SKRIPSI

RANCANG BANGUN SISTEM NAVIGASI ROBOT OTONOM *WAITER-BOT* BERBASIS *ROBOT OPERATING SYSTEM*

Disusun dan diajukan oleh:

ABD. SALAM

D041191099

PROGRAM STUDI SARJANA TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

GOWA

2023

LEMBAR PENGESAHAN SKRIPSI

RANCANG BANGUN SISTEM NAVIGASI ROBOT OTONOM *WAITER-BOT* BERBASIS *ROBOT OPERATING SYSTEM*

Disusun dan diajukan oleh

ABD. SALAM

D041191099

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka
Penyelesaian Studi Program Sarjana Program Studi Teknik Elektro
Fakultas Teknik Universitas Hasanuddin
Pada tanggal 13 Oktober 2023
Dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,

Muh Anshar, S.T., M.Sc (Research), Ph.D.
NIP. 197705172005011003

Pembimbing Pendamping,

Prof. Dr. Ir. Andani Achmad, M.T.
NIP. 196012311987031022

Ketua Program Studi,

Dr. Eng. Ir. Dewiani, MT.
NIP. 196910261994122001

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

Nama : Abd. Salam

NIM : D041191099

Program Studi : Teknik Elektro

Jenjang : S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

RANCANG BANGUN SISTEM NAVIGASI ROBOT OTONOM *WAITER-BOT* BERBASIS *ROBOT OPERATING SYSTEM*

Adalah karya tulisan saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain dan bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

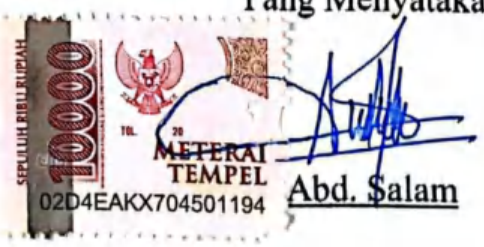
Semua informasi yang ditulis dalam skripsi yang berasal dari penulis lain telah diberi penghargaan, yakni dengan mengutip sumber dan tahun penerbitannya. Oleh karena itu semua tulisan dalam skripsi ini sepenuhnya menjadi tanggung jawab penulis. Apabila ada pihak manapun yang merasa ada kesamaan judul dan atau hasil temuan dalam skripsi ini, maka penulis siap untuk diklarifikasi dan mempertanggungjawabkan segala resiko.

Segala data dan informasi yang diperoleh selama proses pembuatan skripsi, yang akan dipublikasi oleh Penulis di masa depan harus mendapat persetujuan dari Dosen Pembimbing.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Gowa, 12 Oktober 2023

Yang Menyatakan



Abd. Salam

ABSTRAK

ABD. SALAM. *Rancang Bangun Sistem Navigasi Robot Otonom Waiter-Bot Berbasis Robot Operating System (ROS)* (dibimbing oleh Muh Anshar dan Andani Ahmad)

Perkembangan teknologi robotika semakin masif diterapkan di industri manufaktur seiring memasuki era revolusi industri 5.0 dalam membantu meringankan beban manusia serta meningkatkan efisiensi waktu dan tenaga. Salah satu bentuk penerapannya pada penelitian robot otonom *Waiter-Bot* sebagai robot cerdas bebasis *Robot Operating System* (ROS) dengan misi pengantaran logistik secara *autonomus*. Penelitian ini menggunakan metode kuantitatif dengan pengujian robot melakukan perencanaan jalur navigasi dalam mencapai misi. Metode perencanaan jalur navigasi terdiri atas perencanaan jalur lokal *(local planner)* dan jalur global *(global planner)* untuk memperoleh jalur trayektori lintasan terdekat dan teraman. Dari pengamatan deviasi kecepatan pada robot diperoleh nilai ideal untuk perintah kecepatan robot otonom yakni 0-0,23 m/s dengan persentasi error tertinggi 0,09%. Kemudian kemampuan navigasi robot pada lingkungan statis, robot dapat menempuh misi A, B dan C dengan waktu tempuh rata-rata 3 menit 81 detik. Sedangkan kemampuan robot melakukan navigasi pada skenario lingkungan dinamis, robot menempuh misi A, B dan C dengan waktu tempuh rata-rata 4 menit 12 detik. Dalam sistem navigasi robot berbasis ROS pada penelitian ini diketahui jumlah *node* yang aktif yakni 10 *node* dalam 7 topik. Dengan demikian navigasi *Waiter-Bot* dapat menjalankan misi dengan efektif.

Kata Kunci: robot, navigasi, otonom, ROS, *local planner, global planner*

ABSTRACT

ABD. SALAM. *Design of Navigation System for Waiter-Bot Autonomous Robot Based on Robot Operating System (ROS) (supervised by Muh Anshar and Andani Ahmad)*

The development of robotics technology is increasingly massively applied in the manufacturing industry as it enters the era of the Industrial Revolution 5.0 in helping to ease the burden on humans and increase time and labor efficiency. One form of application is in the research of the *Waiter-Bot* autonomous robot as a Robot Operating System (ROS)-based intelligent robot with an autonomous logistics delivery mission. This research uses a quantitative method by testing the robot to plan the navigation path in achieving the mission. The navigation path planning method consists of local path planning (local planner) and global path (global planner) to obtain the closest and safest trajectory path. From the observation of speed deviation on the robot, the ideal value for autonomous robot speed command is 0-0.23 m/s with the highest percentage error of 0.09%. Then the robot's navigation capabilities in a static environment, the robot can travel missions A, B, and C with an average travel time of 3 minutes 81 seconds. While the robot's ability to navigate in dynamic environment scenarios, the robot takes missions A,B, and C with an average travel time of 4 minutes and 12 seconds. This study's ROS-based robot navigation system shows that the number of active nodes is 10 nodes in 7 topics. Thus the *Waiter-Bot* navigation can carry out the mission effectively.

Keywords: robot, navigation, autonomous, ROS, local planner, global planner

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

KATA PENGANTAR

Puji syukur terpanjatkan kehadirat Allah *subhanahu wata'ala* atas limpahan rahmat dan karunia-Nya sehingga penyusunan skripsi tugas akhir ini dapat terselesaikan. Shalawat serta salam tak lupa tercurahkan kepada baginda Rasulullah *sallallahu 'alaihi wasallam.* Penyelesaian skripsi ini merupakan upaya penulis dalam memenuhi salah satu syarat guna memperoleh gelar Sarjana Teknik di Departemen Teknik Elektro Fakultas Teknik Universitas Hasanuddin.

Skripsi ini penulis persembahkan kepada seluruh pembaca dan pengembang riset selanjutnya sebagai literatur studi Pustaka. Kepada yang terkhusus untuk kedua orang tua dan kakak penulis yang telah mendidik dan senantiasa selalu mendoakan dan mendukung penulis selama ini. Semoga ini menjadi salah satu bentuk *wasilah birrul walidayn* kepada orang tua dan seuruh keluarga penulis.

Skripsi ini berjudul Rancang Bangun Sistem Navigasi Robot Otonom *Waiter-Bot* Berbasis *Robot Operating System.* Pelaksanaan rangkaian penelitian dan penyusunan skripsi ini juga tak terlepas dari seluruh bantuan, bimbingan, nasehat dan doa dari berbagai pihak. Sehubungan dengan hal tersebut, maka penulis hendak mengucapkan terima kasih kepada:

1. Bapak Dr. Amil Ahmad Ilham, S.T., M.IT. selaku Wakil Dekan Bidang Akademik dan Kemahasiswaan Fakultas Teknik yang berkenan memberikan arahan dan secara personal banyak memberikan perhatian dan dukungan kepada penulis.
2. Ibu Dr. Eng. Ir. Dewiani,MT. selaku ketua Departemen Teknik Elektro atas izin dan dukungan moril dan materil dalam pelaksanaan penelitian penulis.
3. Bapak Muh Anshar, ST., M.Sc(Research), Ph.D. selaku Pembimbing 1 atas bimbingan dan segala bentuk perhatian dan dorongan kepada penulis, serta Prof. Dr. Ir. Andani Achmad, S.T., M.T., selaku Pembimbing II dengan kerendahan hati mempercayakan penulis sebagai mahasiswa bimbingan penyelesaian tugas akhir ini.
4. Para dosen penguji, Ibu Dr. Andi Ejah Umraeni Salam, S.T., M.T. dan Ibu la Rachmaniar Sahali, ST., M.T. atas segala saran dan masukan kepada enulis dalam menyelesaikan tugas akhir ini.

5. Seluruh tenaga pendidik dan sivitas akademika Departemen Teknik Elektro dan Fakultas Teknik Unhas atas segala bentuk ketulusan pelayanan selama penulis menempuh perkuliahan.
6. Penasehat akademik penulis, Prof. Baharuddin Hamzah, Prof. Wihardi Tjaronge, Bapak Muh. Ramli, Bapak Aries Subagiyo dan Abah Nusran atas segala nasehat, kasih sayang dan pelajaran hidup kepada penulis.
7. Penasehat karir dan pengembangan diri, Bapak Mukti Ali, Bapak Muhammad Rusman, Bapak Yusran, Pak Faisal Mahmuddin, Ibu Intan Sari Areni dan Ibu Nadzirah Ikasari atas segala bentuk keikhlasan, dedikasi dan berkenan membimbing penulis dalam berproses dan bertumbuh selama ini.
8. *The honorable mention* sahabat penulis, Fariz, Hasbih, Arya, Iqrimah, Ocan, Dennis dan ikhwah MADZ 2019 serta grup riset *Cognitive Social Robotics and Advanced Artificial Intelligence Centre* (CSR-2AIR) atas segala bentuk solidaritas dan kekeluargaan selama perkuliahan dan pekerjaan proyek.
9. Keluarga Yayasan Al Fityah, *Super Team* COV.id, Ikhwah Al Muhandis, MADZ DE FT-UH dan jamaah Masjid Al Ilmi IKATEK Unhas atas kepercayaan kepada penulis serta menjadi wadah berproses dan bertumbuh.
10. Teman seperjuangan TR19GER, Teknik Elektro Angkatan 2019 yang telah membersamai penulis selama menempuh perkuliahan. Serta para senior dan junior yang tak dapat penulis sebutkan satu-persatu. *Barakallahu fiikum.*

Penulis menyadari bahwa masih terdapat banyak kekurangan dalam skripsi ini, oleh karennya segala bentuk saran dan masukan yang membangun dari semua pihak. Akhir kata penulis berharap semoga skripsi dapat diterima sebagai aktualisasi tri dharma penulis yang dapat memberikan manfaat bagi penulis dan pembacanya.

Gowa, 12 Oktober 2023

Penulis,

Abd. Salam

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi seiring dengan tuntutan kebutuhan hidup akan pelayanan berubah, kebutuhan serta kesibukan juga meningkat. Pekerjaan manusia dituntut lebih efisien dalam menggunakan waktu untuk memenuhi kebutuhan hidupnya sehingga dapat lebih produktif untuk bekerja. Implementasi robot di industri komersial dan banyak perusahaan lain telah bahwa pengembangan robotika sosial di masa depan sangat potensial. Hal ini membuat banyak usaha jasa yang menyediakan pelayanan dalam pemenuhan kebutuhan manusia bermunculan contohnya usaha restoran. Di sebuah restoran terdapat pelayan yang akan memberikan pelayanan kepada pelanggan. Kinerja pelayan manusia dipengaruhi oleh banyak faktor, antara lain manusia yang masih memiliki rasa lelah dan letih sehingga tidak dapat bekerja secara terus-menerus. Manusia dapat jatuh sakit sehingga tidak dapat melaksanakan tugasnya. Begitupun dalam pelayan rumah sakit telah dikembangkan robot yang terdiri dari robot *Telemedicine (TEL), robot Disinfection (DIS), dan robot Assistance (ASL),* robot dengan jenis *Assistance (ASL)* akibat kebutuhan akan efektifitas pelayanan yang optimal.

Dalam sektor industri tercatat sekitar 422.000 unit robot di seluruh dunia dan diperkirakan akan terus bertambah sebanyak 12% per tahun hingga tahun 2022 (IFR, 2018). *Autonomous mobile robot* merupakan salah satu jenis robot yang berkembang saat ini. Teknologi yang dapat diterapkan untuk membantu dan meringankan pekerjaan pelayanan restoran adalah dengan memanfaatkan kecanggihan robot cerdas. Robot ini harus memiliki kemampuan untuk membawa logistik, seperti mengantar makanan, barang, baju, dan/atau obat-obatan (Sierra Marín dkk., 2021). Permasalahan yang dihadapi sebuah *autonomous mobile robot* adalah masalah navigasi karena robot otonom harus mengenali lingkungannya. Oleh karena itu, penelitian ini mengangkat studi rancang bangun *Smart Waiter-Bot* sebagai robot cerdas yang dirancang untuk dapat melakukan pengantaran makanan onom pada titk tujuan dalam lingkungan dalam ruangan. Penelitian ini ın pengembangan dari navigasi robot kursi beroda penyadang disabilitas bir, 2019) dengan pengoperasian sistem navigasi berbasis ROS agar dapat

beroperasi secara *autonomous.* Sistem robot yang dibangun terdiri dari beberapa komponen diantaranya kamera kinect, sensor *encoder*, MPU-90, dan HCSR yang dijalankan berbasis *middleware Robot Operating System* (ROS). ROS digunakan karena setiap grup riset robotika umumnya berfokus pada topik-topik riset robot tertentu sehingga ROS hadir dalam memudahkan pengembangan suatu topik riset tertentu tanpa harus memikirkan topik riset lainnya. ROS mencakup *tools, libraries* dan *convention* yang bertujuan untuk menciptakan serta mengembangkan robot yang kompleks. Dengan begitu grup riset yang mengembangkan sistem navigasi tidak perlu harus memikirkan pula sistem biomekanik dari robot tersebut melainkan hanya mengembangkan salah satu bagian dari sistem navigasi tersebut tanpa harus membuat sistem navigasi dari dasar. Dalam pengembangan sistem navigasi *Waiter-Bot* secara otonom digunakan metode perencanaan jalur salah satunya dengan metode algoritma *Dynamic Window Approach* (DWA). Selain itu, diperlukan pula sebuah peta yang presisi agar dapat melakukan navigasi, sehingga robot perlu membangun peta dan lokalisasi lingkungannya terlebih dahulu.

1.2 Rumusan Masalah

Adapun rumusan masalah dalam penelitian ini sebagai berikut:

a. Bagaimana deviasi dan respon kecepatan robot otonom dalam mengeksekusi perintah nilai kecepatan pada *ROS Navigation Stack* untuk melakukan navigasi?
b. Bagaimana kemampuan navigasi robot otonom *Waiter-Bot* dalam skenario pengujian lingkungan statis dan dinamis?
c. Bagaimana sistem komunikasi antar node dalam sisem navigasi robot otonom *Waiter-Bot* berbasis *Robot Operating System* (ROS)?

1.3 Tujuan Penelitian

Adapun tujuan penelitian ini sebagai berikut:

a. Menganalisis deviasi dan respon kecepatan robot otonom dalam mengeksekusi perintah nilai kecepatan pada *ROS Navigation Stack* untuk melakukan navigasi.
b. Menguji kemampuan navigasi robot otonom *Waiter-Bot* dalam skenario pengujian lingkungan statis dan dinamis.
c. Menganalisis sistem komunikasi antar node dalam sisem navigasi robot otonom *Waiter-Bot* berbasis *Robot Operating System* (ROS).

1.4 Manfaat Penelitian

Dari penelitian ini, diarapkan dapat memperoleh manfaat sebagai berikut:

a. Menjadi referensi penelitian pengembangan robot otonom berbasis ROS.
b. Memberikan wawasan baru bagi para pengembang robot dalam memafaatkan *framework* ROS untuk topik riset kendali dan robotika modern.
c. Merepresentasi perkembangan teknologi robotika dalam implementasi di berbagai sektor otomasi industri.

1.5 Batasan Masalah

Agar penelitian ini lebih terarah dan terukur, maka difokuskan dalam ruang lingkup sebagai berikut:

a. Peneltian ini merancang mekanik robot otonom *Waiter-Bot* bagian *low level* dan *high level.*
b. Pengujian sistem navigasi robot otonom *Waiter-Bot* dilakukan pada lingkungan *indoor.*
c. Penelitian ini fokus pada pengujian sistem navigasi robot otonom *Waiter-Bot* bagian *navigation stack* berbais ROS.
d. Pengujian kemampuan sistem navigasi *Waiter-Bot* terbatas dalam pemberian perintah target melalui PC pada RViz.

1.6 Metode Penelitian

Metode penelitian yang dilakukan pada tugas akhir ini adalah:

1. Studi literatur

   Pada tahapan ini yang dilakukan adalah mencari sumber-sumber referensi dan beberapa materi pendukung yang dijadikan sebagai landasan dalam penelitian ini, sebelum melakukan penerapan dan pengujian sistem secara langsung.

2. Pengujian

   Pada tahap pengujian ini yang dilakukan adalah penerapan dan pengujian langsung berbasis simulasi menggunakan beberapa skenario dengan tujuan untuk mendapatkan data yang tepat dari hasil pengamatan.

3. Analisis Data

   isis data dilakukan dengan tujuan untuk melihat data yang diperoleh dari ujian sistem navigasi robot otonom dengan menggunakan beberapa ario pengujian.

4. Diskusi dan Konsultasi

   Diskusi dan konsultasi dilakukan secara daring maupun luring kepada dosen pembimbing maupun pihak-pihak yang berkompeten dibidangnya untuk mendapatkan pengetahuan mengenai penelitian yang dilaksanakan.

5. Simpulan

   Simpulan merupakan tahap akhir dari penelitian ini yang diperoleh setelah dilakukan pengambilan dan analisa data mengenai semua permasalahan yang telah dibahas.

1.7 Sistematika Penulisan

Adapun sistematika penulisan laporan Tugas Akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, metode penelitian, dan sistematika penulisan laporan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi teori-teori penunjang terkait penelitian yang dilakukan yang diambil dari berbagai sumber ilmiah yang digunakan dalam penulisan laporan tugas akhir ini.

BAB III METODOLOGI PENELITIAN

Bab ini membahas mengenai rancangan penelitian, waktu dan tempat penelitian, diagram alir perencanaan, dan alat dan bahan yang digunakan pada penelitian tugas akhir ini.

BAB IV HASIL DAN PEMBAHASAN

Bab ini membahas mengenai pengujian dan performa navigasi *Waiter-Bot* secara otonom dengan pendekatan jalur lokal dan jalur global.

BAB V PENUTUP

Bab ini berisi kesimpulan terhadap hasil pengujian dan saran untuk penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terkait

2.1.1 Implementasi Sistem Robot Otonom dengan Sensor Kinect menggunakan *Algoritme Gmapping* dan *Timed Elastic Band*

Peneltian ini mengangkat studi kasus pelayan rumah sakit menggunakan teknologi robot. Atas hal ini diperlukan sistem robot bergerak otonom pada rumah sakit untuk membantu tenaga kerja, terutama dalam hal logistik. Sistem terdiri dari beberapa komponen diantarnya sensor kinect, *encoder*, IMU, dan dengan middleware *Robot Operating System* (ROS). Keunggulan pada penelitian ini adalah bentuk dari purwarupa robot yang memiliki ukuran yang besar. Untuk menciptakan navigasi secara otonom diperlukan metode perencanaan jalur, maka dari itu digunakan algoritme *Timed Elastic Band (TEB).* Sebelum melakukan navigasi, diperlukan sebuah peta yang presisi, maka dari itu dibutuhkan metode pembuatan peta menggunakan algoritme GMapping. Hasil uji pada navigasi menunjukkan akurasi sebesar 50% dari 10 percobaan yang telah berhasil 5 kali. Navigasi dilakukan dengan waktu komputasi algoritme TEB 2,77 detik dan rata-rata waktu pergerakan robot yang berhasil menuju koordinat tujuan adalah 129,98 detik. Untuk pemetaan didapatkan akurasi sebesar 50% dengan akurasi ukuran peta sebesar 98,84% (Arifin, dkk., 2022).

2.1.2 Sistem Navigasi *Waypoint* Pada *Autonomous Mobile Robot*

Dalam penelitian Ahmad Sulkhan Taufik (2013) dirancang sistem navigasi luar ruang berbasis posisi dengan metode *waypoint.* Sistem navigasi diterapkan pada *autonomous mobile robot* yang bergerak di darat. Sistem navigasi dirancang agar *autonomous mobile robot* mampu mengenali posisi dan arah berdasarkan sistem koordinat bumi, mampu malakukan koreksi arah gerak *(bearing correction)* dan memperkirakan jarak yang telah ditempuh (odometer) untuk meningkatkan akurasi dalam mencapai posisi tujuan, dengan rute yang telah ditentukan oleh operator. Modul GPS *receiver* digunakan sebagai penentu posisi, sedangkan modul *compass* digunakan sebagai penentu arah dalam sistem navigasi. Hasil menunjukan bahwa sistem navigasi *waypoint* mampu mengatur gerak

autonomous mobile robot dalam mencapai posisi tujuan dengan akurasi sebesar 11 meter, serta mampu melakukan odometer dengan akurasi sebesar 0,5 meter.

2.1.3 Navigasi Robot Otonom Berbasis ROS Menggunakan Sensor Lidar 2D dan Kamera RGB-D

Penelitian ini berisi tentang implementasi robot bergerak otonom dengan *Robot Operating System (ROS)*. Sistem ini menggunakan kamera 2D LiDAR dan RGB-D dengan ROS 2D *navigation stack*, dengan konsumsi daya yang rendah dan komputer onboard yang murah. Untuk perangkat lunak, digunakan paket ROS resmi dengan perubahan parameter *default* yang minimal. Untuk perangkat keras, keterbatasan peralatan dan pengaturan sistem adalah salah satu tantangan. Sistem yang dibangun bertujuan agar robot dapat melakukan navigasi dengan kemampuan penghindaran rintangan yang dinamis. Fokus pengujian dalam penelitian ini yakni dua pengaturan sistem dari tumpukan navigasi ROS yang diusulkan. Sistem pertama diimplementasikan pada *Raspberry Pi 3* dengan menggunakan LiDAR 2D saja. Sistem kedua diimplementasikan pada Intel NUC menggunakan LiDAR 2D dan kamera RGB-D. Untuk mengevaluasi kinerja, pengujian fungsional dilakukan dalam beberapa percobaan. Hasil percobaan tersebut menunjukkan bahwa robot dapat menghindari objek yang menghalangi atau berhenti jika tidak dapat dihindari (Gatesichapakorn, *et.al*, 2019).

2. 2 Sistem Kendali

Sistem Kendali atau *control system* terdiri dari dua kata yaitu *system* dan *control*. Sistem berasal dari bahasa latin *(systema)* dan bahasa Yunani *(sustema)* adalah suatu kesatuan yang terdiri dari komponen atau elemen yang dihubungkan bersama untuk mencapai suatu tujuan tertentu. *Control* itu memiliki arti mengatur, mengarah dan mengendalikan. Jadi *system control* adalah hubungan antara komponen-komponen fisik yang membentuk suatu konfigurasi sistem sehingga memberikan hasil yang diharapkan yang dapat dipraktekkan secara otomatis. Adapun jenis-jenis sistem kendali terdiri atas dua macam yakni sistem kendali rbuka *(open loop)* dan kalang tertutup *(close loop)*.

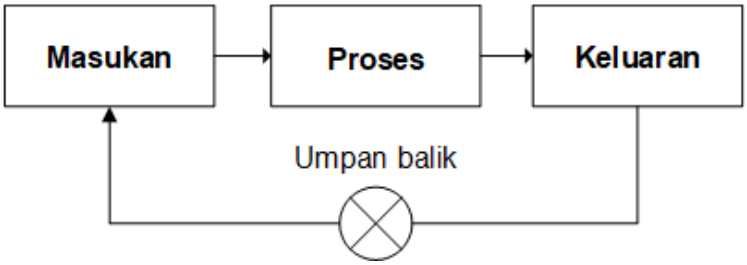
2.2.1 Sistem *Open Loop*

Sistem kendali *open loop* adalah sebuah sistem yang tidak memiliki umpan balik *(feedback),* sehingga bila terdapat gangguan dari dalam maupun dari luar maka sistem tidak dapat melaksanakan tugas seperti yang diharapkan. Suatu sinyal masukan diberikan ke dalam sistem kendali dimana keluarannya bertindak sebagai sinyal penggerak dimana sinyal penggerak ini yang kemudian menghasilkan proses yang akan dikendalikan untuk menghasilkan *output* yang diinginkan. Proses *open loop* dapat dilihat pada Gambar 1.

Gambar 1 *Sistem open loop*

2.2.2 Sistem *Close Loop*

Sistem kendali *close loop* adalah sistem kontrol yang memiliki *feedback,* berbeda dengan sistem *open loop*. Pada bagian *output* dari sistem kontrol ini akan dikirim kembali untuk dibandingkan dengan input yang diberikan. Bila masih terdapat selisih antara *output* dan input, maka sistem masih memiliki eror. Eror merupakan selisih antara input dan *output* atau sistem belum mencapai hasil yang diinginkan. Proses kerja dari sistem *close loop* dapat dilihat pada Gambar 2.



Gambar 2 *Sistem close loop*

2.3 Probabilistik Dalam Robotika

Dalam *Probabilistic Robotics* berdasarkan peneliti Inggris, Thrun, *et.al.* (2005) memberikan penjelasan mengenai ketidakpastian *(uncertainty)* dalam dunia robotika meliputi:

1. Lingkungan

la proses ekplorasi lingkungan yang dilakukan oleh robot. lingkungan ;ap bersifat dinamik, sangat luas dan tidak dapat diprediksi. Komposisi misalnya tumbuhan, angin, air, manusia menjadi satu kondisi yang

memberikan permasalahan dinamik bagi robot untuk memahami lingkungan tersebut.

2. Sensor.

   Sensor atau indra bagi robot juga bersifat dinamik. Kita tidak dapat mengganggap bahwa sensor untuk robot adalah ideal (tidak ada eror). Keterbatasan sensor meliputi jangkauan *(range)* dan resolusi (ketepatan / kepresisian) untuk menyatakan satuan dari pengukuran. Sebagai contoh sensor jarak yang mempunyai beam (lebar pancaran) yang bermacam-macam, tentunya galat pengukuran yang kita dapatkan juga beragam. Contoh selanjutnya kamera yang sangat dipengaruhi oleh faktor pencahayaan, resolusi, kedalaman serta *noise-noise* yang berasal dari lingkungan juga mempengaruhi hasil pengukuran.

3. Aktuator

   Akurasi dari sebuah aktuator juga menjadi penyebab dalam ketidakpastian dalam dunia robotika untuk menyelesaikan suatu permasalahan yang bersifat ideal. Sebagai contoh pada arm robot, motor penggerak dengan derajat gear memberikan pengaruh error letak / posisi dari EoE *(End of Effector)*.

4. Komputasi.

   Robot diharapkan menjadi sistem yang realtime dalam mengolah suatu proses. Tapi dalam kenyataannya fungsi waktu dibutuhkan sehingga mempengaruhi proses yang terjadi. Jika suatu robot mempunyai komputasi yang tinggi maka suatu proses dapat diselesaikan dalam waktu yang cepat. Misalnya robot dengan kemampuan 100 MIPS *(Million Instruction Per Second)* pasti lebih cepat dalam mengolah proses dibandingkan robot dengan kemampuan 10 MIPS *(Million Instruction Per Second)*.

Dalam *probabilistics robotics* sejumlah variabel seperti pengukuran sensor, kontrol, dan posisi robot diasumsikan sebagai variabel yang tidak pasti artinya meskipun dalam pengukuran kita mendapatkan nilai, tapi nilai tersebut mempunyai *range* (jangkauan) yang terkadang sewaktu-waktu juga dapat berubah. Meskipun tidak pasti atau tidak dapat diprediksi tetapi nilai tersebut tetap kita butuhkan untuk uah proses. Pada pembahasan ini akan dijelaskan contoh permasalahan adapi robot pada lingkungan yang bersifat *undeterministic* dan san tentang simbol dan notasi yang dipakai dalam *probabilistic robotics*.

2.4 Robot Otonom

Robot Otonom *(Autonomous Robot)* adalah robot yang dapat melakukan tugas-tugas yang diinginkan dalam lingkungan yang tidak terstruktur tanpa bimbingan manusia terus menerus berdasarkan logika-logika yang diberikan manusia kepada robot. Banyak jenis robot memiliki beberapa tingkat otonomi. Tingkatan otonomi sangat diinginkan dalam bidang-bidang seperti eksplorasi ruang angkasa, membersihkan lantai, memotong rumput, dan pengolahan air limbah (Anggoro, 2013). Salah satu contoh *autonomous robot* adalah *Waiter-Bot* dapat dilihat pada Gambar 3.

Gambar 3 Contoh robot otonom; robot pelayan

Service robot merupakan robot yang digunakan untuk melayani kebutuhan manusia sehari-hari. Robot ini digunakan untuk membantu pekerjaan yang kotor, berbahaya, berulang-ulang dan termasuk pekerjaan rumah tangga termasuk melakukan pengantaran paket dan logistik.

2.5 *Robot Operating System* (ROS)

Robot Operating System atau biasa disingkat ROS adalah sebuah *middleware* sistem robotika yang berisi berbagai koleksi *software framework* untuk pengembangan perangakat lunak robotik. ROS bukanlah sebuah sistem operasi seperti *Windows, MacOS, Android* atau sistem operasi lain yang kita kenal. Namun sebuah perangkat lunak yang menyediakan berbagai *service* yang didesain untuk *heterogenous computer cluster* seperti: abstraksi perangkat keras, pengaturan divais dah, implementasi fungsionalitas yang sering digunakan dalam robot, aian pesan antar proses dan menejemen *package. Software* pada ROS bagi menjadi tiga grup: tools untuk mendistribusikan perangkat lunak

berbasis ROS, implementasi ROS *client library* seperti *roscpp, rospy, roslisp*, dan yang terakhir *package* yang berisi kode pemnghubung dengan aplikasi fungsional robot. *Client libraries* ROS utamanya diperuntukkan untuk sistem UNIX. Untuk hal ini Ubuntu Linux terdaftar sebagai *"supported"* sedangkan varian lain seperti *Fedora Linux, macOS*, serta *Windows* terdaftar sebagai *"experimental"*. Dalam ROS terdapat beberapa istilah yaknik *package, node* dan topik (Ilham, 2019).

Di dalam ROS juga memuat *tools* dan *library* yang bisa digunakan untuk mengembangkan berbagai macam program untuk sistem robot (Rahman, 2020). Tujuan penggunaan ROS juga untuk memudahkan para pengembang robot dalam membuat sistem robot yang di inginkan tanpa harus membuat pengkodean dari awal serta dapat mengembangkan kode sumber bersama – sama (Jalil, 2018). ROS dapat dibagi menjadi 3 bagian yaitu level *file* systems, level grafik komputasi dan level komunitas. Di dalam level *file systems* ROS memuat *packages, metapackages, packages manifests, metapackage manifests, message* dan *service*. Di dalam level grafik komputasi memuat level *node*s, master, parameter server, *message*, *topic*, *services*, dan *bags*. Dan di dalam level komunitas ini juga bisa memisahan *resources* ROS agar dapat saling bertukar pengetahuan dan perangkat lunak. *Resources* ini terdiri dari distribusi *ROS, repository, ROS wiki, bug ticket system, maling list*, *ROS answer and blog* (Jalil, 2019).

2.5.1 ROS *Package*

Perangkat lunak pada ROS terorganisir di dalam *packages*. Didalam *packages* tersebut dapat berisi ROS *Node*, *library* independen, dataset, file konfigurasi dan hal lain yang mendukung modul tersebut. Tujuan dari *packages* adalah untuk menyediakan fungsionalitas yang mudah digunakan dengan tujuan menjadikan perangkat lunak yang dapat dengan mudah digunakan kembali *(reuse)* oleh komunitas pengguna ROS.

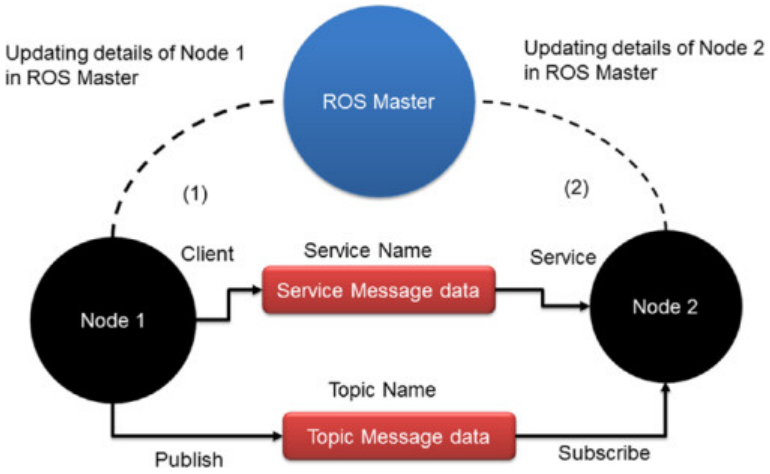
2.5.2 ROS *Node*

Sebuah *node* dalam ROS adalah sebuah proses yang melakukan suatu komputasi. Berbagai *node* ini terkombinasi bersama menjadi suatu *graph* dan [illegible]nikasi dari satu *node* ke *node* lain menggunakan aliran *topics, services*, dan [illegible]r server. *Node* ini diperuntukkan untuk beroperasi pada level yang sangat [illegible]ebagai contoh sebuah sistem kendali robot biasanya terdiri dari banyak [illegible]a *node* mengendalikan laser *range finder*, satu *node* mengendalikan motor,

satu *node* lokalisasi, satu *node* untuk perencanaan jalur *(path planning),* satu *node* untuk *monitoring* dan seterusnya. Penggunaan *node* ini memberi manfaat antara lain: toleransi kegagalan yang terisolasi pada satu fungsionalitas tertentu pada individu *node*, kompleksitas dari kode berkurang dibanding sebuah sistem *monolithic, reusability* serta *maintainability* tinggi dengan penyederhanaan berdasarkan fungsionalitas-fungsionalitas tersebut. Selain itu sebuah *node* dapat dikembangkan menjadi sebuah *nodelet*, yaitu sebuah *node* yang dapat dipanggil dari dalam kode program lainnya menggunakan modul *nodelet manager.*

2.5.3 ROS *Topic*

Topic adalah komponen yang berfungsi sebagai media pertukaran pesan, dalam ROS dikenal dengan istilah *message* yang dinama topik sebagi meda antar *node* yang dinamai sesuai isi dari pesan tersebut. Sistemnya menggunakan *anonymous publish/subscribe semantics*. *Node* yang membutuhkan informasi dari suatu *topic* akan melakukan *subscribe* dan *node* yang menyediakan informasi akan melakukan *publish*. Jumlah *node* yang melakukan *subscribe* ataupun *publish* pada suatu *topic* tidak terbatas jumlahnya. Setiap *topic* memiliki tipe datanya masing-masing sesuai tipe ROS *message* yang di bawanya. Kita dapat membuat tipe *message* sendiri sesuai kebutuhan kita, atau menggunakan tipe *message* yang disediakan oleh ROS *client library*. ROS *topic* juga mendukung komunikasi via TCP/IP atau UDP.



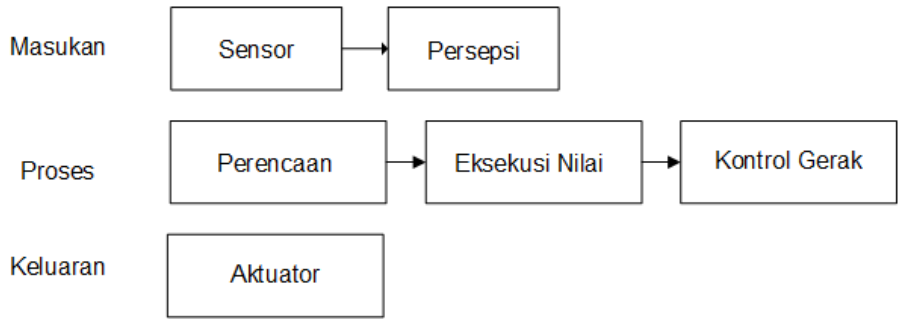
Gambar 4 Blok diagram sistem komunikasi ROS

la Gambar 4 menunjukkan dua program yang ditandai sebagai *node* 1 dan Ketika salah satu program dimulai, sebuah *node* berkomunikasi dengan

program ROS yang disebut master ROS. *Node* mengirimkan semua informasinya ke master ROS, termasuk jenis data yang dikirim atau diterima. *Node* yang mengirim data disebut *node publisher* dan *node* yang menerima data disebut *node subscirber*. ROS Master memiliki semua informasi *publisher* dan *subscirber* yang berjalan di komputer. Jika *node* 1 mengirimkan data tertentu yang disebut "A" dan data yang sama diperlukan oleh *node* 2, maka master ROS mengirimkan informasi ke *node* sehingga mereka dapat berkomunikasi satu sama lain.

Node pada ROS dapat mengirim berbagai jenis data satu sama lain, yang mencakup tipe data seperti *integer, float, string*, dan sebagainya. Berbagai tipe data yang dikirim disebut *message* ROS. Dengan pesan ROS, kita dapat mengirim data dengan satu tipe data atau beberapa data dengan tipe data yang berbeda. Pesan-pesan ini dikirim melalui bus pesan atau jalur yang disebut topik ROS. Setiap topik memiliki nama; misalnya, topik bernama "*chatter*" mengirimkan pesan *string.* Ketika sebuah *node* ROS menerbitkan sebuah topik, *node* tersebut mengirimkan topik ROS dengan pesan ROS, dan memiliki data dengan tipe pesan. Pada Gambar 4-12, topik ROS menerbitkan dan berlangganan *node* 1 dan *node* 2. Proses ini dimulai ketika master ROS saling bertukar detail *node* (Joseph, 2018).

2.6 Navigasi Robot Berbasis ROS



Gambar 5 Hirarki navigasi robot otonom

Dalam sistem navigasi robot otonom diperlukan proses terlebih dahulu yakni pemetaan dan lokalisasi. Setelah robot berhasil melokalisasi, robot dapat diberikan koordinat tujuan dan menggunakan perencana global untuk merencanakan jalur ke koordinat tersebut. Lingkungan direpresentasikan secara internal sebagai gambar

ua dimensi. Lingkungan berisi lima jenis entitas berupa ruang yang belum , ruang kosong yang diketahui, rintangan, radius inflasi, dan robot itu Radius inflasi adalah penyangga yang memancar keluar dari semua

rintangan. Robot memperlakukan radius inflasi sebagai rintangan dan tidak dapat memetakan jalur yang melaluinya. Radius inflasi sama dengan radius robot itu sendiri dan memiliki efek untuk memastikan bahwa robot selalu menghasilkan jalur dengan ruang yang cukup untuk membersihkan rintangan. Hal ini diimplementasikan sebagai penyangga di sekitar rintangan alih-alih diterapkan pada robot itu sendiri karena alasan kesederhanaan, dan mencapai efek yang sama.

Setelah rencana global dibuat, perencana lokal menerjemahkan jalur ini ke dalam perintah kecepatan untuk motor robot. Perencana lokal melakukan hal ini dengan membuat fungsi nilai di sekitar robot, mengambil sampel dan mensimulasikan lintasan di dalam ruang ini, memberi nilai pada setiap lintasan yang disimulasikan berdasarkan hasil yang diharapkan, mengirimkan lintasan dengan nilai tertinggi sebagai perintah kecepatan ke robot dan mengulanginya hingga tujuan tercapai. Alasan mengapa perencana lokal bekerja dengan cara ini adalah karena perencana lokal ditulis dengan sangat umum untuk digunakan oleh robot yang mungkin memiliki jejak kaki yang tidak beraturan, geometri kemudi *ackerman*, pelengkap, dan konfigurasi lain yang tidak akan bekerja menggunakan algoritma yang dirancang khusus untuk robot dengan bentuk yang sederhana, sistem penggerak diferensial dan tanpa pelengkap. Karena selalu ada sejumlah kecil kesalahan dalam gerakan fisik robot, sangat sulit bagi robot untuk mencapai koordinat tujuannya secara tepat. Hal ini dapat menyebabkan robot berosilasi di sekitar tujuan karena terus berusaha mencapai lokasi yang tepat. Perilaku yang tidak diinginkan ini dapat dielakkan dengan mengatur fungsi *latch* yang menyebabkan robot berhenti ketika berada dalam jarak tertentu dari tujuan (Derect, 2012).

2.6 Perencanaan Jalur Navigasi

Dalam konteks robot otonom, perencanaan jalur mengacu pada penentuan jalur yang layak dan efisien untuk diikuti oleh robot saat bergerak dari satu lokasi ke lokasi lain. Hal ini biasanya melibatkan pertimbangan beberapa faktor, seperti lokasi dan orientasi robot saat ini, tata letak lingkungan (termasuk rintangan apa pun yang harus dihindari robot), dan batasan apa pun pada gerakan robot (seperti ı maksimum atau radius belok). Pendekatan spesifik yang digunakan akan ıg pada karakteristik robot dan lingkungan, serta persyaratan aplikasi. mum, tujuan perencanaan jalur adalah untuk menemukan jalur yang

memungkinkan robot mencapai tujuannya secara efisien dan aman sambil menghindari rintangan dan mematuhi batasan apa pun pada gerakannya. Jalur yang dihasilkan oleh perencana jalur adalah urutan titik-titik jalan diskrit yang dicoba untuk diikuti oleh robot. Setelah jalur direncanakan, robot dapat menggunakan informasi tersebut untuk memandu gerakannya dan menavigasi lingkungan.

Hal ini melibatkan penggunaan algoritma pengambilan keputusan tingkat tinggi untuk beradaptasi dengan kondisi yang berubah dan merespons kejadian tak terduga seperti baterai lemah atau informasi baru tentang lingkungan.Untuk merencanakan jalur, digunakan perencana yang diperluas. Perencana ini menggunakan peta kisi sel yang dibuat oleh peta, peta local *costmap*, dan global *costmap*. Kemudian pada setiap langkah, perencana ini menghitung cost yang dibutuhkan untuk memperluas jalur, lihat Gambar 5. Secara khusus, A* memilih jalur yang meminimalkan,

$$f(n) = g(n) + h(n) \quad (1)$$

di mana, n adalah simpul berikutnya dalam jalur, g(n) adalah *costmap* jalur dari simpul awal ke n, dan h(n) adalah fungsi heuristik yang mengestimasi *costmap* jalur termurah dari n ke tujuan. A $*$ dijamin untuk mengembalikan jalur dengan *costmap* paling rendah dari awal ke tujuan.

2.6.1 *Navigation Stack*

Navigation Stack cukup sederhana pada tingkat konseptual meskipun *navigation stack* dirancang untuk tujuan umum, ada tiga persyaratan perangkat keras utama yang membatasi penggunaannya:

a. Ini dimaksudkan untuk penggerak diferensial dan robot beroda holonomik saja. Ini mengasumsikan bahwa pangkalan bergerak dikendalikan dengan mengirimkan perintah kecepatan yang diinginkan untuk dicapai dalam bentuk: kecepatan x, kecepatan y, kecepatan *theta.*
b. Membutuhkan *laser planner* yang dipasang di suatu tempat di pangkalan bergerak. Laser ini digunakan untuk pembuatan peta dan pelokalan.
c. *Navigation Stack* dikembangkan pada robot otonom, sehingga kinerjanya :an menjadi lebih optimal pada robot yang bergerak secara kompleks. Ini :kerja pada robot dengan bentuk dan ukuran yang berubah-ubah, tetapi

mungkin mengalami kesulitan dengan robot persegi panjang besar di ruang sempit seperti pintu.

Istilah *tf* adalah paket yang memungkinkan pengguna melacak beberapa *frame* koordinat dari waktu ke waktu. Istilah *tf* mempertahankan hubungan antara *frame* koordinat dalam struktur pohon yang disangga dalam waktu dan memungkinkan pengguna mentransformasikan titik, vektor, dll. antara dua frame koordinat pada titik waktu yang diinginkan. Sistem robotik biasanya memiliki banyak bingkai koordinat 3D yang berubah seiring waktu, seperti *frame* lingkungan, *frame* dasar dan sebagainya. *tf* melacak semua area lingkungan ini dari waktu ke waktu dan memungkinkan pengembang untuk mengajukan pertanyaan seperti:

a. Di mana posisi robot terhadap lingkungannya, 5 detik yang lalu?
b. Bagaimana pose objek dalam *gripper* relatif terhadap posisi saya?
c. Bagaimana pose saat ini dari bingkai dasar dalam bingkai peta?

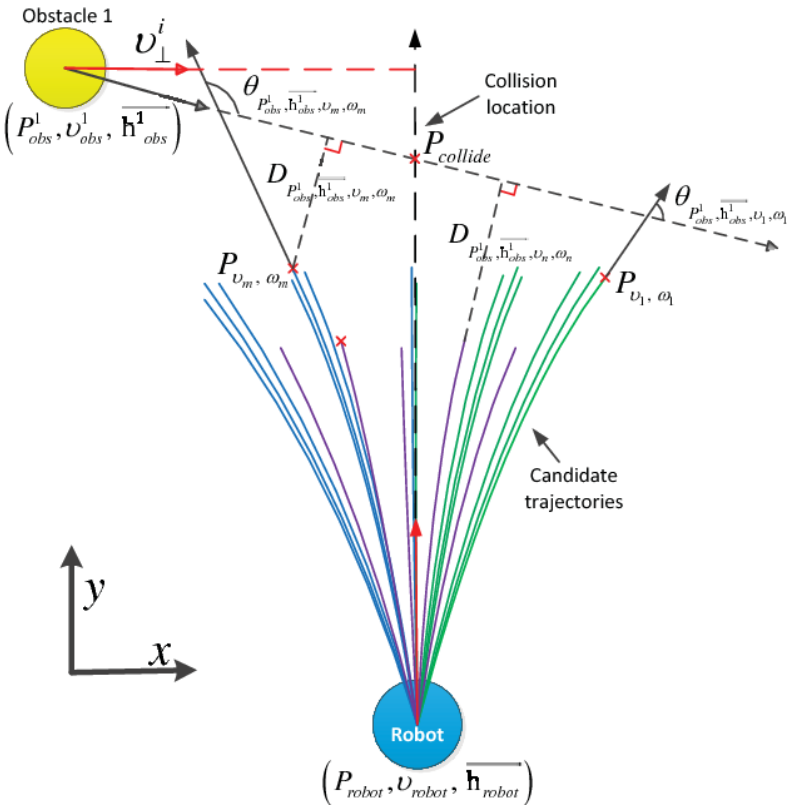
tf dapat beroperasi dalam sistem terdistribusi. Ini berarti semua informasi tentang frame koordinat robot tersedia untuk semua komponen ROS di komputer manapun dalam sistem (*ros.org,* 2018).

2.6.2 *Local Planner*

Local Planner merupakann perencana jalur untuk mengimbangi resolusi peta global yang rendah, perencana lokal dikonfigurasikan untuk menggunakan peta dengan resolusi yang lebih tinggi, yaitu 5 cm per piksel. Karena perencana lokal hanya perlu memproses data peta di area sekitar robot, maka data yang harus diproses secara signifikan lebih sedikit dan kerapatan piksel yang lebih tinggi hanya memiliki sedikit efek pada kinerja. Dalam melakukan perencanaan jalur terpendek dalam cakupan lokal terdapat beberapa metode yang telah ada di *library* ROS misalnya DWA *(Dynamic Window Approach)* yang membuat lintasan berdasarkan peta yang telah dibuat sebelumnya (Dieter, *et.al.* 1997).

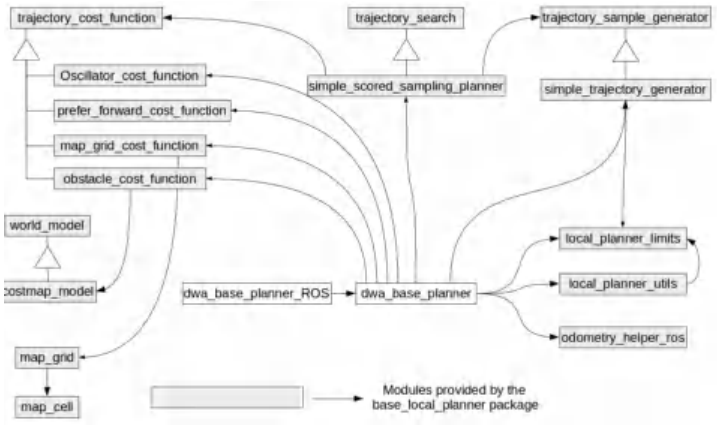
Algoritma DWA akan membuat beberapa macam lintasan dan membuat suatu fungsi nilai untuk menghitung jarak terdekat serta menghindari adanya penghalang enghalang yang statis maupun yang dinamis. Nilai dari jarak yang dibuat an grid yang dibentuk oleh DWA. Fungsi nilai ini berupa dx, dy dan

dtheta yang akan secara simultan dikirimkan ke robot. Ide dasar dari algoritma DWA dapat dilihat pada Gambar 6.



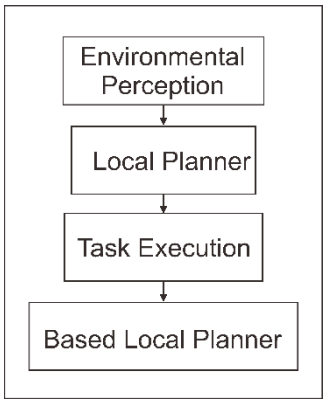
Gambar 6 Visualisasi ide dasar algoritma DWA

Selain menggunakan DWA sebagai penghitung nilai yang akan dilalui, pada base *local planner* juga melakukan simulasi untuk memprediksi serta mencari nilai-nilai terbaik dalam melakukan pencarian jarak terdekat. Terdapat pula beberapa karakteristik yang perlu diperhatikan seperti kedekatan dengan rintangan, kedekatan dengan tujuan kecepatan dan karakteristik lainnya. Setelah dilakukan simulasi maka nilai terbaik yang nantinya akan dipilih oleh robot dalam menentukan lintasannya (Fadil dan Cakra, 2021).



Gambar 7 Sistem navigasi ROS menggunakan algoritma DWA

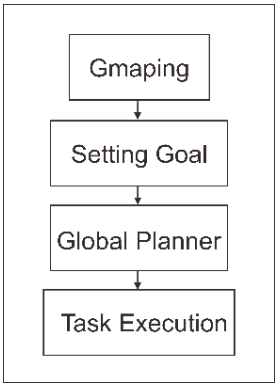
Kutub *encoder* mengukur jarak yang ditempuh oleh robot. Dengan membedakannya, kecepatan roda saat ini dapat diperkirakan. Kemudian, perbedaan antara kecepatan yang diinginkan dan kecepatan saat ini dihitung. Perbedaan ini diberikan pada input pengontrol P sederhana yang bertugas mengubah laju kecepatan yang diinginkan menjadi akselerasi yang diinginkan. Akselerasi ini kemudian dibatasi dan dimasukkan ke dalam pengontrol PID. Pada keluarannya adalah daya motor yang diperlukan untuk mencapai kecepatan tujuan kita. Kontrol PID tambahan ini meningkatkan optimasi pergerakan robot.



Gambar 8 Hirarki sistem *local planning*

2.6.3 *Global Planner*

Dalam menghasilkan jalur yang layak untuk bergerak dari posisi saat ini ke tujuan yang diinginkan, kendala kinematik robot harus diperhitungkan. Sebagian [illegible] encana global saat ini menggunakan representasi diskrit dari keadaan robot ngurangi kompleksitas komputasi dengan mengabaikan kelengkapan. epresentasi diskrit ini menyulitkan untuk memenuhi batasan kemampuan kebanyakan robot.



Gambar 9 Hirarki sistem *global planning*

Keadaan robot diberikan kode dengan posisi (x,y) dan *heading* (θ) untuk memastikan lintasan yang dihasilkan mulus (tanpa perubahan mendadak pada *heading* robot). Namun, kecepatan dan akselerasi dari parameter ini tidak dipertimbangkan. Beberapa algoritma telah dikembangkan untuk mengatasi masalah ini.

2.6.4 *Local Costmap*

Peta *local costmap* digunakan bersama dengan peta *costmap* global dan merepresentasikan pandangan yang lebih lokal dari lingkungan robot. Peta *costmap* lokal diimplementasikan sebagai *plugin* di *Navigation Stack* dan digunakan oleh beberapa *node* lain yang dibahas di sini. Secara bersamaan, mereka menciptakan *potential field* di sekitar semua rintangan di peta dan terlihat oleh kinect. Hal tersebut sudah cukup karena kecepatannya motor yang rendah, tetapi jika robot diharapkan melaju lebih cepat atau rintangan mendekat dengan cepat, *costmap* yang lebih besar bisa diperlukan. Mirip dengan peta *costmap* global, ukuran bidang potensial di sekitar rintangan dapat ditentukan untuk menjaga jarak yang aman. Masalah yang kerap terjadi adalah secara optimal robot harus dapat membedakan antara rintangan untuk menjaga jarak yang lebih jauh dari rintangan yang bergerak seperti orang yang berjalan di area robot.

2.6.5 *Global Costmap*

Peta *costmap* global adalah peta grid yang mewakili pandangan global lingkungan robot dan dibuat sebagai lapisan di atas peta. Sebuah pengaturan
n di mana sel dalam *costmap* ditafsirkan hanya sebagai tiga nilai, bebas,
dan tidak diketahui. Ksaran nilai adalah nilai piksel yang umum (0-255)
255 adalah penghalang, dan 0 adalah bebas. Jika digunakan dalam mode

dengan lebih dari tiga nilai, tempat di mana robot harus mencoba menghindari atau memperlambat, dapat ditambahkan ke peta dengan memiliki nilai tengah. Kemudian, menghitung *costmap* untuk melintasi tempat itu dan menggunakannya dalam perencanaan jalur dan mengikuti perilaku dapat disesuaikan untuk bernegosiasi di sekitar area tersebut. Agar tidak menabrak objek dan dinding, batasan kinematik dan bentuk robot dipertimbangkan dalam peta *costmap*.

2.7 *Motion Control*

Kontrol gerak adalah fase terakhir dalam loop kontrol robot, di mana rencana tingkat tinggi yang dihasilkan pada fase sebelumnya diterjemahkan ke dalam gerakan robot. Oleh karena itu, level ini memproses perintah gerak abstrak dan menghasilkan perintah tingkat rendah untuk mengendalikan kecepatan motor.

Penghindaran rintangan merupakan hal yang menarik, dan dapat diklasifikasikan di bawah kontrol gerak. Ini adalah salah satu masalah utama untuk aplikasi robot bergerak yang sukses, karena memastikan keamanan robot dan entitas di sekitarnya. Strategi penghindaran rintangan berkisar dari algoritme primitif yang hanya menghentikan robot saat rintangan terdeteksi; hingga algoritme kompleks yang memungkinkan robot menghindari rintangan.

Borenstein memperkenalkan algoritma *vector field* histogram (vfh) untuk tugas penghindaran rintangan berdasarkan medan potensial lokal. Dalam pendekatan ini, pertama-tama, data jangkauan diambil sampelnya secara terus menerus, dan kisi-kisi lokal dua dimensi dibuat untuk merepresentasikan lingkungan. Pada tahap berikutnya, histogram polar satu dimensi diekstraksi dari grid lokal dalam bentuk sektor-sektor sudut dengan lebar tertentu. Terakhir, histogram satu dimensi ini diberi ambang batas dan sektor sudut dengan kerapatan tertinggi dipilih sebagai arah. Kecepatan robot juga disesuaikan dalam korelasi dengan jarak dari rintangan. Dalam (Ulrich, et. *al.,* 1998) algoritma ini ditingkatkan dengan memasukkan kinematika robot karena algoritma asli mengasumsikan bahwa robot dapat mengubah arahnya secara instan (dinamai vfh+). Algoritma ini kemudian diperbaiki dan dinamakan sebagai vfh* dalam (Ulrich, et. *al.,* 2000). dengan vfh dan vfh+, yang merupakan algoritme lokal murni berdasarkan an sensor saat ini, vfh* memasukkan algoritma pencarian grafik A* untuk mbangkan lebih dari sekadar lingkungan sekitar robot.

Dalam (Fox *et. al.,* 1997) pendekatan jendela dinamis diperkenalkan sebagai metode penghindaran rintangan. Kendala kinematik dari robot penggerak *Synchro* diperhitungkan dengan mencari ruang kecepatan robot secara langsung. Ruang pencarian selanjutnya direduksi menjadi jendela dinamis, yang berisi kecepatan yang dapat dicapai oleh robot, dengan kecepatan dan akselerasinya. Akhirnya, jendela ini dicari kecepatan yang sesuai dengan arah target robot. Dalam (Brock *et. al.,* 1999) metode ini diadaptasi ke robot *holonomic*, yang memungkinkan penghindaran rintangan berkecepatan tinggi dengan kemampuan manuver yang tinggi.

Terakhir, diagram kedekatan diperkenalkan dalam (Minguez *et. al.,* 2000) yang didasarkan pada aturan heuristik yang disimpulkan dari kemungkinan situasi keamanan tinggi dan rendah yang dapat diakhiri oleh robot. Berdasarkan lima aturan (dua situasi keamanan rendah dan tiga situasi keamanan tinggi), lima perilaku didefinisikan, di mana robot membandingkan situasinya saat ini dengan situasi yang telah ditentukan dan menjalankan perilaku yang sesuai. Hal ini ditunjukkan bahwa pendekatan reaktif ini dapat bekerja dengan baik di lingkungan yang berantakan dengan lorong-lorong yang sempit, dibandingkan dengan pendekatan sebelumnya.

2.8 *Differential Drive Mobile Robot*

Robot dengan penggerak diferensial *(differential drive)* adalah robot yang menggunakan 2 penggerak yang dipasang pada kedua rodanya dan bergerak secara independen. Tipe robot ini dilengkapi dengan 1 atau 2 roda pasif *(caster wheels)* yang bertujuan untuk mempertahankan keseimbangan robot. Setiap roda dapat diatur untuk bergerak dengan kecepatan dan arah putar yang berbeda, sehingga memungkinkan robot untuk dapat bergerak lurus maupun melingkar. Jika robot pergerakan melingkar, robot harus berputar mengilingi suatu titik yang berada sejajar dengan sumbu kedua rodanya, yang disebut sebagai *Instantaneous Center of Curvature* (ICC). Perubahan kecepatan setiap roda akan merubah lintasan robot (Dudek, *et.al.* 2010). Kecepatan sudut robot (terhadap ICC) harus sama untuk kedua

ingga dapat berlaku beberapa persamaan (2) dan (3) sebagai berikut.

$$\left(R + \frac{1}{2}\right) = Vr \qquad (2)$$

$$w\left(R+\frac{1}{2}\right)=Vt \quad (3)$$

Dimana,

i : Jarak antara kedua roda

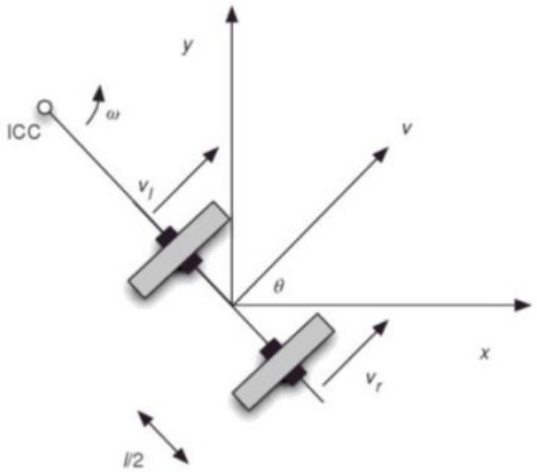
vr : Kecepatan roda kanan terhadap tanah

vt : Kecepatan roda kiri terhadap tanah

R : Jarak titik tengah robot dengan titik pusat rotasi

Nilai kecepatan angular robot terhadap titik pusat rotasi (ω) dapat dicari dengan mengurangkan persamaan (4) dan (5) sehingga didapat persamaan (3) sebagai berikut.

$$w=\frac{Vr-Vl}{l} \quad (4)$$



Gambar 10 Sistem *differential drive robot*

Nilai jarak pusat robot dengan sumbu rotasi (R) dapat dicari dengan menjumlahkan persamaan (1) dan (2) sehingga didapat persamaan (4) sebagai berikut.

$$R=\frac{Vr-Vl}{Vr-vl} \quad (5)$$

Persamaan (4) dan (5) menunjukkan beberapa jenis pergerakan yang dapat dilakukan oleh robot yaitu:

Jika $vr = vl$, robot bergerak lurus karena nilai R tak hingga dan $\omega = 0$ atau tidak

ıtaran.

$-vr$, robot berputar di tempat atau berputar dengan titik pusat perputaran

ıda titik pusat robot ($R = 0$).

Jika $vl = 0$ dan $vr \neq 0$, robot akan berputar berlawanan arah jarum jam dengan pusat rotasi berada pada roda kiri ($R = l2$) dan berlaku sebaliknya.

2.9 Aktuator

Mekanisme perhitungan kecepatan putar dilakukan dengan menghitung jumlah pulsa *rotary encoder* secara periodik. Perputaran motor menghasilkan pulsa *encoder* yang selanjutnya dideteksi oleh kontroler melalui mekanisme *hardware interrupt*. Hasil percobaan menunjukkan *rotary encoder* yang digunakan, seperti yang terlihat pada Gambar 10, mempunyai jumlah 339 pulsa setiap putaran.

Gambar 11 Contoh komponen aktuator

Sehingga kecepatan roda dapat dihitung dengan persamaan (5) sebagai berikut.

$$rpm = \frac{(Jumlah\ Pulsa/339)}{(Time\ sampling \times 60)} \quad (6)$$

Selanjutnya akan dilakukan pembacaan dan kalibrasi sensor kecepatan pada roda kanan dan roda kiri. Pengamatan dan pencatatan dilakukan dengan menggunakan serial monitor yang dikirimkan oleh kontroler. Kalibrasi kecepatan dilakukan dengan membandingkan hasil pembacaan sensor dengan rpm meter. Pengukuran kecepatan dilakukan pada tegangan baterai 28,8 volt (tegangan maksimum 29,4 volt) sebelum masuk ke *buck converter* dan motor berputar pada arah yang menyebabkan robot bergerak maju.

2.10 Kinematika dan *Trajectory Tracking*

ancangan sistem kontrol kinematik dan *trajectory tracking* yang entasikan pada robot menggunakan 2 algoritma kinematik, yaitu *inverse* kinematik untuk kontrol kinematik gerak robot dan *forward velocity*

kinematik yang digunakan untuk sistem *trajectory tracking* (Kim dan Yi, 2014). Pada dasarnya, persamaan kinematik dapat dilihat pada (1). Di mana x adalah vektor pose yang berisi koordinat x koordinat x, koordinat y, dan arah hadap (θ) dari robot pada lintasan di mana robot bergerak (Sorour, dkk. 2017).

$$x = f(q) \tag{7}$$

Persamaan (7) merupakan persamaan dalam dimensi posisi posisi, dimana pengendalian robot dengan kontrol posisi sulit untuk dilakukan (Morales, et.al., 2021). Sehingga persamaan (7) dapat diturunkan terhadap waktu untuk mendapatkan persamaan dalam dimensi kecepatan dalam dimensi kecepatan seperti yang tertulis di bawah ini.

$$\frac{dx}{dt} = \frac{df(q)}{dt} \tag{8}$$

$$\dot{x} = \frac{df(q)}{dq} \times \frac{dq}{dt} \tag{9}$$

$$\dot{\boldsymbol{x}} = \boldsymbol{J}.\dot{\boldsymbol{q}} \tag{10}$$

Dimana **J** adalah matriks Jacobian, dimana matriks tersebut berisi model matematis yang berkorelasi dengan spesifikasi mekanik robot dengan desain kontrol yang dibuat. $\dot{\mathbf{q}}$ adalah vektor yang berisi kecepatan aktuator yang dimiliki oleh robot. Dimana r adalah jari-jari roda robot, L adalah jarak antara titik tengah roda kiri dan roda kanan robot, dan sudut θ adalah sudut dari arah robot ke lintasan.