengujian pendeteksian gawang dengan 1 objek penjaga gawang yang diam. Hal ini dilakukan untuk memastikan bahwa sistem dapat mengenali area kosong dengan baik.
- c. Pengujian perhitungan tendangan ke gawang. Untuk memastikan sistem dapat mencetak goal dari koordiant yang telah dibaca

4. Penulisan Laporan Tugas Akhir

Tahapan ini dilakukan untuk mendokumentasikan hasil dari penelitian yang telah dilakukan, dan disesuaikan dengan sistematika penulisan yang berlaku

1.6 Sistematika Penulisan

Untuk dapat mendokumentasikan hasil dari penelitian dengan baik, diperlukan aturan penulisan yang diatur dalam sistematika sebagai berikut:

- **BAB 1: PENDAHULUAN**

Bab ini menjelaskan hal-hal umum dan mendasar dalam penelitian meliputi latar belakang, perumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan serta relevansi dan manfaat.

- **BAB 2: TINJAUAN PUSTAKA DAN TEORI PENUNJANG**

Pada bab ini akan dijelaskan teori- teori yang menunjang pengerjaan penelitian, yang berasal dari penelitian-penelitian yang telah dilakukan sebelumnya

- **BAB 3: PERANCANGAN SISTEM**

Sistem yang dibuat dalam penelitian ini akan memberikan perubahan baik dalam segi *software* maupun *hardware* pada robot. Perubahan dan penambahan tersebut akan didokumentasikan dalam bab ini

- **BAB 4: PENGUJIAN**

Untuk dapat menentukan apakah metode yang digunakan telah mencapai tujuan yang diinginkan, perlu dilakukan pengujian terhadap

sistem pada robot. Dengan cara pengujian yang telah dijelaskan pada bagian metodologi

- **BAB 5: PENUTUP**

Bab penutup menjelaskan hasil dan kesimpulan yang didapat dari penelitian yang telah dilakukan.

1.7 Relevansi dan Manfaat

Manfaat yang akan didapat dari penelitian ini adalah akan meningkatnya akurasi tendangan ke gawang dan semakin besar peluang untuk mencetak goal

BAB 2

TINJAUAN PUSTAKA DAN TEORI PENUNJANG

2.1 Tim IRIS

Penelitian ini merupakan salah satu riset dibidang robotika yang memiliki bidang khusus yaitu robot sepak bola beroda. Robot pada bidang ini memiliki tujuan layaknya dalam permainan bola, yaitu mendapatkan skor sebanyak mungkin dengan memasukkan bola ke gawang lawan, dan mencegah lawan memasukkan bola ke gawang tim. Penelitian ini akan berpatokan pada teknologi yang dimiliki oleh tim IRIS.



Gambar 2.1 Logo Tim IRIS

Tim IRIS (ITS' Robot with Intelligent System) adalah salah satu tim robot ITS yang telah banyak mengikuti perlombaan diluar maupun di dalam negeri. Saat ini tim IRIS memiliki 5 robot yang dapat beroperasi secara penuh [8]. Hasil dari penelitian ini nantinya akan dapat diimplementasikan pada masing-masing robot

2.1.1 Hardware tim IRIS

Untuk dapat melakukan fungsinya dengan baik, robot memerlukan perencanaan yang baik dari segi mekanik atau *hardware*. *Hardware* dari robot akan sangat menentukan pergerakan-pergerakan yang bisa dilakukan oleh robot tersebut. Gambar 2.2 adalah gambaran dari bentuk robot IRIS



Gambar 2.2 Ilustrasi *Hardware* Robot IRIS

Robot IRIS menggunakan kamera *omni-directional*, agar dapat melihat lingkungan disekitarnya tanpa perlu memutar robot, dilengkapi dengan 2 roda *dribble* untuk menggiring bola, 1 penendang dengan alat gerak motor untuk melakukan tendangan dan operan bola

2.1.2 Software robot IRIS

Software adalah penentu dari pergerakan dan kemampuan robot. Meskipun suatu robot memiliki *hardware* yang bekerja dengan maksimal, semua akan percuma apabila *software* pada robot tidak mampu untuk memberikan perintah dengan baik. Terutama pada robot yang bekerja secara *autonomous* seperti IRIS. Kemampuan *software* untuk memberikan perintah pada *hardware* di setiap kondisi permainan adalah penentu kalah dan menang dalam suatu pertandingan.

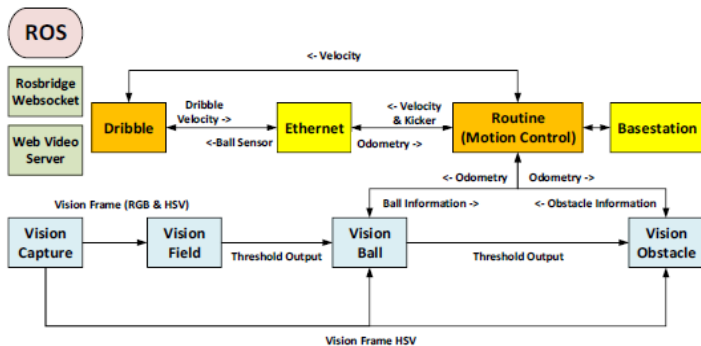


Gambar 2.3 Logo ROS¹

Sejak tahun 2019, IRIS telah menggunakan ROS (*Robot Operating System*)(Gambar 2.3) versi kinetic yang dijalankan dengan *operating*

¹ <https://www.ros.org/>

system Ubuntu 16.04 sebagai *platform* utama untuk pengembangan program. *Platform* ini dipilih karena memudahkan pengembangan program, serta memiliki kemampuan untuk menjalankan banyak proses secara *multi – thread*. Sehingga mempercepat kinerja robot secara keseluruhan. Secara garis besar, struktur program milik IRIS dapat disusun sebagai berikut pada **Error! Reference source not found.**



Gambar 2.4 Struktur Program IRIS²

2.2 Intel RealSense D435i

Proses penentuan jarak menjadi bagian penting dari penelitian ini. Untuk itu dibutuhkan alat yang dapat mengukur jarak dari objek yang ditangkap oleh kamera. Dengan mendapatkan jarak, sistem akan lebih mudah untuk melakukan proses seleksi objek yang diinginkan dari gambar yang ditangkap. Alat yang digunakan dalam penelitian ini adalah *depth camera*. Alat ini dapat menangkap gambar layaknya kamera, namun disertai dengan kemampuan memperkirakan jarak dari tiap *pixel* yang ditangkap oleh kamera. *Depth camera* yang digunakan pada kasus ini adalah Intel RealSense D435i (Gambar 2.5). Secara umum, *depth camera* memiliki 3 metode untuk mendapatkan jarak pada *pixel*, yaitu *passive stereo camera* dan *active stereo camera*. *active stereo camera*

² Dokumentasi IRIS

dibagi menjadi 2 metode yaitu *Structured Light* dan *Time of Flight (TOF)*. Semua metode ini menggunakan 2 kamera yang disinkronisasi.



Gambar 2.5 Intel RealSense D435i

Pada kamera berjenis *active stereo camera*, salah satu lensa kamera digantikan dengan infrared yang ditembakkan ke beberapa bagian gambar. *structured light* menembakkan infrared pada gambar dengan pola yang teratur, dan melihat perubahan pola cahaya yang mengenai objek. Sedangkan prinsip TOF mengukur waktu dari penembakan infrared hingga infrared mengenai objek dan kembali ke kamera[9].

Sedangkan *Passive stereo camera* adalah jenis kamera yang mengukur jarak berdasarkan perbedaan dari gambar yang ditangkap oleh kamera kanan dan kiri. Prinsip ini meniru cara manusia memprediksi jarak berdasarkan kedua mata manusia. *Passive stereo camera* cocok digunakan pada kondisi di dalam dan luar ruangan, namun tidak cocok pada kondisi gelap, sedangkan *active stereo camera* lebih cocok pada kondisi dalam ruangan dan tetap dapat bekerja dalam kondisi gelap[6], [9].

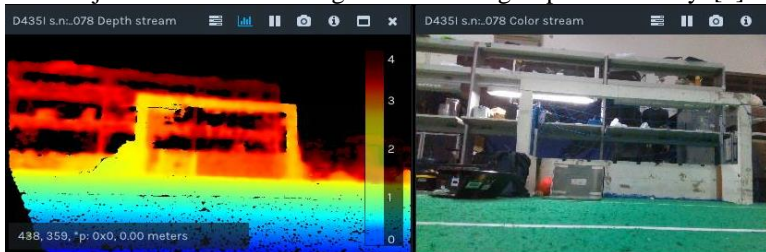
Keputusan menggunakan Intel RealSense dalam penelitian ini diambil berdasarkan spesifikasi Intel Realsense D435i yang dapat menghasilkan 90 fps, resolusi 848 x 480, serta kemampuan untuk melakukan perhitungan jarak pada kamera, dengan ukuran yang relatif kecil[6]

2.2.1 RGB Image dan Depth Image

Gambar yang diambil dari kamera, umumnya merupakan gambar 2 dimensi. Gambar 2 dimensi memiliki tingkat kesulitan yang cukup tinggi untuk memprediksi posisi benda secara 3 dimensi. Hal ini dikarenakan

dengan kamera biasa, jarak dari objek ke benda tidak dapat ditentukan. Selain itu, perubahan koordinat pixel kamera yang tidak linear dengan koordinat nyata menambah kesulitan prediksi. Permasalahan ini dapat diatasi dengan mengambil *depth image* atau citra kedalaman.

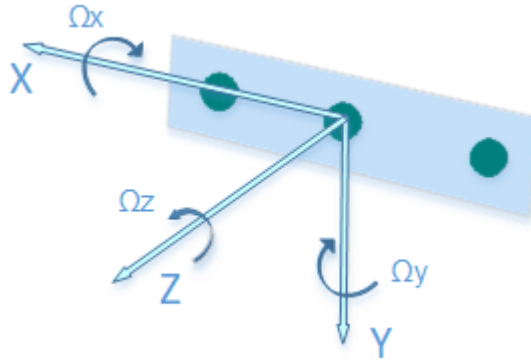
Citra kedalaman adalah citra yang menggambarkan jarak antara kamera dengan objek yang ditangkap oleh kamera. Citra ini ditampilkan dengan perpaduan terang dan gelap. Perpaduan ini menggambarkan jarak antara kamera dan objek. Semakin terang bagian citra, menggambarkan bahwa objek semakin dekat dengan kamera. Begitu pula sebaliknya[3].



Gambar 2.6 Perbandingan *RGB image* dan *Depth Image*

2.2.2 *Inertia Measurement Unit* dan Sistem Koordinat

Salah satu kelebihan yang dimiliki Intel RealSense D435i dengan *depth camera* lain adalah *inertia measurement unit* (IMU) yang dapat diakses untuk mengetahui perubahan posisi dan arah hadap kamera. Fitur ini sangat berguna untuk melakukan konversi dari koordinat yang didapat oleh kamera, ke sistem koordinat robot.



Gambar 2.7 Sistem Koordinat Intel Realsense D435i³

Berdasarkan [10] sistem koordinat pada Intel RealSense D435i memiliki arah sumbu x positif ke kanan, sumbu z positif ke depan, dan sumbu y positif ke bawah

2.3 *Mini-PC Intel NUC 6i7 KYK*

Sebagai perangkat yang berperan untuk menjadi sistem utama, *mini-PC* pada penelitian ini membutuhkan spesifikasi perangkat yang cukup tinggi. Berdasarkan perhitungan harga dan spesifikasi, peneliti memutuskan untuk menggunakan Intel NUC 6i7 KYK sebagai *mini-PC*.

Perangkat ini menggunakan prosesor *intel Core i7* generasi ke-6, dengan *clock speed* mencapai 3.5 GHz. Media penyimpanan yang digunakan adalah *solid state drive* dengan kecepatan transfer data yang lebih tinggi dibanding *Hard Disk Drive*.

Sebagai sistem utama, perangkat ini akan berperan dalam penambilan dan pengolahan data dari *depth camera* dan memberikan perintah pergerakan robot

³ <https://www.intelrealsense.com/how-to-getting-imu-data-from-d435i-and-t265/>



Gambar 2.8 Intel NUC 6i7KYK

2.4 STM32F4

Sensor yang digunakan dalam penelitian ini cukup beragam. Beberapa data pembacaan sensor bahkan perlu untuk diolah dan dikonversi terlebih dahulu sebelum dapat dikirim ke *mini-PC*. Untuk itu digunakan *microcontroller*, yang berfungsi sebagai perangkat penerima dan pengolah data sensor sebelum memasuki *mini-PC*.

Microcontroller yang digunakan dalam penelitian ini adalah STM32F4 dengan *clock speed* maksimal 168 MHz dan merupakan *Microcontroller* berbasis ARM 32-bit. Perangkat ini memiliki 14 timer dan 6 *Universal Asynchronous Receiver Transmitter (UART)* yang masing-masingnya dapat digunakan secara independen tanpa mempengaruhi kinerja satu sama lain[11].



Gambar 2.9 STM32F4⁴

2.5 OpenCV

OpenCV (Open source computer vision library) adalah *library* pemrograman yang berfokus pada bidang *computer vision* dan *machine learning*. *Library* ini bersifat *open-source*. Sehingga seluruh bagian program dapat diakses, dipelajari, dan diubah oleh siapapun[12]. Hal inilah yang mendasari digunakannya *library* OpenCV dalam penelitian ini. Selain mudah untuk dipelajari dan diimplementasikan, *library* ini dapat diakses dengan gratis.

Penelitian ini akan banyak menggunakan OpenCV sebagai media untuk memproses citra yang didapat dari kamera. Salah satu contoh penggunaan OpenCV dalam penelitian ini adalah proses penyeleksian gawang. Proses ini melibatkan citra kedalaman dan citra RGB. Dalam prosesnya, sistem hanya akan mendeteksi suatu objek sebagai gawang jika jarak objek sesuai, dan objek berwarna putih. Penggabungan informasi dari citra RGB dan citra kedalaman inilah yang mengimplementasikan OpenCV dalam sistem.

⁴ P. S. Tantra, "PENENTUAN POSISI ROBOT SEPAK BOLA BERODA BERDASARKAN PENGINDRAAN VISUAL," hlm. 88, 2018.



Gambar 2.10 Logo OpenCV⁵

2.6 Librealsense dan Librealsense 2

Library opencv memungkinkan untuk melakukan pemrosesan dari citra yang ditangkap kamera. Namun, opencv tidak dapat mengakses kamera dengan fungsi khusus seperti Intel Realsense yang digunakan pada penelitian ini. Untuk itu digunakanlah *library* Librealsense 2. *Library* ini dibuat oleh tim Intel Realsense untuk memudahkan penggunaan dari produk *depth camera* yang mereka produksi.

Library ini akan digunakan untuk mengakses citra RGB, citra kedalaman, pembacaan *inertial measurement unit* (IMU) dan melakukan perhitungan koordinat posisi objek yang telah ditangkap oleh kamera.

2.7 Rotary Encoder

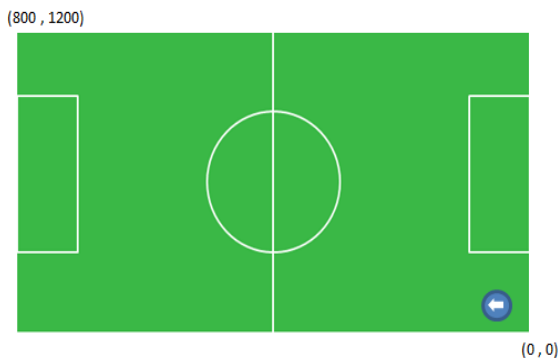
Penting bagi robot sepak bola beroda untuk dapat mengetahui posisi robot relative terhadap lapangan. Kemampuan ini akan sangat menentukan kemampuan robot saat bertanding. Karena dengan mengetahui informasi posisi, robot dapat menentukan sudut ke arah gawang, dan posisi robot lain.

Rotary encoder adalah sensor yang dapat menambahkan kemampuan penentuan posisi pada robot. Pada dasarnya, sensor ini bekerja dengan cara menghitung putaran dari roda robot. Dengan mengetahui banyaknya putaran, jika dikombinasikan dengan jari-jari dari roda, akan didapat perubahan posisi robot dari posisi awal robot.

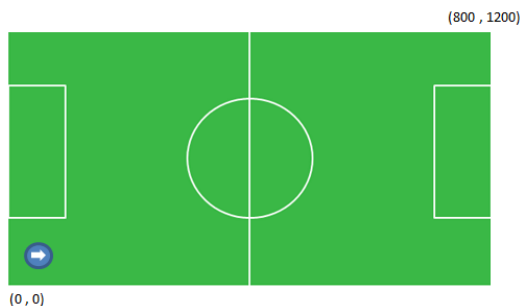
⁵ <https://opencv.org/>

2.8 Sistem Koordinat dan Pengenalan Gawang IRIS

Sistem koordinat yang dirancang dengan matang memudahkan robot untuk melakukan koordinasi berdasarkan posisi lawan, posisi robot satu tim, dan posisi gawang. Dengan adanya sistem koordinat yang seragam antar tiap robot, akan memudahkan sistem untuk membagi informasi posisi yang diterima satu robot ke robot lain.



Gambar 2.11 Gambaran Koordinat lapangan IRIS Sisi Kanan



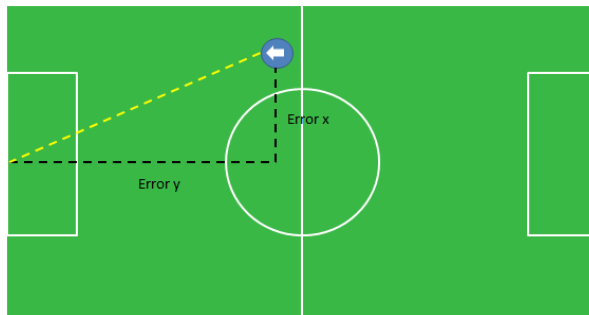
Gambar 2.12 Gambaran Koordinat lapangan IRIS Sisi Kiri

Dalam sistem koordinat tim IRIS, pojok kiri belakang lapangan sisi tim IRIS adalah koordinat $x = 0$ dan $y = 0$. Sedangkan pojok kanan depan lapangan adalah koordinat $x = 800$ dan $y = 1200$. Ukuran lapangan 800×1200 ini adalah ukuran lapangan yang digunakan dalam pertandingan KRI nasional tahun 2020 [7]. Gambar 2.61 menunjukkan gambaran

koordinat lapangan saat robot IRIS menempati sisi kanan lapangan dan Gambar 2.6.2 adalah sebaliknya

2.9 Sistem Penentuan Sudut Gawang IRIS

Sistem yang digunakan tim IRIS untuk menentukan sudut tendangan ke gawang saat ini masih cukup sederhana. Sudut tendangan ke gawang hanya ditentukan dari posisi robot saat ini dan posisi gawang yang telah didefinisikan pada program. Mengacu pada sistem koordinat tim IRIS yang telah dijelaskan pada sub bab sebelumnya, posisi gawang lawan ada pada koordinat $x = 400$ dan $y = 1200$, dan gawang milik tim IRIS berada pada koordinat $x = 400$ dan $y = 0$.



Gambar 2.13 Gambaran proses penentuan sudut tembakan ke gawang

Dengan mendapatkan koordinat robot, dan koordinat gawang, robot dapat menentukan sudut ke gawang untuk melakukan tendangan. Persamaan yang digunakan dalam perhitungan sudut ke gawang adalah sebagai berikut

$$\theta = \tan^{-1} \left(\frac{yg - yr}{xg - xr} \right) \quad (2.1)$$

Yg dan xg adalah koordinat x dan y gawang, sedangkan xr dan yr adalah koordinat x dan y robot. Dengan cara ini, robot hampir akan selalu mengarahkan bola ke tengah gawang tanpa mempedulikan adanya halangan lain didepannya. Untuk itu dalam beberapa kasus tertentu, koordinat gawang digeser ke kiri atau ke kanan untuk membuat sudut tendangan lebih bervariasi.

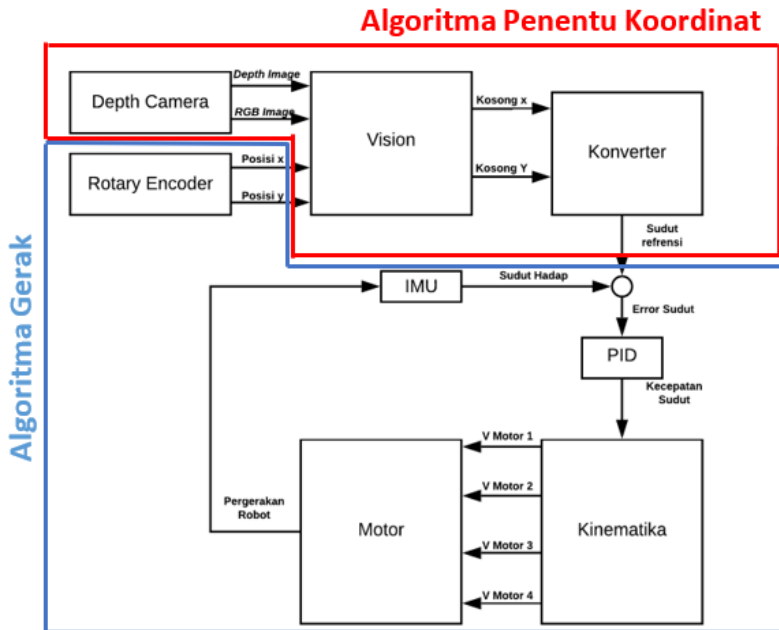
Halaman ini sengaja dikosongkan

BAB 3

PERANCANGAN SISTEM

3.1 Perancangan Sistem

Sistem secara keseluruhan dapat digambarkan pada Gambar 3.1. sistem akan dibagi menjadi 2 bagian yaitu Algoritma Penentu Koordinat dan Algoritma gerak. Sesuai namanya, algoritma penentu koordinat berfungsi untuk menentukan koordinat tendangan robot. Dalam algoritma ini, akan digunakan input berupa data yang didapat dari proses *vision*, yaitu data citra RGB dan citra *Depth* dari Intel Realsense D435i, yang kemudian diolah agar dapat menghasilkan nilai koordinat lapangan. Sedangkan Algoritma gerak adalah susunan proses yang berfungsi untuk mengarahkan robot ke arah koordinat lapangan yang ditunjuk. Algoritma gerak menerima input berupa θ_{ref} atau sudut referensi antara robot dan koordinat yang dituju. Fokus dari penelitian ini adalah penentuan koordinat kosong gawang. Untuk itu akan difokuskan pada bagian algoritma penentu koordinat

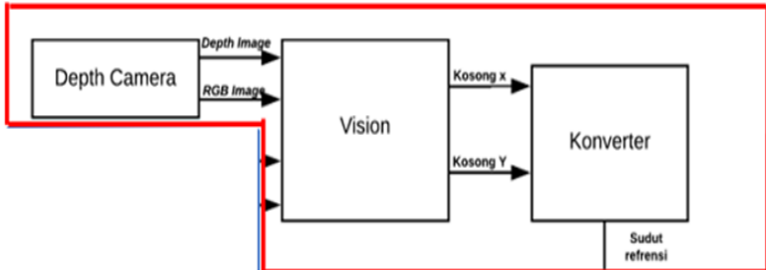


Gambar 3.1 Blok Diagram Sistem

3.1.1 Algoritma Penentu Koordinat

Seperti yang telah dijelaskan sebelumnya, Algoritma ini berfungsi untuk menentukan koordinat sasaran tendang robot. Dalam proses penentuan koordinat, algoritma ini dibagi menjadi 3 bagian bagian yaitu *Depth camera*, *vision* dan konverter. Blok *Depth camera* merupakan tahap *pre processing* gambar, yang meliputi pengaturan parameter kamera, serta pengambilan dan penyesuaian gambar. Gambar dari *depth camera* akan menjadi input ke blok *Vision* untuk dilakukan pencarian area kosong dan koordinat *pixel* nya, sedangkan blok konverter berfungsi untuk mengubah koordinat *pixel* yang didapat dari blok *vision* menjadi koordinat lapangan.

Algoritma Penentu Koordinat

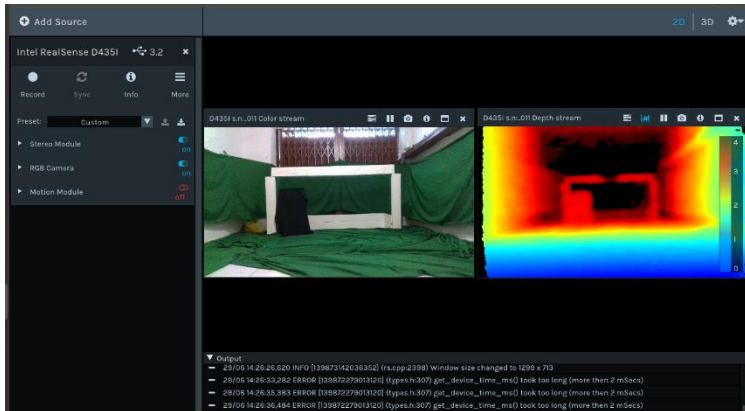


Gambar 3.2 Blok Diagram Algoritma Penentu Koordinat

3.1.1.1 Blok *Depth camera*

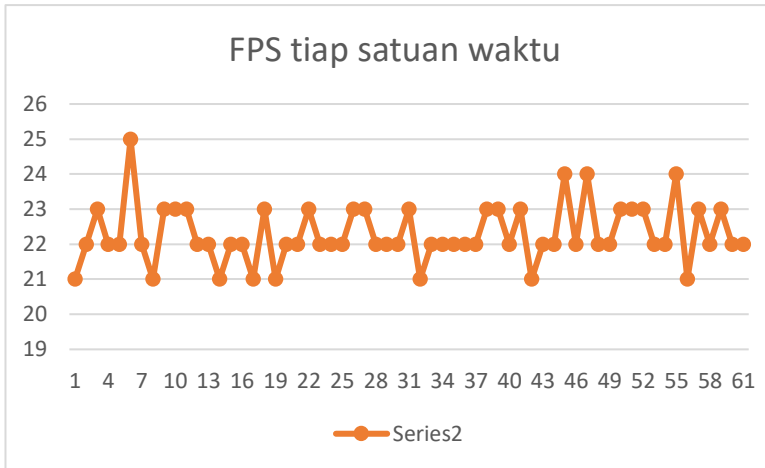
Dalam rangkaian proses untuk menentukan area kosong gawang, langkah pertama yang dilakukan berada pada blok *Depth camera*. Blok ini menjelaskan proses pengaturan kamera dan *pre processing* gambar. Kedua proses ini diperlukan agar proses pada blok *vision* dapat menentukan gawang dengan baik.

Intel Realsense D435i memiliki lebih dari 40 parameter yang dapat menentukan hasil dari gambar yang ditangkap [13]. Namun hubungan antar parameter ini sangat kompleks, sehingga pada penelitian ini akan digunakan pilihan *Depth preset*, yaitu kumpulan parameter yang telah disesuaikan dalam berbagai kondisi. *Preset* yang digunakan adalah *High Density* dengan nilai parameter *exposure* 15000. *Preset high density* digunakan karena pengaturan ini sangat sesuai untuk melakukan pengenalan objek[14]. Sedangkan nilai *exposure* disesuaikan berdasarkan kondisi pencahayaan pada ruang uji.



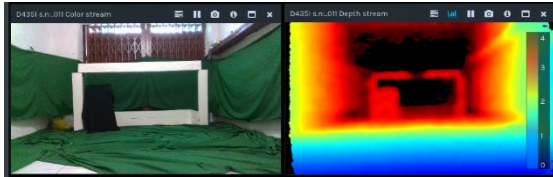
Gambar 3.3 Tampilan Realsense-Viewer Untuk Mengatur Parameter kamera

Selanjutnya adalah pengaturan parameter frame gambar. Intel Realsense D435i memiliki ukuran frame optimal 848 X 480. Namun, pada penelitian ini digunakan resolusi frame 424 X 240, untuk mengurangi beban komputasi. Dengan pengurangan ini, didapatkan rata-rata kecepatan pemrosesan frame sebesar 22 FPS

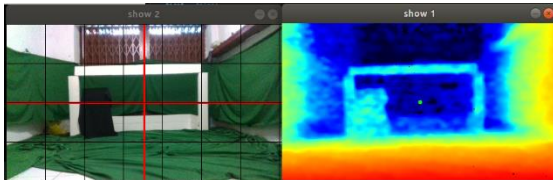


Gambar 3.4 Grafik Perubahan FPS Tiap Satuan Waktu

Setelah melakukan pengaturan pada parameter kamera, selanjutnya adalah melakukan *Pre-Processing* pada gambar yang telah ditangkap. Tahap *Pre-Processing* ini meliputi penyesuaian *Field of Fiew (FOV)* citra kedalaman dan citra RGB, agar kedua gambar ini memiliki koordinat *pixel* yang sesuai.



Gambar 3.5 Perbandingan *Field of View* Sebelum proses



Gambar 3.6 Perbandingan *Field of View* Setelah proses

Dapat dilihat pada Gambar 3.5 bahwa citra *depth* yang didapat memiliki area yang lebih luas dibanding dengan citra RGB. Perbedaan luas ini akan menyebabkan terjadinya perbedaan koordinat *pixel* pada setiap objek yang ditangkap. Untuk itu dilakukan proses *align* untuk menyamakan *field of view* dari gambar, dan didapatkan hasil seperti Gambar 3.6. Dengan proses ini, koordinat pada citra RGB dan *Depth* akan sama di setiap *pixel*.

3.1.1.2 Blok Vision

Pada blok *vision* akan dijelaskan sistem pengambilan gambar melalui kamera, dan proses yang dilakukan untuk mendapatkan data koordinat 3D pada area kosong gawang sebagai acuan tendangan. Untuk dapat mengetahui koordinat 3D area gawang, sistem memerlukan *depth image* dan *RGB image* yang diambil dari *depth camera*. *Block diagram* untuk proses vision ini digambarkan pada Gambar 3.7

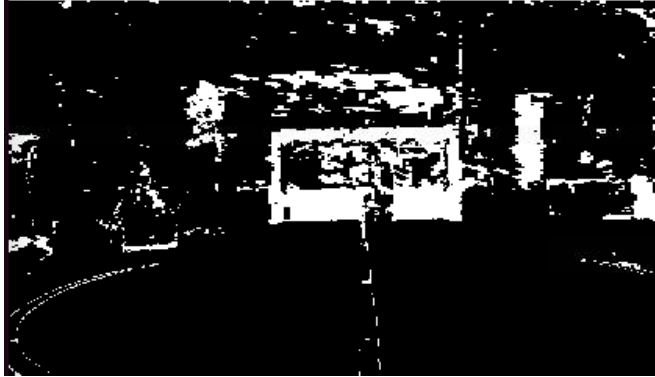


Gambar 3.7 Block Diagram proses vision

Untuk dapat melakukan deteksi gawang dan penjaga gawang, sistem akan terlebih dahulu memilih warna dan jarak sebagai penanda. Hal ini dilakukan untuk mengurangi *noise* yang didapat dari masing-masing gambar. Seperti yang terlihat pada Gambar 3.9 dan Gambar 3.10, jika hanya mengandalkan salah satu penanda, *noise* yang didapat dari masing-masing gambar akan terlalu besar dan menyebabkan adanya kesalahan dalam pendeteksian objek. Gambar 3.11 menunjukkan hasil penggabungan Gambar 3.9 dan Gambar 3.10



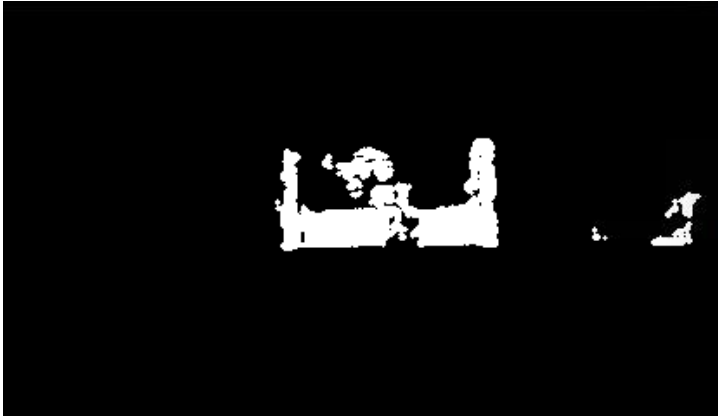
Gambar 3.8 RGB Image



Gambar 3.9 Pemilihan Objek berwarna putih



Gambar 3.10 Pemilihan objek dengan jarak relatif dari robot ke gawang



Gambar 3.11 Hasil Akhir Gawang

Dengan proses ini, pendeteksian gawang menjadi lebih mudah karena gawang dapat didefinisikan sebagai objek berwarna putih dengan jarak rg , sedangkan penjaga gawang didefinisikan sebagai objek berwarna hitam dengan jarak rk . Jarak pendeteksian ditentukan berdasarkan posisi robot dan gawang yang dihitung berdasarkan persamaan

$$rg = \sqrt{(xr - xg)^2 - (yr - yg)^2} \quad (3.1)$$

$$rk = rg + offset \quad (3.2)$$

dimana xr dan yr adalah koordinat x dan y robot, sedangkan xg dan yg adalah koordinat x dan y gawang. Dikarenakan posisi penjaga gawang yang sangat dekat dengan koordinat gawang, maka koordinat penjaga gawang didefinisikan sebagai koordinat gawang ditambahkan dengan sedikit pergeseran (*offset*) yang didefinisikan dalam program

Dengan didapatkannya posisi gawang dan penjaga gawang, sistem dapat memperkirakan area kosong pada gawang sebagai acuan tendangan. Proses ini dilakukan dengan mengurangi area yang terdeteksi sebagai gawang, dengan area yang terdeteksi sebagai penjaga gawang. Dengan demikian akan ditemukan area gawang yang tidak dijaga seperti pada

Gambar 3.12. Dengan kotak hijau menandakan area kosong, kotak biru menandakan gawang, dan merah sebagai penjaga gawang



Gambar 3.12 Deteksi Area Kosong pada Gawang

3.1.1.3 Blok Konverter

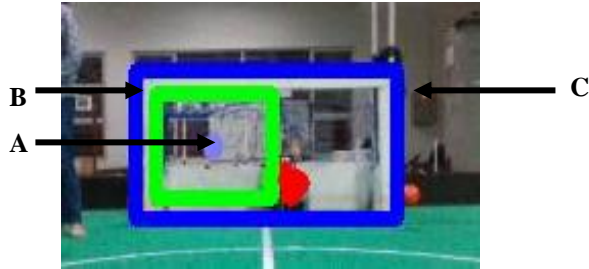
Data koordinat *pixel* yang didapat oleh bagian *vision* akan diteruskan pada blok konverter. Blok ini akan memproses nilai koordinat *pixel* yang didapat menjadi koordinat lapangan. Untuk itu, koordinat yang didapat akan dikonversi menjadi posisi 3 dimensi relatif terhadap posisi *depth camera*. Posisi ini didapatkan dengan persamaan sebagai berikut

$$Z = \text{depth}$$

$$X = Z (px - ppx)/fx \quad (3.3)$$

$$Y = Z (py - ppy)/fy$$

Dengan px dan py adalah koordinat *pixel* dari gambar yang ditangkap, ppx dan ppy adalah koordinat *pixel principal point* (titik tengah proyeksi), serta fx dan fy adalah *focal length* dari gambar yang diambil. Nilai X,Y,Z mengacu pada sistem koordinat yang telah dijelaskan pada sub bab 2.2.2



Gambar 3.13 Gambaran Deteksi Gawang

Pada Gambar 3.13, koordinat *pixel* yang dikonversi adalah titik berwarna biru muda di tengah kotak hijau (Panah A). Dengan *depth* yang digunakan selalu bernilai konstan sebesar 1200, hal ini dilakukan karena nilai *depth* pada posisi panah A tidak dapat mencerminkan jarak sesungguhnya karena selalu berada di belakang gawang.

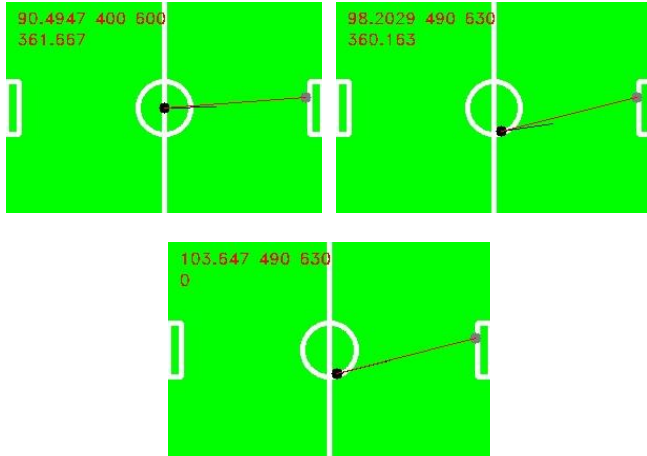
Langkah selanjutnya adalah menentukan posisi relatif batas gawang terhadap posisi intel realsense D435i, yang dimaksud sebagai batas gawang adalah bagian paling kanan dan paling kiri dari area gawang (Gambar 3.13 panah B dan panah C). ketiga titik ini akan dikonversi menjadi posisi relatif dengan persamaan $pg - pmin \cdot pmax - pmin + 300$ (3.4)

Ketiga nilai posisi ini akan dikonversi menjadi koordinat lapangan yang dapat ditulis dengan persamaan berikut

$$\left(\frac{pg - pmin}{pmax - pmin} \right) + 300 \quad (3.4)$$

Dengan *pg* adalah posisi relatif gawang, *pmax* adalah posisi relatif batas kanan gawang, *pmin* adalah posisi relatif batas kiri gawang dan nilai 300 adalah posisi x batas kiri gawang pada koordinat lapangan.

Dengan demikian, nilai yang didapat dari perhitungan kamera akan selalu berada didalam batas gawang. Untuk memberikan gambaran area kosong yang didapat, akan digunakan permodelan lapangan pada Gambar 3.14



Gambar 3.14 Contoh Penggambaran Posisi Area Kosong yang Didapat Oleh sistem

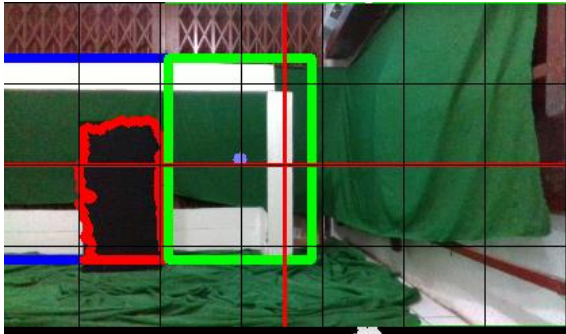
3.1.2 Algoritma Gerak

Pada sub bab 3.1.1 telah dijelaskan proses pengambilan data vision dan langkah-langkah yang diperlukan untuk melakukan konversi dari data vision yang didapat menjadi data koordinat lapangan. Namun untuk dapat melakukan tendangan yang berpotensi untuk mencetak goal, perlu digunakan suatu algoritma tertentu yang berfungsi untuk mengarahkan robot menuju titik yang telah didapat. Algoritma tersebut dapat digambarkan dengan blok diagram pada gambar berikut



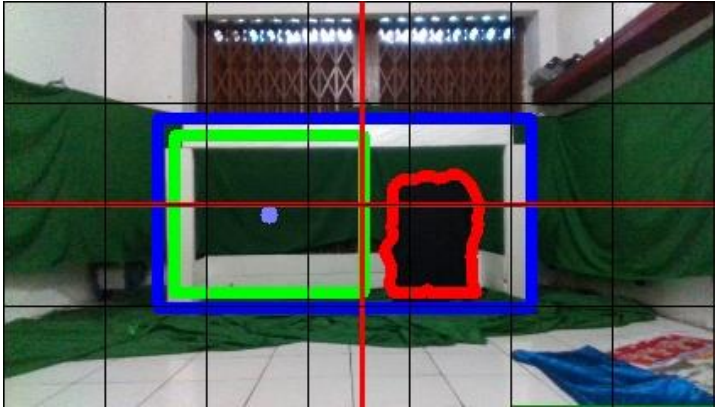
Gambar 3.15 Flow Chart Algoritma Tendang

Seperti yang telah diperlihatkan dalam gambar, sistem akan terlebih dahulu mengarahkan robot ke tengah gawang. Proses ini dilakukan untuk memastikan bahwa objek yang ditangkap oleh kamera benar-benar objek yang diinginkan. Jika tidak dilakukan proses ini, besar kemungkinan sistem akan mendeteksi objek lain selain gawang, atau memiliki interpretasi yang salah atas posisi gawang.

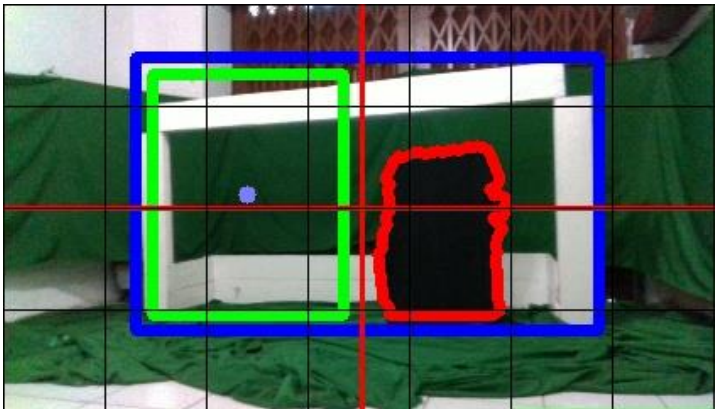


Gambar 3.16 Interpretasi yang Tidak Sesuai (Gambar Gawang terpotong)

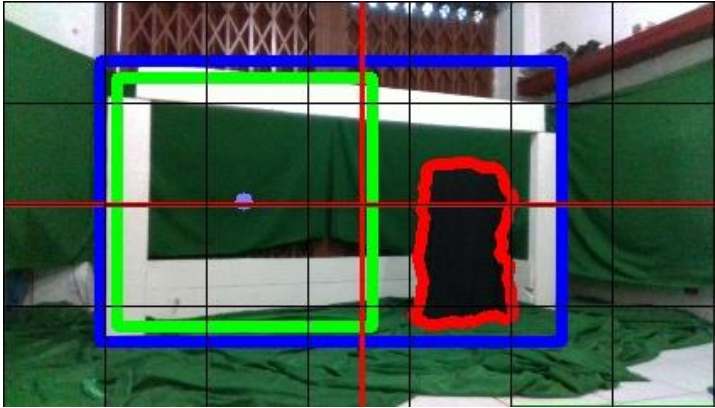
Dengan mengarahkan robot tepat ke tengah koordinat gawang, maka gambar interpretasi gawang akan memiliki tingkat ketepatan yang cukup tinggi seperti pada contoh berikut



Gambar 3.17 Interpretasi Posisi gawang di Depan Gawang



Gambar 3.18 Interpretasi Posisi gawang di Kanan Gawang



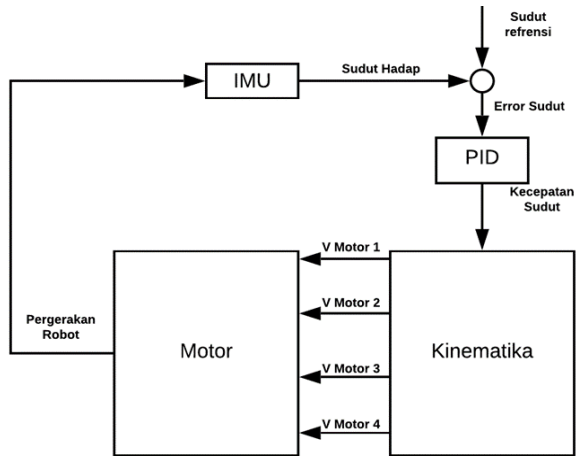
Gambar 3.19 Interpretasi Posisi gawang di Kiri Gawang

Saat robot telah mengarah tepat ke tengah gawang, sistem akan menyimpan koordinat *pixel* area kosong gawang untuk kemudian dikonversi menjadi koordinat lapangan dengan proses yang telah tergambar pada blok diagram Gambar 3.15 Flow Chart Algoritma Tendang. proses ini diperlukan untuk menentukan arah hadap robot yang sesuai dengan koordinat yang didapat.

Sistem pergerakan dan penentuan sudut robot menggunakan *close loop control system* yang akan dijelaskan lebih lanjut pada sub bab 3.2. sistem ini membutuhkan error maksimum antara sudut hadap robot, dan sudut target sebesar 0.5° . jika nilai error yang didapat kurang dari error maksimum, sistem akan memberikan perintah tendang, untuk melakukan tendangan pada arah yang dituju.

3.2 Kontrol Pergerakan

Sistem yang dibuat dalam penelitian ini sangat mementingkan akurasi dari sudut hadap robot. Oleh karena itu diperlukan adanya proses yang berfungsi untuk melakukan kontrol dari pergerakan robot khususnya kontrol pada sudut. Seperti halnya sistem kontrol pada umumnya, proses ini memerlukan *close loop control system* agar mendapatkan akurasi yang tinggi. *close loop control system* yang digunakan adalah sistem kontrol PID. Dengan sensor IMU sebagai *feedback* pada sistem. Blok diagram dari sistem tersebut dapat digambarkan sebagai berikut



Gambar 3.20 Blok Diagram Kontrol Robot

Tabel 3.1 Parameter PID

Parameter	Nilai
KP Posisi	0.4
KI Posisi	0.1
KD Posisi	0.2
KP Sudut	0.7
KI Sudut	0.1
KD Sudut	0.1

Dalam diagram tersebut, nilai keluaran PID akan dimasukkan pada blok kinematika. Blok ini berfungsi untuk mengubah nilai θ yang merupakan output dari PID menjadi nilai kecepatan dari masing-masing motor. Setiap nilai perubahan sudut akan dibaca oleh sensor IMU untuk menjadi feedback pada blok PID

3.2.1 Sensor IMU

IMU (*inertia measurement unit*) adalah suatu sensor yang digunakan untuk mengukur perubahan sudut dari suatu objek. Sensor ini memiliki 2 bagian yaitu *accelerometer* dan *gyrometer*. *Accelerometer* digunakan untuk mengukur percepatan dari objek. Sedangkan *Gyrometer* digunakan untuk mengukur kecepatan sudut dari objek.

Dalam penggunaan *gyrometer*, dikenal istilah yang melambangkan arah perubahan sudut robot. Istilah tersebut adalah *yaw*, *pitch*, dan *roll*. Masing-masing istilah ini menggambarkan arah perubahan sudut robot pada suatu sumbu koordinat tertentu. Mengacu pada sub bab 2.2.2, *yaw* didefinisikan sebagai perubahan sudut pada sumbu Y, *roll* adalah perubahan pada sumbu Z, sedangkan *pitch* adalah perubahan pada sumbu X.

Penelitian ini menggunakan *built-in IMU* yang tersedia dalam *depth camera* intel Realsense D435i. IMU pada intel Realsense d435i memiliki 6 Dof (*Degree of Freedom*) yang menandakan bahwa sensor ini memiliki kemampuan untuk menghitung percepatan dari 3 sumbu (X, Y, Z) dan melakukan perhitungan *yaw*, *pitch*, dan *roll*.

Karena dalam penelitian ini jenis robot yang digunakan adalah robot yang bergerak di permukaan datar, arah hadap robot pun menjadi terbatas pada bagian *yaw* saja. Untuk itu nilai pembacaan IMU yang diambil dari *depth camera* hanya *gyrometer* pada bagian *yaw*, tanpa mengambil nilai percepatan dari *accelerometer*.

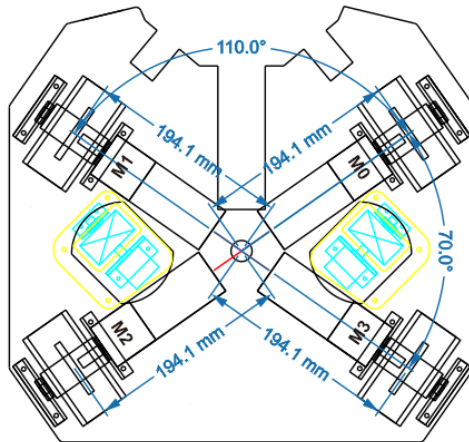
Seperti yang telah dijelaskan sebelumnya, nilai yang diambil dari *gyrometer* adalah nilai kecepatan sudut. Sedangkan untuk melakukan perhitungan error, nilai yang dibutuhkan adalah nilai perubahan sudut. Persamaan ini digunakan untuk mendapat nilai perubahan sudut

$$\int \omega_{yaw} dt \quad (3.5)$$

Nilai perubahan sudut didapat dengan melakukan integral pada nilai kecepatan sudut (ω_{yaw}). Dalam penerapannya pada program, digunakan iterasi penjumlahan tiap satuan waktu

3.2.2 Kinematika dan Motor

Robot yang digunakan dalam penelitian ini adalah robot dengan sistem penggerak 4 buah motor. Semua pergerakan robot akan ditentukan dari kombinasi kecepatan dari 4 buah motor tersebut. Untuk itu, semua perintah kecepatan rotasi dan translasi robot perlu untuk dikonversi menjadi perintah kecepatan tiap motor. Proses konversi ini lah yang dilakukan oleh blok kinematika.



Gambar 3.21 Konfigurasi posisi motor pada base robot

Gambar 3.21 Menggambarkan konfigurasi sudut dan peletakan motor pada *base* robot. Untuk dapat melakukan konversi kecepatan translasi dan rotasi robot digunakan persamaan berikut

$$\begin{bmatrix} \dot{\theta}m0 \\ \dot{\theta}m1 \\ \dot{\theta}m2 \\ \dot{\theta}m3 \end{bmatrix} = \begin{bmatrix} -\sin(35^\circ) & \cos(35^\circ) & 1 \\ -\sin(145^\circ) & \cos(145^\circ) & 1 \\ -\sin(-35^\circ) & \cos(-35^\circ) & 1 \\ -\sin(-145^\circ) & \cos(-145^\circ) & 1 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \quad (3.6)$$

3.2.3 Rotary Encoder

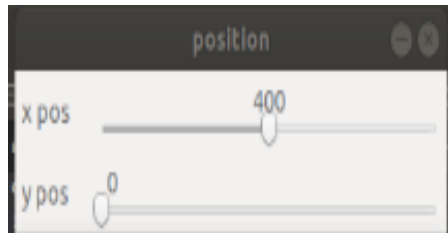
Seperti yang telah dijelaskan pada sub bab 2.5, *rotary encoder* adalah sensor yang digunakan untuk melakukan perhitungan jumlah

putaran dari suatu roda. Robot IRIS memiliki 2 buah *rotary encoder* yang memiliki sudut 90° . Hasil pembacaan dari kedua *rotary encoder* ini akan dikonversi menjadi pembacaan posisi dengan persamaan sebagai berikut

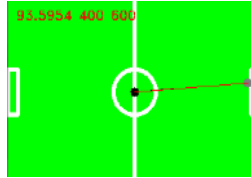
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\theta + 45^\circ) & \cos(\theta + 135^\circ) \\ \sin(\theta + 45^\circ) & \sin(\theta + 135^\circ) \end{bmatrix} \begin{bmatrix} RE0 \\ RE1 \end{bmatrix} \quad (3.7)$$

dalam persamaan tersebut digunakan θ yang didapat dari *IMU* untuk melakukan perubahan posisi robot berdasarkan arah hadapnya

Dikarenakan adanya kendala saat melakukan penelitian ini, beberapa sistem pada robot tidak dapat diakses, dan harus diganti dengan menggunakan sistem lain. Salah satu sistem yang terdampak adalah *rotary encoder*. Untuk itu, agar sistem pendeteksiian gawang dapat tetap menerima posisi robot, *rotary encoder* diganti dengan menggunakan *trackbar* yang merupakan salah satu fitur dari library OpenCV. Berikut adalah tampilan dari *trackbar* yang digunakan



Gambar 3.22 *Trackbar* Sebagai Input Posisi



Gambar 3.23 Interpretasi Posisi Robot (Titik Hitam) Berdasarkan *Trackbar*

3.3 Desain mekanik robot

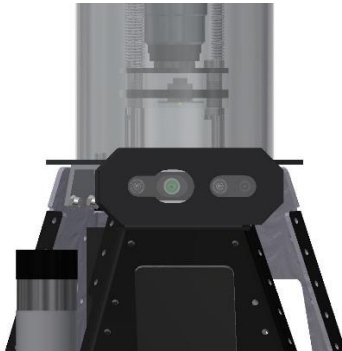
Robot pada penelitian ini didesain untuk dapat berfungsi dalam kondisi perlombaan, dimana kontak fisik antar robot (baik satu tim maupun dengan tim lain) sangat mungkin terjadi. Terlebih, tumbukan dengan kecepatan tinggi antar robot akan memiliki potensi cukup besar untuk merusak sistem pada robot. Oleh karena itu, desain mekanik robot dibuat dengan memperhitungkan adanya kontak fisik yang cukup keras antar robot.

Bahan yang digunakan sebagai desain pelindung luar dan *base* pada robot adalah *stainless-steel*. Campuran logam ini dipilih sebagai bahan utama pembentuk mekanik karena tingkat kekerasannya yang cukup tinggi, tidak mudah patah maupun bengkok.

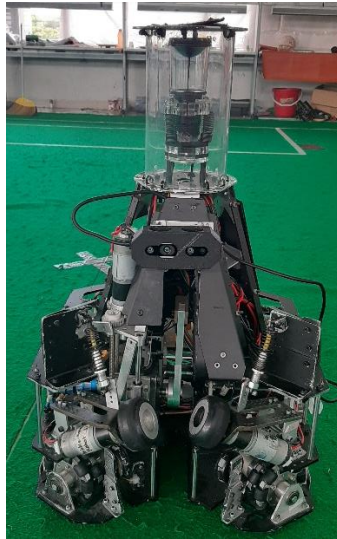
Keseluruhan bagian dari mekanik robot dibentuk menggunakan mesin *cnc laser cutting* agar memiliki tingkat akurasi tinggi. Hal ini sangat diperlukan untuk memastikan semua sistem robot dapat berjalan dengan baik, terutama motor dan *rotary encoder* yang sangat dipengaruhi oleh sudut peletakan



Gambar 3.24 Desain Robot



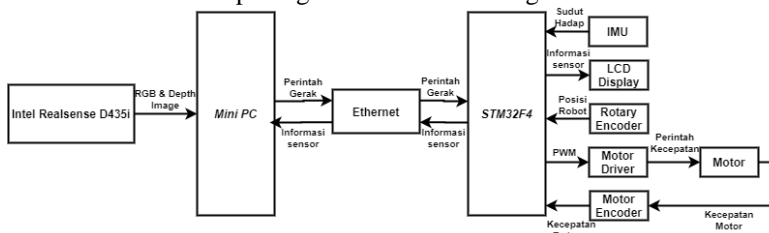
Gambar 3.25 Posisi *Depth Camera* pada Robot



Gambar 3.26 Realisasi Robot

3.4 Desain elektronik robot

Data yang diperoleh oleh sensor *rotary encoder* dan *IMU*, memerlukan adanya *microcontroller* untuk melakukan pengolahan data sebelum data dari masing-masing sensor dikirim ke *Mini PC* sebagai sistem utama dalam penelitian ini. Komunikasi antara *microcontroller*, sensor dan *Mini PC* dapat digambarkan dalam diagram berikut



Gambar 3.27 Diagram Komunikasi Sistem Elektronik

- Intel Realsense D435i merupakan sensor utama yang digunakan pada penelitian ini. Dengan sensor ini, sistem dapat melakukan deteksi koordinat area kosong

- Mini PC berfungsi sebagai perangkat pemroses utama dalam keseluruhan sistem.
- Ethernet digunakan untuk melakukan pertukaran informasi antara mini pc dan intel realsense d435i
- STM32F4 merupakan *Microcontroller* utama yang digunakan untuk menggerakkan aktuator dan membaca informasi dari sensor
- IMU merupakan perangkat sensor yang digunakan untuk melakukan deteksi arah hadap robot
- *LCD Display* digunakan untuk menampilkan hasil pembacaan sensor
- *Rotary Encoder*, seperti yang telah dijelaskan pada sub bab 3.2.3, merupakan perangkat yang digunakan untuk melakukan deteksi posisi robot
- *Motor Driver* digunakan untuk mengatur arah gerak motor beserta kecepatannya. Sedangkan *Motor Encoder* digunakan untuk mengetahui kecepatan motor yang dihasilkan

Realsasi dari diagram rangkaian tersebut, dapat direpresentasikan dengan skema pada gambar berikut. Skema ini hanya menunjukkan keterhubungan pin untuk setiap pada STM32F4 saja. Namun diagram keseluruhan sistem dapat dilihat pada lampiran

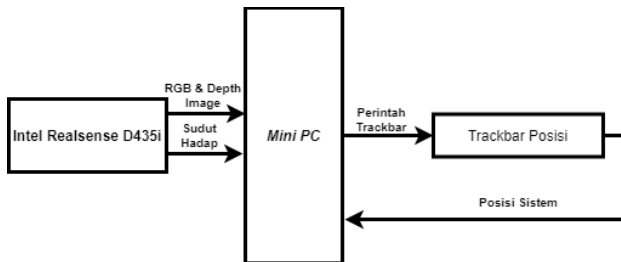
STM32F4-DISCOVERY BREAKOUT



P31	GND	GND1	P32		P351	GND6	GND7	P352	GND
P33	VDD	VDD1	P34	P34	P353	3V	3V2	P354	VCC
P35	GND2	NRST	P36	RESET	P355	VDD2	VDD3	P356	
	MDC	PC0	P38		P357	PH0	PH1	P358	
	PC1	PC0	P38	PHOTO_2		PH2	PH3	P359	
	PC2	PC2	P39	RS		PH4	PH5	P360	DIR2B
	PA1	PA0	P40	MDIO		PH6	PH7	P361	DIR1A
	PA2	PA2	P41	MDIO		PH8	PH9	P362	DIR1A
	PA3	PA3	P42	MDIO		PH10	PH11	P363	ENC_ODD02
	P315	PA5	PA4	P315GP	RD1A		PE0	P366	
	CRS_DV	PA7	PA6	P316	PWM4		PE1	P368	
	EVT	PC5	PC4	P320	RX0		PE2	P370	SHOT
	GARIS3	P321	P30	P322GARIS2			PE3	P372	
		GND3	P32	P324	DIR1A		PE4	P374	ENC_ODD1R
	DIR1R	P323	P32	P326	DIR2A		PE5	P378	ENC3R
	ENC1A	P327	P32	P328	DIR2B		PE6	P379	ENC2B
	ENC1B	P329	P32	P330S			PE7	P380	DIR4R
	D7	P331	P32	P332D4			PE8	P382	DIR4A
	D8	P333	P32	P334TX_USART3			PE9	P384	S_1
	TX_EN	P335	P32	P336	TX0		PE10	P385	S_1
	TX1	P337	P32	P338	PPR11		PE11	P387	S_1
	ENABLE	P339	P32	P340	S		PE12	P389	ENC2A
	RX_USART3	P341	P32	P342	S		PE13	P392	
		P343	P32	P344	7		PE14	P394	RX_USART1
	GP2	P347	P32	P346	ENC5B		PE15	P396	S_1
		GND4	GND5	P350			PE16	P398	ENC4A
							PE17	P3100	
							PE18		
							PE19		
							PE20		
							PE21		
							PE22		
							PE23		
							PE24		
							PE25		
							PE26		
							PE27		
							PE28		
							PE29		
							PE30		
							PE31		
							PE32		
							PE33		
							PE34		
							PE35		
							PE36		
							PE37		
							PE38		
							PE39		
							PE40		
							PE41		
							PE42		
							PE43		
							PE44		
							PE45		
							PE46		
							PE47		
							PE48		
							PE49		
							PE50		
							PE51		
							PE52		
							PE53		
							PE54		
							PE55		
							PE56		
							PE57		
							PE58		
							PE59		
							PE60		
							PE61		
							PE62		
							PE63		
							PE64		
							PE65		
							PE66		
							PE67		
							PE68		
							PE69		
							PE70		
							PE71		
							PE72		
							PE73		
							PE74		
							PE75		
							PE76		
							PE77		
							PE78		
							PE79		
							PE80		
							PE81		
							PE82		
							PE83		
							PE84		
							PE85		
							PE86		
							PE87		
							PE88		
							PE89		
							PE90		
							PE91		
							PE92		
							PE93		
							PE94		
							PE95		
							PE96		
							PE97		
							PE98		
							PE99		

Gambar 3.28 Skema PinOut STM32F4 Discovery

Dalam diagram tersebut, dapat dilihat bahwa sistem elektronik robot memiliki *IMU* tersendiri, dan tidak perlu mengakses *IMU* pada *depth camera* Intel Realsense D435i. Namun seperti halnya yang telah dijelaskan dalam sub bab 3.2.3, adanya kendala dalam melakukan penelitian menyebabkan sebagian besar sensor tidak dapat diakses oleh peneliti. Untuk itu digunakan sistem elektronik pengganti yang lebih sederhana, agar sistem dapat berjalan dengan baik. Sistem elektronik pengganti dapat digambarkan dalam diagram berikut.



Gambar 3.29 Blok Diagram Elektronik Pengganti

Halaman ini sengaja dikosongkan

BAB 4

PENGUJIAN

Pengujian dilakukan untuk mengetahui tingkat keandalan sistem yang telah dibuat. Dikarenakan sistem pada penelitian ini cukup kompleks, pengujian akan dilakukan dalam beberapa tahapan. Pengujian yang akan dilakukan adalah uji penentuan koordinat, pengujian jarak pada 2 titik, Uji pendeteksian gawang, dan terakhir, pengujian jumlah goal.

Dikarenakan pengujian erat kaitannya dengan pengukuran jarak, akan digunakan alat pengukuran berupa *laser rangefinder*, sebuah alat untuk mengukur jarak suatu objek dengan menggunakan sinar inframerah. Alat yang digunakan adalah xiaomi DUKA LS-P dengan akurasi jarak mencapai 1 mm, dan mampu mengukur jarak hingga 50 m



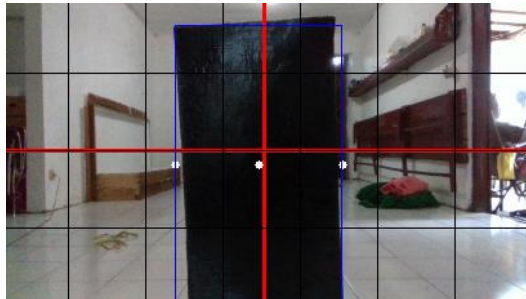
Gambar 4.1 Xiaomi DUKA LS-P

4.1 Uji Penentuan Koordinat

Pengujian ini adalah pengujian pertama dan merupakan pengujian paling mendasar. Hal ini dikarenakan sistem akan bergantung kepada kemampuan Intel Realsense D435i sebagai *depth camera* untuk memperkirakan koordinat objek. Objek yang digunakan adalah objek berwarna hitam dengan tinggi 80 cm dan lebar 43cm.

Pengukuran objek dilakukan dengan cara yang mirip dengan algoritma pemilihan gawang. Sistem hanya akan mendeteksi objek dengan warna dan jarak yang sesuai, untuk selanjutnya melakukan perhitungan koordinat pada pixel yang telah dipilih.

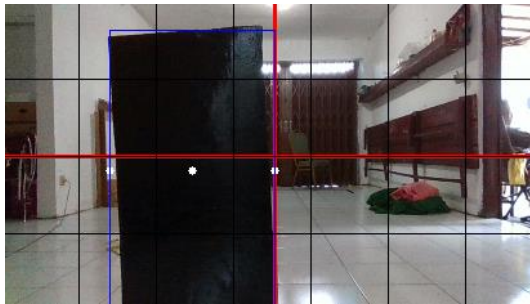
Pengujian dilakukan dengan Batasan nilai horizontal (sumbu X Intel Realsense D435i) sebesar -100 cm hingga 100 cm (dengan nilai negatif menandakan objek berada di sebelah kiri kamera, dan positif menandakan bahwa objek berada di sebelah kanan kamera) dan Batasan jarak / (sumbu Z Intel Realsense D435i) mulai dari 100 cm hingga 600 cm. Pengukuran horizontal akan dilakukan setiap 20cm dan pengukuran jarak akan dilakukan setiap 100 cm.



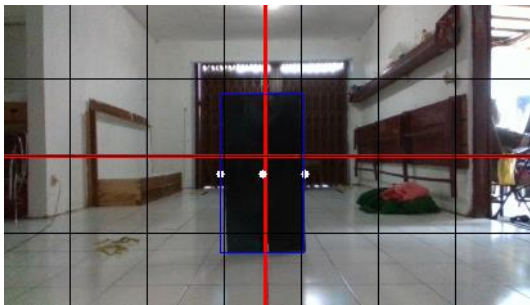
Gambar 4.2 Pengujian Koordinat



Gambar 4.3 Pergeseran 20 cm ke Kanan



Gambar 4.4 Pergeseran 20 cm ke Kiri



Gambar 4.5 Pergeseran Jarak 100 cm

Tabel 4.1 Hasil Pengukuran Jarak

Pembagian jarak	jarak laser	x sesungguhnya	x realsense	z realsense	Error x (%)	Error z (%)
100	101.9	0	-0.32	101.2		0.69
		20	20.7	100.6	3.5	1.28
		40	39.3	101.4	1.7	0.49
		60				
		80				
		100				
		-20	-20.3	102.3	1.5	0.39
		-40	-40.1	101.4	0.2	0.49
		-60				
		-80				
		-100				
300	301	0	1.9	300.5		0.17
		20	21.5	303.2	7.5	0.73
		40	41.3	301.9	3.2	0.30
		60	61.3	299.2	2.1	0.60
		80	81.8	299.2	2.2	0.60
		100	97.9	294.1	2.1	2.29
		-20	-18.50	299.20	7.5	0.60
		-40	-39.6	297.9	1	1.03
		-60	-59.8	301.9	0.3	0.30
		-80	-82.1	300.5	2.6	0.17
		-100	-98.1	315.8	1.9	4.92
500	497.2	0	1.6	503.9		1.35
		20	24.4	500.2	22	0.60
		40	42.3	503.9	5.7	1.35
		60	62.9	496.6	4.8	0.12
		80	83.7	511.4	4.6	2.86
		100	103.4	507.6	3.4	2.09
		-20	-16.6	507.6	17	2.09
		-40	-39.6	496.6	1	0.12
		-60	-60.3	503.9	0.5	1.35
		-80	-78.2	489.5	2.2	1.55
		-100	-112.5	523.1	12.5	5.21

Pada gambar 4.2 sampai dengan 4.5, objek yang diukur ditunjukkan dengan kotak biru. Nilai titik tengah *pixel* pada kotak biru ini yang akan dilakukan perhitungan nilai koordinat objek. Hasil pengujian untuk jarak 100 cm, 300 cm, dan 500 cm akan disajikan pada tabel berikut, sedangkan hasil pengujian keseluruhan akan disajikan dalam lampiran

Pengujian hanya dapat dilakukan dengan 60 titik karena 6 titik pada jarak 100 cm tidak memungkinkan untuk dilakukan pengujian karena keterbatasan *Point of View* dari kamera (ditandai dengan bagian merah).

Data ini disediakan dalam 7 kolom, yaitu pembagian jarak, jarak laser, x sesungguhnya, x realsense, z realsense, dan error x serta error z . Pembagian jarak adalah jarak peletakan objek dari kamera. Namun, terkadang peletakan objek memiliki selisih jarak dengan nilai yang diinginkan. Untuk dapat melakukan pengukuran dengan akurasi lebih tinggi, ditambahkan variabel jarak laser, yang merupakan jarak yang terbaca oleh *laser rangefinder xiaomi LS-P* sebagai pembanding. Variabel x sesungguhnya adalah nilai koordinat x peletakan objek, sedangkan variabel x realsense dan z realsense adalah koordinat x dan z yang terbaca oleh *depth camera*. Variabel error x dan error z ditampilkan untuk memberi gambaran error pembacaan nilai x dan z dalam persen. Dikarenakan adanya nilai 0 pada variabel x sesungguhnya, beberapa nilai realsense x tidak dapat dilakukan perhitungan error (ditandai dengan bagian merah pada kolom error x)

Nilai error pada sumbu x didapat dengan mengurangkan variabel x realsense, dengan variabel jarak sesungguhnya, sedangkan error pada sumbu z didapat dengan mengurangkan variabel z realsense dengan jarak laser. Hasil pengukuran didapat rata-rata presentase error nilai sumbu x sebesar 5.7%. sedangkan nilai sumbu z memiliki rata-rata presentase error sebesar 1.49%. Kenaikan error yang signifikan terjadi mulai jarak antara kamera dan objek sejauh 300 cm hingga 600 cm.

4.2 Pengujian Jarak Pada 2 Titik

Dalam sub bab 3.1.1 telah dijelaskan proses penentuan koordinat kosong pada gawang. Proses ini sangat bergantung pada kemampuan kamera untuk dapat memperkirakan jarak antara 2 titik. Pengujian ini dilakukan untuk mengetahui tingkat ketepatan sistem dalam memperkirakan jarak dari 2 titik tersebut.

Pengujian ini menggunakan objek yang sama dengan pengukuran 4.1, namun dikarenakan adanya kondisi cahaya yang berbeda

menyebabkan pendeteksian warna hitam menjadi lebih rumit. Untuk itu warna objek diganti dengan warna biru.

Pengujian dilakukan dengan cara melakukan *threshold* dengan warna dan jarak yang sesuai untuk menyeleksi objek dari *background*. Kemudian memberikan tanda berupa *bounding box* pada objek. Titik yang diambil adalah titik paling kiri dan paling kanan *bounding box*. Dengan membandingkan nilai pembacaan kedua titik ini, akan diketahui kemampuan kamera untuk melakukan perhitungan jarak antara 2 titik



Gambar 4.6 Pengujian 2 Titik

Hasil pengukuran disajikan dalam table berikut

Tabel 4.2 Hasil Pengujian Jarak 2 Titik

jarak	x	lebar	Error (%)
90	0	42.6	0.93
90	30	42.7	0.70
90	60		
90	90		
90	-30	43.5	1.16
90	-60		
90	-90		
180	0	41.7	3.02
180	30	41.8	2.79
180	60	42.7	0.70
180	90	41.7	3.02
180	-30	43.5	1.16
180	-60	45.5	5.81
180	-90	45.5	5.81
270	0	41.5	3.49
270	30	41.9	2.56
270	60	44.1	2.56
270	90	42.2	1.86
270	-30	43.6	1.40
270	-60	44.1	2.56
270	-90	43.2	0.47
360	0	40.9	4.88
360	30	41.4	3.72
360	60	42.4	1.40
360	90	40.9	4.88
360	-30	43.4	0.93
360	-60	43.4	0.93
360	-90	43.7	1.63
450	0	42.3	1.63
450	30	42.1	2.09
450	60	39.3	8.60
450	90	39	9.30
450	-30	41.8	2.79
450	-60	44.6	3.72
450	-90	45.2	5.12
540	0	44.3	3.02
540	30	44.6	3.72
540	60	40.8	5.12
540	90	41.8	2.79
540	-30	40.5	5.81
540	-60	41.8	2.79
540	-90	44.9	4.42

Berbeda dengan pengujian sebelumnya yang disajikan dalam 7 kolom, pengujian ini diajikan hanya dengan 4 kolom. Perbedaan dikarenakan jumlah variabel yang diamati lebih sedikit dibanding sebelumnya. Variabel jarak dan x adalah koordinat peletakan objek. Pada pengamatan ini tidak ditambahkan pembacaan *laser rangefinder*, karena selisih posisi yang cukup kecil tidak memberikan pengaruh terhadap pembacaan. Variabel lebar adalah lebar objek, atau nilai jarak dari 2 titik yang diukur. Dari hasil pengujian didapat bahwa rata-rata presentase error adalah 3.14%.

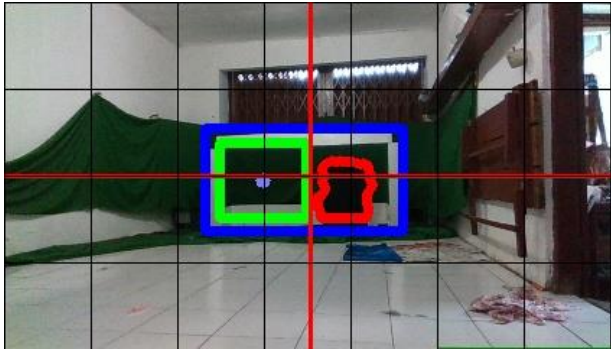
4.3 Uji Deteksi Gawang

Pengujian ini dilakukan untuk mencari tingkat keberhasilan dari sistem untuk mengenali gawang. Sedikit berbeda dengan pengujian-pengujian sebelumnya, dimana data yang ditampilkan berupa tabel dan data numerik, pengujian ini dilakukan dengan mengambil sample gambar dari beberapa titik di ruang uji untuk kemudian diperiksa apakah gambar yang diambil dapat mendeteksi gawang dengan baik.

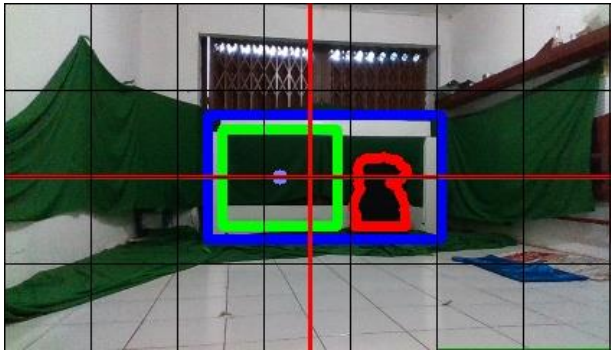
Gambar yang diambil harus memenuhi beberapa syarat untuk dapat dikategorikan sebagai gambar yang berhasil, antara lain:

1. Semua bagian gawang harus masuk dalam kotak biru
2. Objek penjaga gawang telah diberi tanda berupa garis berwarna merah
3. Area kosong terbesar diberi tanda berupa kotak hijau

Berdasarkan pengamatan, seluruh gambar yang diambil telah memenuhi syarat tersebut. Dikarenakan banyaknya jumlah gambar, tidak semua gambar akan dicantumkan pada sub bab ini. Beberapa contoh gambar yang diambil adalah sebagai berikut



Gambar 4.7 Contoh Uji Deteksi Gawang 1



Gambar 4.8 Contoh Uji Deteksi Gawang 2



Gambar 4.9 Contoh Uji Deteksi Gawang 3

4.4 Pengujian Jumlah Goal

Hasil akhir dari sistem ini ditentukan berdasarkan kemampuan sistem untuk mengenali koordinat area kosong gawang dan mengubah arah hadap robot menuju titik tersebut. Untuk dapat mengetahui keberhasilan dari kemampuan tersebut, robot akan diposisikan di beberapa titik sample pengujian, kemudian menjalankan algoritma program layaknya pada sub bab 3.1. Seperti halnya penjelasan pada sub bab 3.3 dan 3.23, penelitian ini dilakukan dengan sistem pengganti, sehingga tidak dapat melakukan prosedur tendangan. Oleh karena itu pengujian hanya dilakukan dengan membandingkan koordinat area kosong yang terbaca dengan koordinat area kosong sesungguhnya.

Pembacaan koordinat pada pengujian ini akan menggunakan koordinat robot, dimana sumbu x berada pada bagian lebar lapangan sedangkan sumbu y berada pada bagian Panjang lapangan. Nilai y pada koordinat ini akan selalu bernilai konstan sebesar 1200. Sehingga koordinat yang terbaca adalah (nilai x, 1200).

Pengujian dilakukan dengan berbagai titik posisi robot, dan posisi objek penjaga gawang yang diubah-ubah. Pada tabel selanjutnya hanya akan ditampilkan sebagian data pengujian, yaitu data untuk posisi objek penjaga gawang berjarak 30 cm dan 60 cm dari kanan gawang, sedangkan data keseluruhan akan ditampilkan dalam lampiran

Tabel 4.3 Pengujian Goal dan Koordinat deteksi

pos_x	pos_y	Nilai x	Kosong min	Kosong max	kiri min	kiri max	Kanan min	kanan max	deteksi	Error Nilai x	Error Kosong min (%)	Error Kosong max (%)
400	600	362.185	300	424.37	300	430	470	500	1	0.77	0.00	1.31
490	630	355.856	300	411.111	300	430	470	500	1	2.51	0.00	4.39
310	630	365.625	300	429.688	300	430	470	500	1	0.17	0.00	0.07
400	690	360.432	300	425.18	300	430	470	500	1	1.25	0.00	1.12
490	690	358.394	298.54	419.708	300	430	470	500	1	1.81	0.49	2.39
310	690	367.153	300	429.927	300	430	470	500	1	0.59	0.00	0.02
400	780	360.355	300	422.353	300	430	470	500	1	1.27	0.00	1.78
490	780	358.537	300	417.073	300	430	470	500	1	1.77	0.00	3.01
310	780	363.804	298.824	432.941	300	430	470	500	1	0.33	0.39	0.68
400	870	360.633	299.095	421.267	300	430	470	500	1	1.20	0.30	2.03
490	870	355.399	299.057	411.321	300	430	470	500	1	2.63	0.31	4.34
310	870	368.545	300.935	437.383	300	430	470	500	1	0.97	0.31	1.72
400	960	361.842	298.689	423.934	300	430	470	500	1	0.87	0.44	1.41
490	960	355.147	298.54	408.759	300	430	470	500	1	2.70	0.49	4.94
310	960	369.853	300	439.706	300	430	470	500	1	1.33	0.00	2.26
400	600	343.333	298.333	390	300	400	440	500	1	1.90	0.56	2.50
490	630	339.669	303.252	386.179	300	400	440	500	1	2.95	1.08	3.46
310	630	350.406	298.361	400	300	400	440	500	1	0.12	0.55	0.00
400	690	345.714	300	389.362	300	400	440	500	1	1.22	0.00	2.66
490	690	343.796	300	391.971	300	400	440	500	1	1.77	0.00	2.01
310	690	352.555	300	400.73	300	400	440	500	1	0.73	0.00	0.18
400	780	348.521	300	397.647	300	400	440	500	1	0.42	0.00	0.59
490	780	345.122	298.773	389.571	300	400	440	500	1	1.39	0.41	2.61
310	780	351.534	300	403.659	300	400	440	500	1	0.44	0.00	0.91
400	870	347.534	300	398.214	300	400	440	500	1	0.70	0.00	0.45
490	870	342.654	300	385.849	300	400	440	500	1	2.10	0.00	3.54
310	870	352.941	300	406.863	300	400	440	500	1	0.84	0.00	1.72
400	960	348.562	298.71	398.065	300	400	440	500	1	0.41	0.43	0.48
310	960	356.522	300	414.079	300	400	440	500	1	1.86	0.00	3.52
400	600	440.678	384.746	500	300	330	370	500	1	1.31	3.99	0.00

Semua bagian data ini mengacu pada sistem koordinat robot yang telah dijelaskan pada sub bab 2.8. Makna dari masing-masing variabel yang disajikan adalah sebagai berikut

- Pos_x dan Pos_y : koordinat sistem saat pengambilan data
- Nilai x : Nilai x pada koordinat tendang. Dengan nilai y selalu menunjukkan nilai yang konstan sebesar 1200. Seperti yang telah dijelaskan sebelumnya.
- kosong max dan kosong min : batas area yang masih dianggap kosong oleh sistem
- kiri max dan kiri min : batas area kosong pada kiri objek penjaga gawang yang sesungguhnya
- kanan max dan kanan min : batas area kosong pada kanan objek penjaga gawang yang sesungguhnya
- deteksi : berhasil atau tidaknya sistem untuk mendeteksi koordinat yang sesuai. Bernilai 1 (koordinat terdeteksi dengan benar) jika variabel Nilai x ada di dalam batas area kosong, dan 0 (tidak dapat mendeteksi dengan benar) jika sebaliknya.

Pada data ini didapatkan 1.84% rata-rata error untuk batas paling kiri area yang terdeteksi, 1.03% untuk batas kanan, dan 1.3% untuk nilai koordinat. Sedangkan data akurasi deteksi mencapai 100%. Nilai akurasi ini mencapai 100% karena area kosong gawang yang cukup luas, sehingga meskipun terjadi sedikit perbedaan antara koordinat yang terdeteksi dengan koordinat sesungguhnya, jika koordinat yang didapat masih berada pada batasan area kosong sesungguhnya maka sistem tetap dianggap berhasil menentukan koordinat

Halaman ini sengaja dikosongkan

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan serangkaian pengujian yang telah dilakukan, dapat diambil beberapa poin kesimpulan sebagai berikut:

1. Pengujian jarak memiliki hasil yang sesuai dengan spesifikasi intel realsense D435i yang menunjukkan bahwa semakin jauh jarak objek, akan semakin besar RMS error dari nilai yang didapat. Rata-rata presentase error pengujian koordinat pada subab 4.1 adalah 5.7 % untuk Koordinat x dan 1.49% untuk Koordinat Z.
2. Metode yang digunakan untuk melakukan perhitungan koordinat area kosong gawang dengan membandingkan jarak antara 2 titik dan memproyeksikan jarak tersebut pada koordinat lapangan, memiliki presentase error sebesar 3.14%.
3. Pengujian keseluruhan sistem menghasilkan akurasi deteksi area kosong mencapai 100%. Sistem yang dikembangkan relatif telah berhasil mencapai tujuannya.

5.2 Saran

Penelitian ini memiliki potensi untuk lebih dikembangkan. Beberapa saran yang dapat diberikan untuk pengembangan lebih lanjut dari penelitian ini adalah:

1. Menggunakan sistem pada robot IRIS secara menyeluruh untuk pengujian total sistem, terutama pada bagian pergerakan robot.
2. Menggunakan *depth camera* yang memiliki tingkat akurasi lebih tinggi untuk mengurangi kesalahan pembacaan.

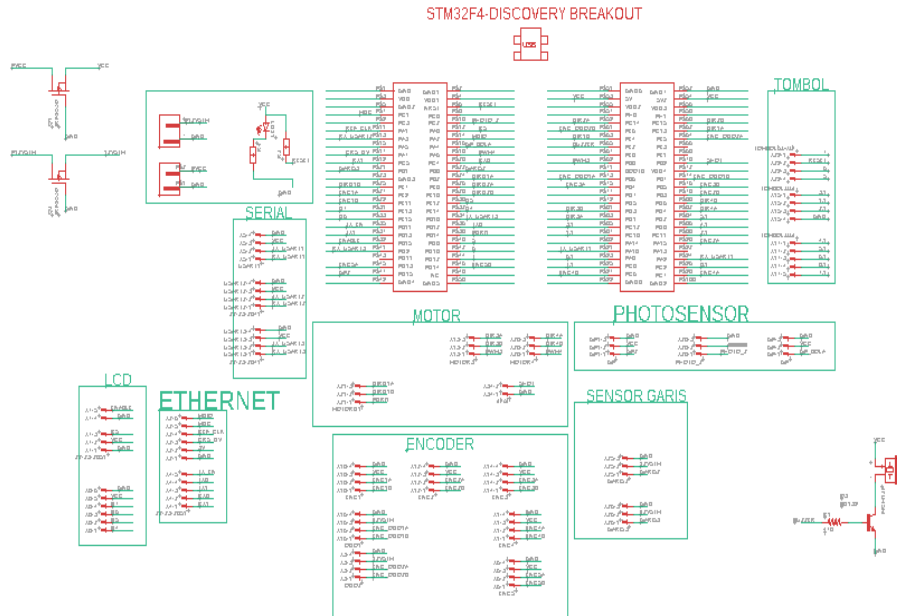
DAFTAR PUSTAKA

- [1] A. López Peláez, “From the Digital Divide to the Robotics Divide? Reflections on Technology, Power, and Social Change,” dalam *The Robotics Divide*, A. López Peláez, Ed. London: Springer London, 2014, hlm. 5–24.
- [2] Balkeshwar Singh, N. Sellappan, dan Kumaradhas P, “Evolution of Industrial Robots and their Applications,” *Int. J. Emerg. Technol. Adv. Eng.*
- [3] Teng Wang, Leping Bu, dan Zhongyi Huang, “A new method for obstacle detection based on Kinect depth image,” dalam *2015 Chinese Automation Congress (CAC)*, Wuhan, China, Nov 2015, hlm. 537–541, doi: 10.1109/CAC.2015.7382559.
- [4] Rostam Affendi Hamzah, Hasrul Nisham Rosly, dan S. Hamid, “An obstacle detection and avoidance of a mobile robot with stereo vision camera,” dalam *2011 International Conference on Electronic Devices, Systems and Applications (ICEDSA)*, Kuala Lumpur, Malaysia, Apr 2011, hlm. 104–108, doi: 10.1109/ICEDSA.2011.5959032.
- [5] S. B. Mane dan S. Vhanale, “Real time obstacle detection for mobile robot navigation using stereo vision,” dalam *2016 International Conference on Computing, Analytics and Security Trends (CAST)*, Pune, India, Des 2016, hlm. 637–642, doi: 10.1109/CAST.2016.7915045.
- [6] M. S. Ahn, H. Chae, D. Noh, H. Nam, dan D. Hong, “Analysis and Noise Modeling of the Intel RealSense D435 for Mobile Robots,” dalam *2019 16th International Conference on Ubiquitous Robots (UR)*, Jeju, Korea (South), Jun 2019, hlm. 707–711, doi: 10.1109/URAI.2019.8768489.
- [7] “PETUNJUK PELAKSANAAN KONTES ROBOT INDONESIA (KRI) TAHUN 2020,” hlm. 136.
- [8] A. R. Tinkar *dkk.*, “Team Description Paper: IRIS Team 2020,” hlm. 8.
- [9] O. Wasenmüller dan D. Stricker, “Comparison of Kinect V1 and V2 Depth Images in Terms of Accuracy and Precision,” dalam *Computer Vision – ACCV 2016 Workshops*, vol. 10117, C.-S. Chen, J. Lu, dan K.-K. Ma, Ed. Cham: Springer International Publishing, 2017, hlm. 34–45.

- [10] “How-to: Getting IMU data from D435i and T265,” *Intel® RealSense™ Depth and Tracking Cameras*, Feb 04, 2019. <https://www.intelrealsense.com/how-to-getting-imu-data-from-d435i-and-t265/> (diakses Jun 08, 2020).
- [11] P. S. Tantra, “PENENTUAN POSISI ROBOT SEPAK BOLA BERODA BERDASARKAN PENGINDRAAN VISUAL,” hlm. 88, 2018.
- [12] “OpenCV.” <https://opencv.org/> (diakses Jun 08, 2020).
- [13] A. Grunnet-Jepsen, J. N. Sweetser, dan J. Woodfill, “Best-Known-Methods for Tuning Intel® RealSense™ D400 Depth Cameras for Best Performance,” hlm. 10.
- [14] “Intel® RealSense™ D400Series/SR300 Viewer,” hlm. 32.

LAMPIRAN

1. Gambar Skema Rangkaian Elektronik



2. Data pengujian keseluruhan untuk sub bab 4.1

jarak	jarak laser	x real	x realsense	z realsense	Error x (%)	Error z (%)
100	101.9	0	-0.32	101.2		0.69
		20	20.7	100.6	3.50	1.28
		40	39.3	101.4	1.75	0.49
		60				
		80				
		100				
		-20	-20.3	102.3	1.50	0.39
		-40	-40.1	101.4	0.25	0.49
		-60				
		-80				
200	198.1	0	0.6	196.3		0.91
		20	21.9	198	9.50	0.05
		40	41.4	198.6	3.50	0.25
		60	59.7	198.6	0.50	0.25
		80	81.7	197.5	2.13	0.30
		100	101.9	201	1.90	1.46
		-20	-18.6	198	7.00	0.05
		-40	-38.7	198	3.25	0.05
		-60	-60.1	196.3	0.17	0.91
		-80	-80.7	201.5	0.88	1.72
300	301	0	1.9	300.5		0.17
		20	21.5	303.2	7.50	0.73
		40	41.3	301.9	3.25	0.30
		60	61.3	299.2	2.17	0.60
		80	81.8	299.2	2.25	0.60
		100	97.9	294.1	2.10	2.29
		-20	-18.50	299.20	7.50	0.60
		-40	-39.6	297.9	1.00	1.03
		-60	-59.8	301.9	0.33	0.30
		-80	-82.1	300.5	2.62	0.17
400	397.9	0	2.6	403.1		1.31
		20	21.5	403.1	7.50	1.31
		40	43.5	389.3	8.75	2.16
		60	63.4	403.1	5.67	1.31
		80	83.7	391.6	4.63	1.58
		100	103.1	410.3	3.10	3.12
		-20	-18.8	407.9	6.00	2.51
		-40	-38.7	410.3	3.25	3.12
		-60	-57.7	403.1	3.83	1.31
		-80	-81.6	405.5	2.00	1.91
500	497.2	0	1.6	503.9		1.35
		20	24.4	500.2	22.00	0.60
		40	42.3	503.9	5.75	1.35
		60	62.9	496.6	4.83	0.12
		80	83.7	511.4	4.63	2.86

		100	103.4	507.6	3.40	2.09
		-20	-16.6	507.6	17.00	2.09
		-40	-39.6	496.6	1.00	0.12
		-60	-60.3	503.9	0.50	1.35
		-80	-78.2	489.5	2.25	1.55
		-100	-112.5	523.1	12.50	5.21
		0	3.8	595.9		1.01
		20	22.5	580.7	12.50	3.54
		40	47.8	611.8	19.50	1.63
		60	68.3	611.8	13.83	1.63
		80	82.2	622.4	2.75	3.39
600	602	100	107.5	622.9	7.50	3.47
		-20	-13.2	601.1	34.00	0.15
		-40	-37.8	595.9	5.50	1.01
		-60	-63.4	611.8	5.67	1.63
		-80	-78.5	617.3	1.88	2.54
		-100	-98.9	622.9	1.10	3.47
		Rata-rata			5.7	1.49

3. Uji deteksi gawang memiliki 60 gambar yang berbeda untuk setiap titik. Gambar tersebut tidak dapat ditampilkan pada lampiran karena jumlahnya yang terlalu banyak. Untuk itu, gambar dapat diunduh pada link berikut https://itsacid-my.sharepoint.com/:f/g/personal/2216100016_mahasiswa_integra_its_ac_id/Epzm6d4Nww5MltNuumrTN40B60CZUHzFVIHdkIemvTtpJw?e=PJtcII

4. Data pengujian goal dan batas area kosong untuk sub bab 4.4

pos_x	pos_y	Nilai x	Kosong min	Kosong max	kiri min	kiri max	Kanan min	kanan max	goal	Error Nilai x	Error kosong min (%)	Error Kosong max (%)
400	600	362.185	300	424.37	300	430	470	500	1	0.77	0.00	1.31
490	630	355.856	300	411.111	300	430	470	500	1	2.51	0.00	4.39
310	630	365.625	300	429.688	300	430	470	500	1	0.17	0.00	0.07
400	690	360.432	300	425.18	300	430	470	500	1	1.25	0.00	1.12
490	690	358.394	298.54	419.708	300	430	470	500	1	1.81	0.49	2.39
310	690	367.153	300	429.927	300	430	470	500	1	0.59	0.00	0.02
400	780	360.355	300	422.353	300	430	470	500	1	1.27	0.00	1.78
490	780	358.537	300	417.073	300	430	470	500	1	1.77	0.00	3.01
310	780	363.804	298.824	432.941	300	430	470	500	1	0.33	0.39	0.68
400	870	360.633	299.095	421.267	300	430	470	500	1	1.20	0.30	2.03
490	870	355.399	299.057	411.321	300	430	470	500	1	2.63	0.31	4.34
310	870	368.545	300.935	437.383	300	430	470	500	1	0.97	0.31	1.72
400	960	361.842	298.689	423.934	300	430	470	500	1	0.87	0.44	1.41
490	960	355.147	298.54	408.759	300	430	470	500	1	2.70	0.49	4.94
310	960	369.853	300	439.706	300	430	470	500	1	1.33	0.00	2.26
400	600	343.333	298.333	390	300	400	440	500	1	1.90	0.56	2.50
490	630	339.669	303.252	386.179	300	400	440	500	1	2.95	1.08	3.46
310	630	350.406	298.361	400	300	400	440	500	1	0.12	0.55	0.00
400	690	345.714	300	389.362	300	400	440	500	1	1.22	0.00	2.66
490	690	343.796	300	391.971	300	400	440	500	1	1.77	0.00	2.01
310	690	352.555	300	400.73	300	400	440	500	1	0.73	0.00	0.18
400	780	348.521	300	397.647	300	400	440	500	1	0.42	0.00	0.59
490	780	345.122	298.773	389.571	300	400	440	500	1	1.39	0.41	2.61
310	780	351.534	300	403.659	300	400	440	500	1	0.44	0.00	0.91
400	870	347.534	300	398.214	300	400	440	500	1	0.70	0.00	0.45
490	870	342.654	300	385.849	300	400	440	500	1	2.10	0.00	3.54
310	870	352.941	300	406.863	300	400	440	500	1	0.84	0.00	1.72
400	960	348.562	298.71	398.065	300	400	440	500	1	0.41	0.43	0.48
310	960	356.522	300	414.079	300	400	440	500	1	1.86	0.00	3.52

pos_x	pos_y	Nilai x	Kosong min	Kosong max	kiri min	kiri max	Kanan min	kanan max	goal	Error Nilai x	Error Kosong min (%)	Error Kosong max (%)
400	600	440.678	384.746	500	300	330	370	500	1	1.31	3.99	0.00
490	630	439.2	381.6	500	300	330	370	500	1	0.97	3.14	0.00
310	630	446.341	390.164	501.639	300	330	370	500	1	2.61	5.45	0.33
400	690	441.429	382.857	504.286	300	330	370	500	1	1.48	3.47	0.86
490	690	438.13	380.576	500	300	330	370	500	1	0.72	2.86	0.00
310	690	447.445	394.891	500	300	330	370	500	1	2.86	6.73	0.00
400	780	440	382.353	500	300	330	370	500	1	1.15	3.34	0.00
490	780	437.349	376.647	500	300	330	370	500	1	0.54	1.80	0.00
310	780	442.683	392.941	500	300	330	370	500	1	1.77	6.20	0.00
400	870	440.639	383.258	500	300	330	370	500	1	1.30	3.58	0.00
490	870	436.232	375.728	500	300	330	370	500	1	0.28	1.55	0.00
310	870	445.098	391.176	500	300	330	370	500	1	2.32	5.72	0.00
400	960	439.934	380.921	499.342	300	330	370	500	1	1.13	2.95	0.13
490	960	432.867	366.667	500	300	330	370	500	1	0.49	0.90	0.00
310	960	448.364	398.54	500	300	330	370	500	1	3.07	7.71	0.00
400	600	454.622	410.924	500	300	360	400	500	1	1.03	2.73	0.00
490	630	454.098	408.943	498.374	300	360	400	500	1	0.91	2.24	0.33
310	630	460.976	423.577	500	300	360	400	500	1	2.44	5.89	0.00
400	690	453.957	411.429	500	300	360	400	500	1	0.88	2.86	0.00
490	690	457.664	407.353	500	300	360	400	500	1	1.70	1.84	0.00
310	690	457.971	422.302	500	300	360	400	500	1	1.77	5.58	0.00
400	780	453.846	410.059	500	300	360	400	500	1	0.85	2.51	0.00
490	780	450.296	401.775	492.899	300	360	400	500	1	0.07	0.44	1.42
310	780	458.537	421.951	500	300	360	400	500	1	1.90	5.49	0.00
400	870	453.153	410.619	500	300	360	400	500	1	0.70	2.65	0.00
490	870	449.763	400	499.057	300	360	400	500	1	0.05	0.00	0.19
310	870	458.852	421.154	501.923	300	360	400	500	1	1.97	5.29	0.38
400	960	453.443	407.843	499.346	300	360	400	500	1	0.77	1.96	0.13
490	960	443.86	390.845	500	300	360	400	500	1	1.36	2.29	0.00
Rata-rata										1.3	1.84	1.03

Program untuk algoritma penentu koordinat

```
#include <librealsense2/rs.hpp>
#include <iostream>
#include <math.h>
#include <opencv2/opencv.hpp>
#include <librealsense2/rsutil.h>
#include <librealsense2/rs_advanced_mode.hpp>
#include <ros/ros.h>
#include "geometry_msgs/Pose2D.h"
#include "geometry_msgs/Twist.h"
#include "std_msgs/Float32MultiArray.h"
#include "std_msgs/Float32.h"
#include "std_msgs/Bool.h"
#include <strings.h>
#include <fstream>

#define OPTIMAL_WIDTH 848
#define OPTIMAL_HEIGHT 480
#define WIDTH 424
#define HEIGHT 240

#define FIELD_X 800
#define FIELD_Y 1200

#define GOAL_X FIELD_X / 2
#define GOAL_Y FIELD_Y
#define OFFSET_X 100

using namespace cv;
using namespace std;
using namespace ros;

ros::Timer tim_color;
ros::Timer tim_clear_area;

ros::Subscriber sub_odometry;
ros::Subscriber sub_capture;
ros::Subscriber sub_temp_goal;

ros::Publisher pub_coordinate;
ros::Publisher pub_goal_area;

//field coordinate
int x = WIDTH/2 ,y = HEIGHT/2;
int x_max = 0, x_min = 0;
int free_max = 0, free_min = 0; //ambil data
float coordinate[3];
```

```

float goal_min[3];
float goal_max[3];
float center_view[3]; //ambil data
float free_area_min[3]; //ambil data
float free_area_max[3]; //ambil data

float pos_x_buffer = 400, pos_y_buffer = 1100,
theta_buffer;

//thresholding
int h_low = 0, s_low = 0, v_low = 70;
int h_high = 180, s_high = 74, v_high = 255;

int h_low_gray = 0, s_low_gray = 0, v_low_gray = 0;
int h_high_gray = 180, s_high_gray = 120, v_high_gray =
30;

int gray_trim = 16;

//helper
int far_offset = 550, near_offset = 562;
// int far_offset = 540, near_offset = 560;
int keeper_far_offset = far_offset, keeper_near_offset =
near_offset; //dibuat agar offset bisa bernilai negatif
int erode_size = 10;
int scan_up = 0, scan_down = 0;

bool status_capture = false;
float temp_goal_x = 0;

Rect goal_ROI;
Rect free_ROI;

//single channel opencv mat
Mat goal_depth_threshold = Mat::zeros(Size(WIDTH, HEIGHT),
CV_8UC1);
Mat keeper_depth_threshold = Mat::zeros(Size(WIDTH,
HEIGHT), CV_8UC1);
Mat goal_color = Mat::zeros(Size(WIDTH, HEIGHT), CV_8UC1);
Mat obstacle_color = Mat::zeros(Size(WIDTH, HEIGHT),
CV_8UC1);
Mat final_threshold = Mat::zeros(Size(WIDTH, HEIGHT),
CV_8UC1);
Mat goal_threshold = Mat::zeros(Size(WIDTH, HEIGHT),
CV_8UC1);
Mat goal_threshold_display = Mat::zeros(Size(WIDTH,
HEIGHT), CV_8UC1);
Mat goal_obstacle = Mat::zeros(Size(WIDTH, HEIGHT),
CV_8UC1);

```

```

Mat goal_clear = Mat::zeros(Size(WIDTH, HEIGHT), CV_8UC1);

//multiple channel opencv mat
Mat display_color = Mat::zeros(Size(WIDTH, HEIGHT),
CV_8UC3);
Mat display_final = Mat::zeros(Size(WIDTH, HEIGHT),
CV_8UC3);
Mat color_raw = Mat::zeros(Size(WIDTH, HEIGHT), CV_8UC3);
Mat color_hsv = Mat::zeros(Size(WIDTH, HEIGHT), CV_8UC3);
// Mat depth_threshold = Mat::zeros(Size(WIDTH, HEIGHT),
CV_8UC3);

//substitute mat
Mat imshow1;
Mat imshow2;
int show1 = 0, show2 = 0;

void cllbck_sub_odometry(const
geometry_msgs::Pose2DConstPtr &msg)
{
    pos_x_buffer = msg->x;
    pos_y_buffer = msg->y;
    theta_buffer = msg->theta;
}

void cllbck_sub_capture(const std_msgs::BoolConstPtr &msg)
{
    status_capture = msg->data;

    // status_capture = false;
}

void cllbck_temp_goal(const geometry_msgs::TwistConstPtr
&msg)
{
    temp_goal_x = msg->linear.x;
}

void cllbck_tim_color(const ros::TimerEvent &event)
{
    // algoritma pengambilan gambar gawang dan kiper:
    //mengambil gambar RGB dan melakukan threshold putih
    //bitwise_and pada depth gawang dan threshold putih
    //memberi kotak pada threshold
    //mengurangkan dengan keeper (objek berwarna bukan
    putih dan jarak = gawang)
}

```

```

    if(!color_raw.empty() &&
!goal_depth_threshold.empty())
    {
        display_color = color_raw.clone();

        Mat gray;
        cvtColor(color_raw, color_hsv, COLOR_RGB2HSV);
        cvtColor(color_raw, gray, COLOR_RGB2GRAY);
        inRange(color_hsv, Scalar(h_low, s_low, v_low),
Scalar(h_high, s_high, v_high), goal_color);
        inRange(color_hsv, Scalar(h_low_gray, s_low_gray,
v_low_gray), Scalar(h_high_gray, s_high_gray,
v_high_gray), obstacle_color);
        erode(obstacle_color, obstacle_color,
getStructuringElement(MORPH_ELLIPSE, Size(3, 3)));
        dilate(obstacle_color, obstacle_color,
getStructuringElement(MORPH_ELLIPSE, Size(7, 7)));
        // bitwise_not(obstacle_color, obstacle_color);
        // threshold(gray, obstacle_color, gray_trim, 255,
THRESH_BINARY_INV);

        bitwise_and(goal_color, goal_depth_threshold,
goal_threshold);
        bitwise_and(obstacle_color,
keeper_depth_threshold, goal_obstacle);
        dilate(goal_threshold, goal_threshold,
getStructuringElement(MORPH_ELLIPSE, Size(3, 3)));

        goal_threshold_display = goal_threshold.clone();

        vector<vector<Point>> contours;
        Point2f center;
        float radius;

        float min_area = 0;
        int largest_index = 0;

        //////////////////////////////////////
        //////////////////////////////////////
        findContours(goal_threshold, contours,
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);

        vector<vector<Point>>
contours_area(contours.size());
        vector<vector<Point>> hull_area(contours.size());

        for(int i = 0; i < contours.size(); i++)

```

```

        {
            if(contourArea(contours[i]) > min_area)
            {
                min_area = contourArea(contours[i]);
                largest_index = i;

                contours_area[0] = contours[i];
            }
        }

    final_threshold = Scalar(0);
    if(contours.size() && contours_area[0].size())
    {
        goal_ROI =
        boundingRect(Mat(contours_area[0]));
        x_max = goal_ROI.x + goal_ROI.width;
        x_min = goal_ROI.x;

        rectangle(final_threshold, goal_ROI,
        Scalar(255), -1);
        rectangle(display_color, goal_ROI,
        Scalar(255,0,0), 5);
    }

    //#####
    #####

    //////////////////////////////////////
    //////////////////////////////////////
    //////////////////////////////////////
    //////////////////////////////////////
    // erode(final_threshold, final_threshold,
    getStructuringElement(MORPH_ELLIPSE, Size(erode_size,
    erode_size)));
    bitwise_and(final_threshold, goal_obstacle,
    goal_obstacle);

    goal_clear = goal_obstacle.clone();
    vector<vector<Point>> contours_obstacle;
    float min_goal_area = 0;
    int largest_goal_index = 0;

    findContours(goal_obstacle, contours_obstacle,
    CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);
    for(int i = 0; i < contours_obstacle.size(); i++)
    {
        if(contourArea(contours_obstacle[i]) >
        min_goal_area)
        {

```

```

        min_goal_area =
contourArea(contours_obstacle[i]);
        largest_goal_index = i;
    }
}
Rect obstacle_ROI;

// goal_clear = Scalar(0);
if(contours_obstacle.size())
{
    drawContours(display_color, contours_obstacle,
largest_goal_index, Scalar(0,0,255), 5);

drawContours(Mat(contours_obstacle[largest_goal_index]),
contours_obstacle, 0, Scalar(255), -1);
    //
minEnclosingCircle(contours_obstacle[largest_goal_index],
center, radius);
    obstacle_ROI =
boundingRect(contours_obstacle[largest_goal_index]);
    rectangle(goal_obstacle, Point(obstacle_ROI.x,
0), Point(obstacle_ROI.x + obstacle_ROI.width,
final_threshold.rows), Scalar(255), -1);

    subtract(final_threshold, goal_obstacle,
goal_clear);
}

// if(goal_clear.empty()) goal_clear =
final_threshold.clone();
}

//#####
#####
}

void cllbck_tim_clear_area(const ros::TimerEvent &event)
{
    if(!goal_clear.empty() && !display_color.empty())
    {
        display_final = display_color.clone();
        vector<vector<Point>> contours_clear;
        float minimum_detected_area = 0;
        int largest_clear_index = 0;

        findContours(goal_clear, contours_clear,
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);
        for(int i = 0; i < contours_clear.size(); i++)
        {

```



```

        if(contourArea(contours_clear[i]) >
minimum_detected_area)
        {
            minimum_detected_area =
contourArea(contours_clear[i]);
            largest_clear_index = i;
        }
    }

    Point2f center_clear;
    float radius_clear;

    if(contours_clear.size())
    {
        // minimum_detected_area = 0;
        // ROS_INFO("%f", minimum_detected_area);
        drawContours(display_final, contours_clear,
largest_clear_index, Scalar(0,255,0), 5);

minEnclosingCircle(contours_clear[largest_clear_index],
center_clear, radius_clear);

        free_ROI =
boundingRect(Mat(contours_clear[largest_clear_index]));
        free_max = free_ROI.x + free_ROI.width;
        free_min = free_ROI.x;

        x = center_clear.x;
        y = center_clear.y;
        circle(display_final, center_clear, 5,
Scalar(255,125,125), -1);
    }

    else
    {
        findContours(final_threshold, contours_clear,
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);
        //jika tidak ditemukan halangan, semua area
gawang yang terdeteksi dianggap kosong

        if(contours_clear.size())
        {
            drawContours(display_final,
contours_clear, 0, Scalar(0,255,0), 5);
            minEnclosingCircle(contours_clear[0],
center_clear, radius_clear);

            free_ROI =
boundingRect(Mat(contours_clear[0]));
            free_max = free_ROI.x + free_ROI.width;

```

```

        free_min = free_ROI.x;

        x = center_clear.x;
        y = center_clear.y;
        circle(display_final, center_clear, 5,
Scalar(255,125,125), -1);
    }
}
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "bg_subs");
    ros::NodeHandle NH;
    // ros::MultiThreadedSpinner MTS;

    //timer
    tim_color = NH.createTimer(ros::Duration(0.02),
cllbck_tim_color);
    tim_clear_area = NH.createTimer(ros::Duration(0.02),
cllbck_tim_clear_area);

    //subscriber & publisher
    sub_odometry = NH.subscribe("robot_odometry", 16,
cllbck_sub_odometry);
    sub_capture = NH.subscribe("capture_status", 16,
cllbck_sub_capture);
    sub_temp_goal = NH.subscribe("temporary_goal", 16,
cllbck_temp_goal);
    pub_coordinate =
NH.advertise<std_msgs::Float32MultiArray>("shooting_coordi
nate", 16);
    pub_goal_area =
NH.advertise<std_msgs::Float32MultiArray>("prediction_coor
dinate", 16);

    //////////////////////////////////////realsense
config////////////////////////////////////
    rs2::pipeline pipe;
    rs2::colorizer color_map;
    rs2::align aligner(RS2_STREAM_COLOR);
    rs2::config cfg;

    cfg.enable_stream(RS2_STREAM_COLOR, WIDTH, HEIGHT,
RS2_FORMAT_BGR8, 60);
    // cfg.enable_stream(RS2_STREAM_DEPTH, WIDTH, HEIGHT,
RS2_FORMAT_Z16, 60);
    cfg.enable_stream(RS2_STREAM_DEPTH);
    auto prof = pipe.start(cfg);

```

```

rs2::device dev = prof.get_device();
auto advanced = dev.as<rs400::advanced_mode>();
ifstream
file("../ROS/program/sftp_dzul_TA_copy/high_density_auto.j
son");
// ifstream file("high_density_auto.json");
string str((istreambuf_iterator<char>(file)),
istreambuf_iterator<char>());
// cout << str << endl;
advanced.load_json(str);

////////////////////////////////////
////////////////////////////////////

int x_display = 0;
int y_display = 0;

int status_trackbar = 0;
bool first = true;

static int capture_counter = 0;
while(ok())
{
    spinOnce();

    rs2::frameset frames = pipe.wait_for_frames();
    rs2::frame color_display =
frames.get_color_frame();

    frames = aligner.process(frames);
    rs2::frame depth_display =
frames.get_depth_frame().apply_filter(color_map);
    rs2::depth_frame depth = frames.get_depth_frame();

    // auto depth_profile =
prof.get_stream(RS2_STREAM_DEPTH).as<rs2::video_stream_pro
file>();
    auto intrinsic =
depth.get_profile().as<rs2::video_stream_profile>().get_in
trinsics();

    const int w =
color_display.as<rs2::video_frame>().get_width();
    const int h =
color_display.as<rs2::video_frame>().get_height();

    // if(!first)
    // destroyWindow("distance");

```

```

        namedWindow("distance", WINDOW_AUTOSIZE);
        // createTrackbar("status trackbar", "distance",
&status_trackbar, 2, on_trackbar);

        // switch (status_trackbar)
        // {
        // case 0:
        createTrackbar("f_o", "distance", &far_offset,
1000);
        createTrackbar("n_o", "distance",
&near_offset, 1000);
        createTrackbar("k_f_o", "distance",
&keeper_far_offset, 1000);
        createTrackbar("k_n_o", "distance",
&keeper_near_offset, 1000);
        createTrackbar("gray", "distance", &gray_trim,
255);
        // createTrackbar("h+", "distance", &h_high,
255);
        // createTrackbar("h-", "distance", &h_low,
255);
        // createTrackbar("s+", "distance", &s_high,
255);
        // createTrackbar("s-", "distance", &s_low,
255);
        // createTrackbar("v+", "distance", &v_high,
255);
        // createTrackbar("v-", "distance", &v_low,
255);
        // break;

        // case 1:
        // createTrackbar("h+", "distance", &h_high,
180);
        // createTrackbar("h-", "distance", &h_low,
255);
        // createTrackbar("s+", "distance", &s_high,
255);
        // createTrackbar("s-", "distance", &s_low,
255);
        // createTrackbar("v+", "distance", &v_high,
255);
        // createTrackbar("v-", "distance", &v_low,
255);
        // // break;

        // // case 2:
        createTrackbar("h+ obstacle", "distance",
&h_high_gray, 180);

```

```

        createTrackbar("h- obstacle", "distance",
&h_low_gray, 255);
        createTrackbar("s+ obstacle", "distance",
&s_high_gray, 255);
        createTrackbar("s- obstacle", "distance",
&s_low_gray, 255);
        createTrackbar("v+ obstacle", "distance",
&v_high_gray, 255);
        createTrackbar("v- obstacle", "distance",
&v_low_gray, 255);
        //      break;

        // default:
        //      break;
        // }

        createTrackbar("show 1", "distance", &show1, 5);
        createTrackbar("show 2", "distance", &show2, 6);

        //untuk membatasi jika ada objek dekat berwarna
        putih diatas gawang
        if(FIELD_Y - pos_y_buffer > 550) scan_up = 55;
        else if(FIELD_Y - pos_y_buffer > 450) scan_up =
45;
        else if(FIELD_Y - pos_y_buffer > 350) scan_up =
35;
        else scan_up = 0;

        //melakukan perhitungan jarak dari robot ke gawang
        //dinagi menjadi 3 bagian tengah kiri kanan

        float var_dist = sqrt(pow(pos_x_buffer - GOAL_X,
2) + pow(pos_y_buffer - GOAL_Y, 2));
        float var_dist_l = sqrt(pow(pos_x_buffer - (GOAL_X
- OFFSET_X), 2) + pow(pos_y_buffer - GOAL_Y, 2));
        float var_dist_r = sqrt(pow(pos_x_buffer - (GOAL_X
+ OFFSET_X), 2) + pow(pos_y_buffer - GOAL_Y, 2));

        float max_pinggir = 0;
        float min_pinggir = 0;

        float far, near, keeper_far, keeper_near;

        if (pos_x_buffer < FIELD_X / 2 - 90 ||
pos_x_buffer > FIELD_X / 2 + 90)
        {
            max_pinggir = -10;
            min_pinggir = 28;
        }

```

```

        far = (var_dist + max_pinggir + far_offset - 500)
/ 100;
        near = (var_dist - (near_offset + min_pinggir -
500)) / 100;
        keeper_far = (var_dist + keeper_far_offset - 500)
/ 100;
        keeper_near = (var_dist - (keeper_near_offset -
500)) / 100;

        Mat color(Size(w, h), CV_8UC3,
(void*)color_display.get_data(), Mat::AUTO_STEP);
        color_raw = color.clone();

        ////////////////////////////////////////////////////////////////////untuk debugging depth.
mencari depth pada posisi x,y//////////////////////////////////////////////////////////////////
        createTrackbar("x", "distance", &x_display, w);
        createTrackbar("y", "distance", &y_display, h);
        Mat image(Size(w, h), CV_8UC3,
(void*)depth_display.get_data(), Mat::AUTO_STEP);
        circle(image, Point(x,y), 3, Scalar(0, 255, 0), -
1);

        float pixel_distance =
depth.get_distance(x_display, y_display);
        // float pixel_distance = var_dist;
        // cout << pixel_distance << endl;

//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////

        //untuk menghitung rata2 dari jarak gawang
        float avr_count = 0;
        float avr = 0;

        //untuk menghitung jarak terjauh/terdekat dari
gawang
        float furthest = 0;
        float nearest = 1000;

        goal_depth_threshold = Scalar(0);
        keeper_depth_threshold = Scalar(0);
        // depth_threshold = Scalar(0);

        for (int i = 0; i < w; i++)
        {
                for (int j = 0 + scan_up; j < h - scan_down;
j++)
                {
                        float distance = depth.get_distance(i, j);

                        if(distance > near && distance < far)

```

```

        {
            if(!goal_ROI.empty())
            {
                if(i >= goal_ROI.x && i <=
goal_ROI.x + goal_ROI.width
                && j >= goal_ROI.y && j <=
goal_ROI.y + goal_ROI.height)
                {
                    avr += distance*100;
                    avr_count++;

                    if (distance > furthest)
                    furthest = distance;
                    if (distance < nearest)
                    nearest = distance;
                }
            }
            circle(goal_depth_threshold,
Point(i,j), 1, Scalar(255));
            // circle(depth_threshold, Point(i,j),
1, Scalar(255,0,0));
        }

        if(distance > keeper_near && distance <
keeper_far)
        {
            circle(keeper_depth_threshold,
Point(i,j), 1, Scalar(255));
            // circle(depth_threshold, Point(i,j),
1, Scalar(0,0,255));
        }
    }

    avr = avr/(avr_count * 100);

    //menghitung koordinat tembak
    std_msgs::Float32MultiArray msg_koordinat;
    std_msgs::Float32MultiArray msg_predicted;
    // float pixel_distance = var_dist;
    float pixel[2] = {x, y};
    float pixel_center[2] = {w/2, h/2}; //ambil data
    float pixel_max[2] = {x_max, y};
    float pixel_min[2] = {x_min, y};

    float free_pixel_max[2] = {free_max, y};
    float free_pixel_min[2] = {free_min, y};

    rs2_deproject_pixel_to_point(coordinate,
&intrinsic, pixel, nearest);

```

```

        rs2_deproject_pixel_to_point(goal_max, &intrinsic,
pixel_max, nearest);
        rs2_deproject_pixel_to_point(goal_min, &intrinsic,
pixel_min, nearest);

        rs2_deproject_pixel_to_point(center_view,
&intrinsic, pixel_center, nearest);//ambil data
        rs2_deproject_pixel_to_point(free_area_max,
&intrinsic, free_pixel_max, nearest);//ambil data
        rs2_deproject_pixel_to_point(free_area_min,
&intrinsic, free_pixel_min, nearest);//ambil data

        msg_koordinat.data.push_back(coordinate[0]);
        msg_koordinat.data.push_back(coordinate[1]);
        msg_koordinat.data.push_back(coordinate[2]);

        pub_coordinate.publish(msg_koordinat);

        float predicted_x = (((coordinate[0] -
goal_min[0])/(goal_max[0] - goal_min[0])) * 200) + (GOAL_X
- 100);
        float view_x = (((center_view[0] -
goal_min[0])/(goal_max[0] - goal_min[0])) * 200) + (GOAL_X
- 100);//ambil data
        float predicted_y = nearest * 100 + pos_y_buffer ;
        float free_area_coord[2];
        free_area_coord[0] = (((free_area_min[0] -
goal_min[0])/(goal_max[0] - goal_min[0])) * 200) + (GOAL_X
- 100);
        free_area_coord[1] = (((free_area_max[0] -
goal_min[0])/(goal_max[0] - goal_min[0])) * 200) + (GOAL_X
- 100);
        msg_predicted.data.push_back(predicted_x);
        msg_predicted.data.push_back(predicted_y);
        pub_goal_area.publish(msg_predicted);

        // cout << predicted_x << "\t" << view_x << endl;
        // cout << far * 100 << "\t" << near * 100 << "\t"
<< keeper_far << "\t" << keeper_near << "\t" <<
pixel_distance * 100 << "\t"
        // << var_dist_l << "\t" << var_dist << "\t" <<
var_dist_r << endl;

        if(
!display_final.empty()
&& !image.empty()
&& !goal_color.empty()
&& !display_color.empty()
&& !goal_threshold_display.empty()

```



```

        && !goal_obstacle.empty()
        && !goal_clear.empty()
    )
    {

        //untuk mengetahui kemiringan kamera. diberi
        garis pada frame
        line(display_final, Point(0, HEIGHT/2),
        Point(WIDTH, HEIGHT/2), Scalar(0,0,255), 2);
        line(display_final, Point(WIDTH/2, 0),
        Point(WIDTH/2, HEIGHT), Scalar(0,0,255), 2);

        for(int i = 0; i < WIDTH; i += 60)
            line(display_final,Point(i, 0), Point(i,
            HEIGHT), Scalar(0));

        for(int i = 0; i < HEIGHT; i += 60)
            line(display_final,Point(0, i), Point(WIDTH,
            i), Scalar(0));

        // circle(image,Point(x_alternate,
        y_alternate), 2, Scalar(125,125,125),-1);

        namedWindow("show 1");
        namedWindow("show 2");

        switch (show1)
        {
        case 0: imshow1 = display_final.clone();
        break;
        case 1: imshow1 = image.clone(); break;
        case 2: imshow1 = goal_color.clone(); break;
        case 3: imshow1 = obstacle_color.clone();
        break;
        case 4: imshow1 =
        goal_depth_threshold.clone(); break;
        case 5: imshow1 =
        keeper_depth_threshold.clone(); break;

        default:
            break;
        }

        switch (show2)
        {
        case 0: imshow2 = display_final.clone();
        break;
        case 1: imshow2 = image.clone(); break;
        case 2: imshow2 = goal_color.clone(); break;

```

```

        case 3: imshow2 = obstacle_color.clone();
break;
        case 4: imshow2 =
goal_depth_threshold.clone(); break;
        case 5: imshow2 =
keeper_depth_threshold.clone(); break;
        case 6: imshow2 =
goal_threshold_display.clone(); break;

        default:
            break;
    }

    imshow("show 1", imshow1);
    imshow("show 2", imshow2);

    if(status_capture) //ambil data
    {
        stringstream ss;
        ss << capture_counter << "view" <<
pos_x_buffer << pos_y_buffer << ".jpg";
        String directory = "../Pictures/";
        String filename = ss.str();
        cout << directory + filename;

        stringstream note_stream;
        note_stream << capture_counter << "\t" <<
pos_x_buffer << "\t"
        << pos_y_buffer << "\t" <<
predicted_x << "\t"
        << view_x << "\t" <<
temp_goal_x << "\t"
        << free_area_coord[0] << "\t"
<< free_area_coord[1] << endl;

        stringstream tabel_stream;
        tabel_stream << "no" << "\t" << "pos_x" <<
"\t"
        << "pos_y" << "\t" <<
"predicted_x" << "\t"
        << "view_x" << "\t" <<
"temp_goal_x" << "\t"
        << "free_area_min" << "\t" <<
"free_area_max" << endl;

        if(first)
        {
            ofstream
out_file("../Pictures/log.csv");
            out_file << tabel_stream.str();

```

```

        out_file << note_stream.str();
        out_file.close();
        first = false;
    }
    else
    {
        ofstream
in_file("../Pictures/log.csv", std::ios_base::app);
        in_file << note_stream.str();
        in_file.close();
    }

    Mat capture_display =
display_final.clone();

    bool check_image = imwrite(directory +
filename, capture_display);

    capture_counter++;
    status_capture = false;

    // cout << check_image<< endl;
}
}

    waitKey(1);
}
}

```

Program untuk *IMU* dan Pengganti *Rotary Encoder*

```

#include <librealsense2/rs.hpp>
#include <iostream>
#include <math.h>
#include <opencv2/opencv.hpp>
#include <librealsense2/rsutil.h>
#include <librealsense2/rs_advanced_mode.hpp>
#include <ros/ros.h>
#include "geometry_msgs/Pose2D.h"
#include "std_msgs/Float32MultiArray.h"
#include <fstream>

#define PI 3.14159265359

using namespace cv;
using namespace std;
using namespace ros;

```

```

rs2_vector gyro_sample;
rs2_vector accel_sample;

ros::Timer tim_dummy_pos;

ros::Publisher pub_gyro;
ros::Publisher pub_pos;

int pos_x = 400, pos_y = 0;

float gyro_yaw = 0;
float gyro_pitch = 0;
float gyro_roll = 0;
float accel_yaw = 0;
float accel_pitch = 0;
float accel_roll = 0;

int config_exposure = 1;
int config_auto_exposure = 1;

void cllbck_tim_dummy_pos(const ros::TimerEvent &event)
{
    geometry_msgs::Pose2D msg_pos;

    msg_pos.x = pos_x;
    msg_pos.y = pos_y + 600;
    msg_pos.theta = gyro_yaw * -180/PI;
    // msg_pos.theta = 0;

    pub_pos.publish(msg_pos);
}

static void on_config(int, void*)
{
    destroyWindow("distance");
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "realsense_gyro");
    ros::NodeHandle NH;

    ros::Rate loop_rate(200);
    tim_dummy_pos = NH.createTimer(ros::Duration(0.02),
    cllbck_tim_dummy_pos);

    pub_pos =
    NH.advertise<geometry_msgs::Pose2D>("stm2pc/odometry_buffer", 16);
}

```

```

////////////////////////////////////realsense
config////////////////////////////////////
    rs2::pipeline pipe;
    rs2::config cfg;

    cfg.enable_stream(RS2_STREAM_ACCEL,
RS2_FORMAT_MOTION_XYZ32F);
    cfg.enable_stream(RS2_STREAM_GYRO,
RS2_FORMAT_MOTION_XYZ32F);

    rs2::pipeline_profile prof = pipe.start(cfg);

////////////////////////////////////
////////////////////////////////////

    rs2::device device = prof.get_device();
    auto depth_sensor = device.first<rs2::depth_sensor>();

    static double timer_start = ros::Time::now().toSec();
    float resultant = 0;

    static int avr_counter = 0;
    static float avr_gyro = 0;

    namedWindow("position", WINDOW_AUTOSIZE);

    while(ok())
    {
        rs2::frameset frames = pipe.wait_for_frames();
        rs2::motion_frame motion =
frames.as<rs2::motion_frame>();
        // rs2::depth_frame depth =
frames.as<rs2::depth_frame>();

        if (rs2::motion_frame accel_frame =
frames.first_or_default(RS2_STREAM_ACCEL))
        {
            accel_sample = accel_frame.get_motion_data();
            resultant = sqrtf(pow(accel_sample.x, 2) +
pow(accel_sample.y, 2) + pow(accel_sample.z, 2));

            accel_pitch = atan2f(accel_sample.x,
resultant);
            accel_roll = atan2f(accel_sample.y,
accel_sample.z);
        }

        if (rs2::motion_frame gyro_frame =
frames.first_or_default(RS2_STREAM_GYRO))
        {

```

```

        gyro_sample = gyro_frame.get_motion_data();
        double timer_get_data =
ros::Time::now().toSec();
        double dt = timer_get_data - timer_start;
        timer_start = timer_get_data;

        if(gyro_sample.y < 0.01 && gyro_sample.y > -
0.01) gyro_sample.y = 0;

        avr_gyro += gyro_sample.y;

        gyro_yaw -= gyro_sample.y * dt;
        gyro_pitch += gyro_sample.y * dt;
        gyro_roll += gyro_sample.y * dt;

        avr_counter++;

        // ROS_INFO("Gyro : %.7f   %.7f",
gyro_sample.y, dt);
        // ROS_INFO("dt %.7f", dt);
    }

    gyro_pitch = (gyro_pitch * 0.98) + accel_pitch *
0.02;
    gyro_roll = (gyro_roll * 0.98) + accel_roll *
0.02;
    // gyro_yaw = gyro_yaw *0.98;

    // ROS_INFO("yaw = %.7f   %.7f", gyro_yaw, gyro_yaw
* 180/PI);
    // ROS_INFO("pitch = %.7f   %.7f", gyro_pitch,
gyro_pitch * 180/PI);
    // ROS_INFO("roll = %.7f   %.7f", gyro_roll,
gyro_roll * 180/PI);

    // ROS_INFO("average = %.7f   %d", avr_gyro /
avr_counter );

    namedWindow("position", WINDOW_AUTOSIZE);
    createTrackbar("x pos", "position", &pos_x, 800);
    createTrackbar("y pos", "position", &pos_y, 600);

    spinOnce();
    waitKey(1);
    loop_rate.sleep();
}
}

```


Halaman ini sengaja dikosongkan

BIODATA PENULIS



Dzulfikar Ahmad Samhan adalah seorang mahasiswa ITS yang berperan sebagai *Programmer* di tim IRIS. Lahir di Mojokerto pada 21 juni 1997, dan menjalani masa pendidikan SMA di SMAN 1 SOOKO

Email:

dzulfikar.ahmad.smhn@gmail.com