Implementasi Robot Sorting Barang dengan Python untuk Machine Learning dan Arduino untuk Sistem Kendali

Raphael Nazareth[1], Samuel Jordan Widjaja[1], Willsan A Jantho[1], Efran Leonard Putra Satria[1], Alifito Marciano Camil[1], Dr. Jemie Muliadi, S.T., M.T.[2]

[1]*Binus ASO School of Engineering, Jl. Alam Sutera Boulevard No.1, Alam Sutera, 15325, Indonesia*

[2]*Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta, 11480, Indonesia*

Abstract

Perkembangan teknologi di era modern telah membawa dampak signifikan pada sektor industri, termasuk dalam pengelolaan gudang. Salah satu tantangan utama yang dihadapi dalam operasional gudang adalah kecelakaan kerja akibat sistem penanganan barang yang tidak efisien, serta kesalahan penempatan barang akibat kelalaian pekerja. Selain itu, efisiensi tata letak gudang menjadi isu yang mendesak, mengingat banyak gudang hanya mampu memanfaatkan sebagian kecil dari kapasitas ruang yang tersedia. Untuk menjawab tantangan tersebut, laporan ini membahas penelitian terkait penerapan teknologi robot *sorting* berbasis *machine learning* yang terintegrasi dengan sistem kamera untuk melakukan penataan barang secara optimal. Penelitian ini berfokus pada pengembangan dan analisis kinematika serta dinamika lengan robot yang dikendalikan melalui simulasi menggunakan MATLAB, Simulink, dan Simscape. Sistem *machine learning* diterapkan dengan memanfaatkan algoritma OpenCV untuk identifikasi dimensi barang dan algoritma RectPack untuk optimasi tata letak. Hasil dari penelitian ini diharapkan dapat meningkatkan efisiensi ruang, mempercepat proses penyortiran, serta mengurangi risiko kecelakaan kerja di gudang.

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

Di era teknologi yang semakin canggih seperti saat ini, perkembangan teknologi memberikan dampak besar pada sektor industri. Salah satu dampaknya adalah peningkatan kecepatan dan volume produksi di berbagai industri. Dengan meningkatnya produksi, kebutuhan akan gudang untuk menyimpan hasil produksi juga meningkat. Namun, banyak permasalahan sering muncul di dalam operasional gudang. Salah satu permasalahan utama adalah kecelakaan yang terjadi selama proses penyortiran barang. Hal ini sering kali disebabkan oleh sistem penanganan barang berat yang tidak tepat, yang dapat membahayakan keselamatan pekerja. Selain itu, banyak gudang masih mengandalkan tenaga kerja manusia, yang berisiko menimbulkan kesalahan penempatan barang akibat human error, serta membuat proses penyortiran menjadi lebih lambat [1].

Masalah lainnya adalah tata letak gudang yang tidak efisien. Berdasarkan survei pergudangan pada tahun 2018, sebagian besar gudang hanya memanfaatkan sekitar 68% dari kapasitas ruang yang tersedia. Bahkan, hanya 15% gudang yang mampu memanfaatkan 100% kapasitasnya. Ketidakefisienan ini tidak hanya berdampak pada optimalisasi ruang, tetapi juga memperumit manajemen logistik di gudang. Oleh karena itu, diperlukan solusi inovatif untuk mengatasi permasalahan ini, salah satunya adalah dengan menerapkan teknologi modern seperti robot sorting untuk meningkatkan efisiensi ruang, mempercepat proses, dan meningkatkan keselamatan kerja [2].

1.2 Manfaat dan Tujuan Penelitian

Penelitian ini bertujuan untuk mengembangkan sebuah robot yang dilengkapi dengan *microcontroller* yang mampu mengintegrasikan kinematika dan dinamika dalam pergerakannya, sehingga dapat menjalankan tugas penyortiran barang secara presisi dan efisien. Selain itu, sistem robot ini dirancang untuk memanfaatkan teknologi machine learning dalam bentuk *computer vision*, yang memungkinkan robot untuk mengidentifikasi, mengukur, dan mengklasifikasikan benda secara otomatis berdasarkan data visual. Dengan menggabungkan teknologi ini, sistem mampu mengoptimalkan proses penataan barang di gudang secara *real-time*.

Manfaat yang diharapkan dari penelitian ini mencakup tiga aspek utama. Pertama, peningkatan efisiensi penggunaan ruang di dalam gudang, di mana tata letak barang dapat diatur secara optimal

untuk memaksimalkan kapasitas penyimpanan. Kedua, terciptanya lingkungan kerja yang lebih aman dan terkendali, karena proses penanganan barang berat dan berisiko dapat dialihkan dari pekerja manusia ke robot, sehingga mengurangi potensi kecelakaan kerja. Ketiga, munculnya peluang pekerjaan baru yang berfokus pada perawatan dan pengoperasian perangkat berbasis teknologi ini, yang sejalan dengan kebutuhan tenaga kerja di era industri 4.0.

Adapun inovasi dan aplikasi utama dari penelitian ini meliputi pemanfaatan teknologi microcontroller dan machine learning untuk meningkatkan efisiensi dalam sektor pergudangan. Robot yang dikembangkan akan terintegrasi dengan perangkat lunak berbasis computer vision, memungkinkan proses identifikasi dan penyortiran barang dilakukan secara otomatis dan lebih cepat dibandingkan metode konvensional. Dengan penerapan teknologi ini, industri pergudangan tidak hanya dapat meningkatkan produktivitas dan efisiensi, tetapi juga mengadopsi solusi teknologi canggih untuk menghadapi tantangan masa depan.

1.3 Batasan Masalah

Batasan masalah dalam proyek ini dirumuskan sebagai berikut. Barang yang diangkut berbentuk silinder, sehingga orientasi barang tidak menjadi kendala dan tidak memerlukan rotasi. Akurasi dalam pengangkutan barang memiliki toleransi sebesar 2–3 cm, yang disebabkan oleh adanya offset pada end-effector yang tidak sejajar dengan sumbu putar dan desain dari robot memiliki beberapa kekurangan yang dapat dikembangkan. Selain itu, barang yang diangkat tidak boleh berwarna hitam, karena sistem machine learning mendeteksi perbedaan warna dalam skala grayscale dan mengasumsikan alas sebagai warna hitam. Oleh karena itu, warna barang harus lebih terang dibandingkan dengan warna alas. Sebagai tambahan, barang harus dilengkapi dengan besi pada bagian atasnya agar dapat diangkat menggunakan magnet.

BAB II MATERI DAN METODE

2.1 Kinematika

Kinematika adalah studi tentang hubungan antara koordinat sendi robot dan posisi spasialnya, serta merupakan topik fundamental dalam robotika. Untuk mendapatkan posisi yang akurat, kinematika dapat diselesaikan melalui metode numerik maupun analitis. Misalnya, *forward* kinematika memungkinkan kita menentukan posisi *end-effector* di suatu titik di ruang, merancang mekanisme yang menggerakkan alat dari titik A ke titik B, atau memprediksi apakah gerakan robot akan bertabrakan dengan rintangan. Sementara itu, kinematika terbalik atau *Inverse kinematic* membantu menghitung nilai sudut setiap sendi yang dibutuhkan untuk mencapai posisi tertentu. Kinematika hanya memperhitungkan kondisi sesaat koordinat robot, tanpa mempertimbangkan gerakan yang dipengaruhi oleh gaya atau torsi, yang akan dibahas lebih lanjut dalam dinamika. Masalah kinematika mungkin terlihat sederhana untuk robot tertentu, seperti robot bergerak yang dianggap benda kaku, namun membutuhkan kajian lebih mendalam pada robot dengan banyak sendi, seperti robot humanoid atau mekanisme parallel.

2.2 Kinematika Terbalik (*Inverse Kinematics*)

Dalam dunia robotika, inverse kinematics atau kinematika terbalik adalah salah satu konsep utama yang digunakan untuk menentukan variabel-variabel pada sambungan robot berdasarkan posisi dan orientasi yang diinginkan dari efektor akhir (*end-effector*). Kinematika terbalik bertujuan untuk menyelesaikan masalah yang berkebalikan dengan *forward kinematics*, di mana *forward kinematics* menghitung posisi dan orientasi dari efektor akhir berdasarkan variabel sambungan yang diketahui. Pada kinematika terbalik, input utama adalah posisi dan orientasi target dari efektor akhir, sementara outputnya adalah nilai-nilai variabel pada sambungan, seperti posisi, sudut rotasi, atau panjang aktuator yang diperlukan untuk mencapai target tersebut. Proses ini melibatkan solusi persamaan nonlinear yang kompleks, tergantung pada konfigurasi mekanisme robot dan derajat kebebasannya (*Degrees of Freedom*, DoFs). Salah satu aplikasi penting dari inverse kinematics adalah pada kontrol dan pergerakan lengan robot dengan tiga sambungan (*three-link arm robot*), yang sering digunakan dalam aplikasi penyortiran barang di gudang atau pengambilan objek pada posisi tertentu. *Three-link arm robot* memiliki tiga DoFs yang biasanya berupa rotasi di masing-masing sambungan. Untuk mencapai posisi yang diinginkan, seperti menjangkau suatu objek pada koordinat

tertentu, kinematika terbalik digunakan untuk menghitung sudut rotasi yang diperlukan pada setiap sambungan.

Sebagai contoh, jika efektor akhir diminta untuk mencapai suatu titik dengan koordinat tertentu pada bidang dua dimensi (x, y), kinematika terbalik akan menghitung sudut-sudut pada masing-masing sambungan secara dinamis. Dengan kata lain, kinematika invers menerjemahkan koordinat target menjadi perintah spesifik berupa sudut rotasi yang harus dicapai oleh masing-masing sambungan. Pada robot ini, pendekatan matematika yang digunakan melibatkan trigonometri untuk memecahkan persamaan posisi dari setiap sambungan, dengan mempertimbangkan panjang masing-masing segmen lengan dan koordinat target. Dalam penelitian ini, kinematika terbalik digunakan untuk mengatur pergerakan lengan robot yang melakukan proses penyortiran barang di gudang. Hal ini memastikan efektor akhir dapat mencapai posisi yang diinginkan dengan akurasi tinggi sesuai kebutuhan. Implementasi kinematika terbalik pada three-link arm robot memberikan solusi yang efisien untuk berbagai aplikasi robotika, mulai dari pengelolaan inventaris hingga proses manufaktur presisi [3].

2.3 Dynamics

Dinamika robotik adalah bidang studi yang mempelajari perilaku gerak robot sebagai respons terhadap gaya atau torsi yang diberikan. Dalam robotik, dinamika memainkan peran penting dalam menjelaskan hubungan antara kontrol (seperti gaya atau torsi pada aktuator) dan perubahan keadaan robot, termasuk posisi, kecepatan, dan percepatan. Dengan memahami dinamika, kita dapat merancang kontrol yang memungkinkan robot bergerak secara presisi sesuai dengan tugas yang diinginkan, meskipun terdapat gangguan atau interaksi dengan lingkungannya. Dinamika robotik juga memungkinkan prediksi perilaku robot berdasarkan model matematisnya, sehingga sistem dapat dioptimalkan untuk stabilitas dan efisiensi energi.

Secara umum, model dinamika robot dapat dijelaskan melalui persamaan:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = u,$$

di mana qq adalah posisi sambungan (joint), q˙q˙ adalah kecepatan, dan q¨q¨ adalah percepatan. Matriks inersia M(q)M(q)menggambarkan distribusi massa robot dan sifat dinamisnya, sedangkan C(q,q˙)C(q,q˙) mewakili gaya sentripetal dan Coriolis yang muncul akibat gerakan. Komponen G(q)G(q) melambangkan gaya gravitasi yang bekerja pada robot, dan uu adalah gaya atau torsi kontrol yang diberikan pada aktuator. Persamaan ini mencakup semua interaksi dinamis dalam robot, termasuk efek kinematika, inersia, gravitasi, dan gaya-gaya lain yang bekerja pada sistem.

Dalam implementasi kontrol robot, pemahaman terhadap dinamika sangat penting. Misalnya, pada sistem kontrol berbasis lintasan, dinamika digunakan untuk menghitung percepatan yang diinginkan berdasarkan lintasan referensi (qrefqref) yang diinginkan. Selain itu, teknik kontrol seperti PID (Proportional-Integral-Derivative) atau kontrol ruang operasional memanfaatkan model dinamika ini untuk mengatur gaya atau torsi sehingga robot dapat mengikuti lintasan dengan presisi tinggi. Dengan pendekatan ini, robot dapat menangani interaksi yang kompleks, seperti perubahan beban atau pengaruh lingkungan, dan tetap menjaga kinerja yang optimal [4].

2.4 Three Link Planar Robot

Three link planar robot merupakan suatu jenis lengan robot yang memiliki tiga titik sendi, sehingga robot memiliki sebanyak tiga derajat kebebasan. Dalam membuat sebuah *three link planar* robot terdapat beberapa aspek yang perlu dipahami terlebih dahulu. Yang pertama adalah panjang dari setiap bagian dari lengan pada sebuah robot atau dapat dikatakan jarak antar dua sendi pada robot yang terdiri dari (I1, I2, I3), dan yang kedua adalah sudut yang dibentuk antara dua sendi pada lengan robot yang terdiri dari (θ_1, θ_2, θ_3). Sudut pertama pada lengan robot (θ_1) diukur dari sumbu x yang bersifat tetap, kemudian sendi-sendi selanjutnya (θ_2, θ_3) diukur berdasarkan orientasi pada sendi yang sebelumnya. Salah satu bagian paling penting dari sebuah lengan robot yaitu end effector merupakan bagian dari robot yang berinteraksi dengan lingkungan dan memberikan kemampuan bagi robot untuk menjalankan tugasnya. Dalam sebuah end effector, untuk mengetahui lokasi dari lengannya x,y, dan φ (phi), perlu didefinisikan terlebih dahulu. x dan y merupakan koordinat yang memberikan informasi mengenai posisi dari end effector. Sedangkan φ (phi) merupakan sudut yang memberikan gambaran mengenai posisi dan orientasi end effector [5].

2.5 Jacobian

Matriks Jacobian adalah konsep fundamental dalam robotika yang menggambarkan hubungan matematis antara kecepatan sendi robot manipulator dan kecepatan end-effector. Matriks ini, yang dilambangkan sebagai J(q), bergantung pada konfigurasi atau posisi robot saat ini. Setiap kolom dalam matriks Jacobian mencerminkan kontribusi kecepatan dari suatu sendi terhadap kecepatan linier dan sudut dari end-effector. Secara umum, matriks Jacobian terdiri dari dua komponen utama: Jv, yang berkaitan dengan kecepatan linier end-effector, dan Jw, yang berhubungan dengan kecepatan sudutnya. Untuk menghitung Jv, kinematika maju (forward kinematics) digunakan untuk memodelkan fungsi posisi end-effector (x,y,z) terhadap variabel sendi. Di sisi lain, Jw diperoleh dengan mempertimbangkan kecepatan sudut, yang dihitung sebagai hasil kali silang antara vektor sumbu rotasi dengan laju rotasi dari masing-masing sendi. Kombinasi Jv dan Jw menghasilkan matriks Jacobian lengkap yang memiliki peran penting dalam analisis kecepatan, kontrol gerak, serta penyelesaian kinematika balik (inverse kinematics). Matriks ini memungkinkan analisis hubungan antara pergerakan sendi robotik dan gerakan end-effector, menjadikannya alat esensial dalam pengendalian dan pemrograman robot, khususnya dalam memastikan akurasi dan efisiensi pergerakan robot [6].

2.6 Open CV

Open CV (Open Source Computer Vision Library) adalah pustaka sumber terbuka yang dirancang khusus untuk aplikasi visi komputer dan pemrosesan gambar secara real-time. Open CV banyak digunakan dalam berbagai bidang, seperti kecerdasan buatan, robotika, dan augmented reality, berkat efisiensinya dan kemampuannya yang serbaguna. OpenCV mendukung berbagai bahasa pemrograman, seperti Python, C++, Java, dan MATLAB, sehingga memudahkan pengembang dari berbagai platform untuk memanfaatkannya. Selain itu, OpenCV kompatibel dengan sistem operasi utama seperti Windows, macOS, dan Linux.

Prinsip kerja OpenCV dalam pemrosesan gambar didasarkan pada cara komputer memperlakukan gambar sebagai kumpulan piksel, di mana setiap piksel direpresentasikan dengan nilai numerik yang sesuai dengan intensitasnya (untuk gambar grayscale) atau saluran warnanya (seperti RGB atau format lainnya). Representasi numerik ini memungkinkan komputer untuk memproses dan menganalisis gambar secara matematis. OpenCV menyederhanakan tugas pemrosesan gambar yang kompleks dengan menyediakan fungsi-fungsi bawaan untuk operasi seperti penyaringan gambar, deteksi tepi, ekstraksi fitur, dan transformasi geometris.

Sebagai contoh, dalam proses deteksi tepi, OpenCV menggunakan operasi matematis seperti perhitungan gradien untuk mengidentifikasi area dalam gambar yang memiliki perubahan intensitas signifikan, yang biasanya menunjukkan batas objek. Sementara itu, dalam penyaringan gambar, kernel atau masker konvolusi diterapkan untuk memodifikasi nilai piksel secara sistematis, memungkinkan tugas seperti pengurangan noise atau penajaman gambar. Desain modular OpenCV dan algoritma yang dioptimalkan membuatnya menjadi alat yang sangat kuat untuk mengimplementasikan proses ini dengan efisien, bahkan dalam aplikasi real-time [7].

2.7 Rectangle Packing

Algoritma *rectangle packing* sangat penting dalam grafik komputer dan pengembangan game untuk menyusun berbagai persegi panjang secara efisien dalam sebuah persegi panjang yang lebih besar tanpa saling tumpang tindih. Proses ini sangat penting untuk tugas-tugas seperti menyusun bitmap font ke dalam satu tekstur untuk mengoptimalkan kinerja GPU. Tantangan utama dalam masalah ini adalah sifatnya yang bersifat *NP-hard*, yang berarti tidak ada solusi sempurna, sehingga diperlukan pendekatan heuristik untuk mencapai hasil yang memadai.

Salah satu algoritma yang terkenal dalam bidang ini adalah metode *Skyline Bottom-Left* (Skyline BL), yang diimplementasikan dalam pustaka stb_rect_pack. Algoritma ini menggunakan struktur data yang merepresentasikan "skyline" dari persegi panjang yang sudah disusun, yaitu posisi tepi atas tertinggi di mana persegi panjang baru dapat ditempatkan. Ketika menempatkan persegi panjang baru, algoritma ini mencari posisi paling kiri pada *skyline* di mana persegi panjang tersebut dapat muat, dengan memastikan penempatannya serendah mungkin. Strategi ini bertujuan untuk meminimalkan tinggi keseluruhan dari susunan persegi panjang dan mengurangi ruang yang terbuang.

Algoritma Skyline BL bekerja dengan iterasi melalui node dalam *linked list* yang merepresentasikan *skyline*, mengevaluasi potensi posisi untuk persegi panjang baru. Setelah menemukan tempat yang sesuai, algoritma ini memperbarui skyline untuk mencerminkan penambahan tersebut, baik dengan menghapus maupun menyesuaikan node yang tumpang tindih dengan persegi panjang baru. Metode ini menawarkan keseimbangan antara efisiensi dan kesederhanaan, sehingga menjadi pilihan praktis untuk banyak aplikasi meskipun kompleksitas masalah *rectangle packing* cukup tinggi [8].

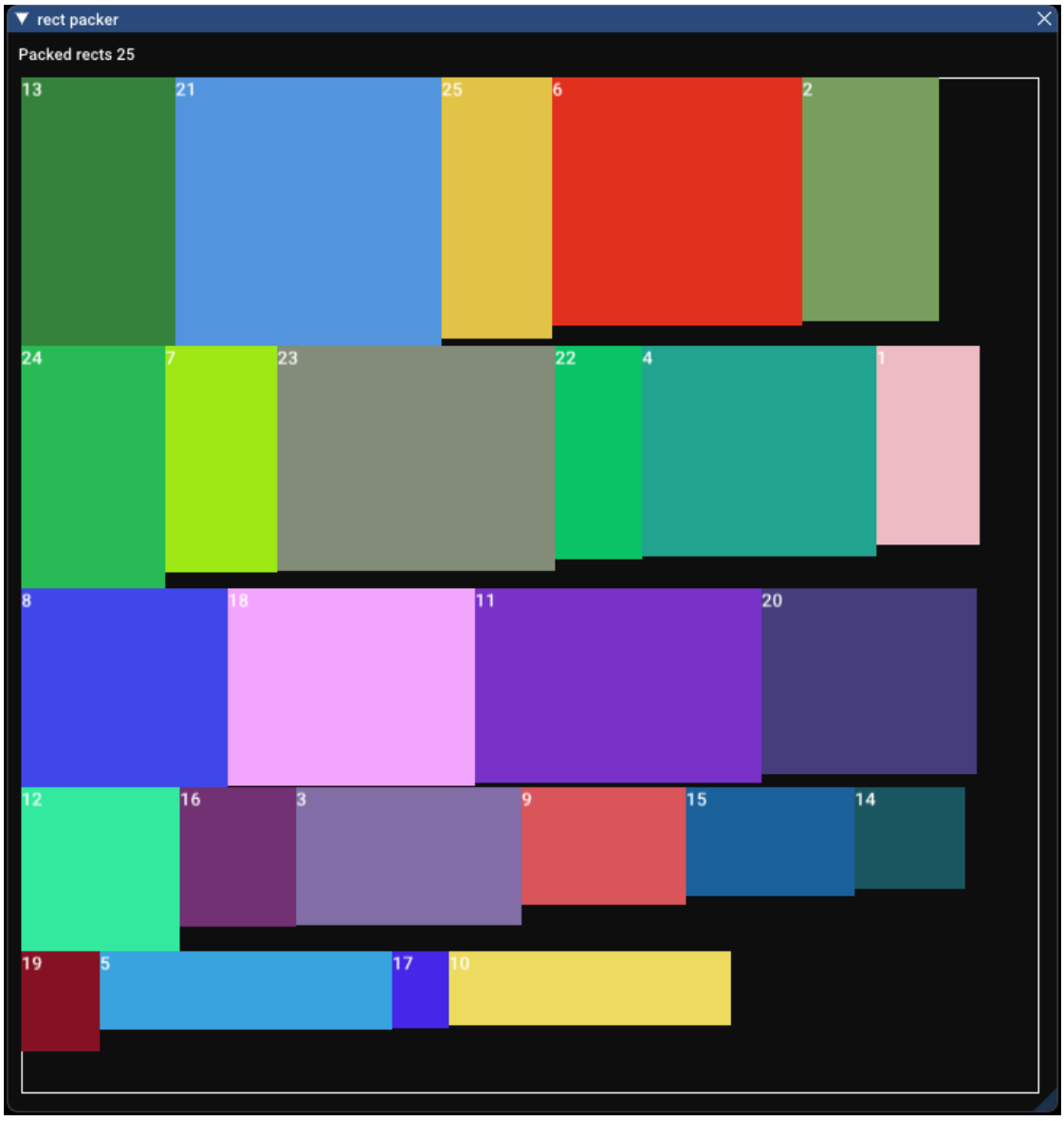


Figure 1. Rectpack

2.8 PID

Kontrol Proportional-Integral-Derivative (PID) adalah salah satu metode sistem kontrol yang paling banyak digunakan dalam otomasi dan pengendalian proses. Kontroler PID digunakan untuk terus menerus menghitung nilai *error*, yaitu selisih antara titik yang diinginkan (*setpoint*) dan variabel proses yang terukur. Berdasarkan kesalahan ini, kontroler menerapkan tindakan korektif untuk menjaga keluaran sistem sedekat mungkin dengan titik setel yang diinginkan. Pendekatan ini sangat cocok untuk mengatur proses fisik seperti suhu, kecepatan, tekanan, dan tegangan.

Kontroler PID terdiri dari tiga komponen: ***proportional* (P)**, ***integral* (I)**, dan ***derivative* (D)**, yang masing-masing memberikan kontribusi pada keluaran kontrol. Komponen *proportional* memberikan respons langsung terhadap kesalahan saat ini dengan menerapkan koreksi yang sebanding dengan besarnya kesalahan. Namun, jika hanya bergantung pada komponen *proportional*, sistem dapat mengalami kesalahan keadaan tunak (*steady-state error*), di mana keluaran sistem tetap sedikit berbeda dari titik setel yang diinginkan.

Komponen *integral* mengatasi masalah ini dengan mengakumulasi *error* di masa lalu seiring waktu dan menerapkan koreksi untuk menghilangkan SSE. Komponen ini mengintegrasikan sinyal *error*, memastikan bahwa setiap deviasi yang terus-menerus dari titik setel diperbaiki dari waktu ke waktu. Namun, komponen integral dapat menyebabkan osilasi dan *overshoot* jika tidak disetel dengan benar.

Di sisi lain, komponen *derivative* memprediksi perilaku sistem di masa depan dengan menghitung laju perubahan kesalahan. Kemampuan prediktif ini membantu kontroler meredam osilasi dan meningkatkan stabilitas sistem, terutama pada proses yang mengalami fluktuasi cepat. Dengan merespons laju perubahan kesalahan, komponen derivative secara efektif memperhalus aksi kontrol.

Dengan mengkombinasikan ketiga komponen ini, kontroler PID menawarkan strategi kontrol yang seimbang, meminimalkan kesalahan, mengurangi osilasi, dan meningkatkan stabilitas sistem. Kontribusi relatif dari komponen P, I, dan D ditentukan dengan menyetel parameter masing-masing, yang sering disebut nilai gain. Penyelarasan parameter ini sangat penting untuk mencapai kinerja optimal untuk aplikasi tertentu, karena penyetelan yang tidak tepat dapat menyebabkan ketidakstabilan atau kontrol yang tidak optimal [9].

2.9 ROS

ROS adalah kerangka kerja open-source yang dirancang untuk mengembangkan dan mengendalikan robot. ROS menyediakan kumpulan alat, pustaka, dan driver untuk membuat aplikasi robotik yang kompleks. Dengan sistem pesan berbasis publish-subscribe, ROS memungkinkan komunikasi antara berbagai komponen robot, sehingga mempermudah integrasi sensor, aktuator, dan algoritma kontrol. ROS mendukung pemrograman multi-bahasa dan desain modular, menjadikannya fleksibel untuk berbagai platform robot. Ekosistemnya yang luas dengan paket-paket bawaan mempercepat pengembangan tugas seperti persepsi, navigasi, dan perencanaan gerak [10].

2.10 Moveit

MoveIt adalah kerangka kerja perencanaan gerak yang kuat yang dibangun di atas ROS dan banyak digunakan untuk lengan robotik dan manipulator. MoveIt menawarkan alat untuk inverse kinematics, perencanaan jalur, deteksi tabrakan, dan optimasi lintasan. MoveIt memungkinkan robot untuk melakukan pergerakan yang presisi dan aman di lingkungan yang kompleks. MoveIt terintegrasi dengan baik dengan perangkat keras maupun lingkungan simulasi, menyediakan visualisasi di RViz dan interaksi dengan pengontrol. Fleksibilitasnya memungkinkan pengembang untuk menyesuaikan algoritma dan membuat aplikasi untuk tugas seperti pick-and-place, manipulasi objek, dan pergerakan otonom [11].

2.11 Rviz

RViz adalah alat visualisasi di ROS yang memungkinkan pengguna untuk mengamati dan memeriksa sistem robot dalam lingkungan virtual. RViz dapat menampilkan data sensor, model robot, lintasan, dan peta secara real-time, sehingga menjadi alat penting untuk memahami bagaimana robot merasakan dan berinteraksi dengan lingkungannya. Pengembang menggunakan RViz untuk menguji algoritma, memverifikasi keadaan robot, dan memantau pergerakan tanpa memerlukan perangkat keras fisik. Dengan arsitektur berbasis plugin, RViz memungkinkan kustomisasi untuk kebutuhan tertentu, seperti menampilkan visualisasi 3D dari jalur robot atau titik awan dari sensor [12].

2.12 Gazebo

Gazebo adalah lingkungan simulasi berbasis fisika untuk menguji dan membuat prototipe sistem robotik. Gazebo menyediakan pemodelan realistis untuk robot, sensor, dan lingkungan, lengkap dengan gravitasi, gesekan, dan properti fisik lainnya. Gazebo memungkinkan pengembang untuk memvalidasi algoritma seperti perencanaan gerak atau navigasi tanpa memerlukan robot fisik. Simulasi ini terintegrasi dengan baik dengan ROS, memungkinkan pertukaran data antara simulasi dan node ROS secara mulus. Dengan menyediakan platform yang aman dan hemat biaya, Gazebo sangat cocok untuk pengembangan cepat, pengujian, dan pelatihan model pembelajaran mesin untuk aplikasi robotik [13].

2.13 Alat & Bahan

2.12.1 Alat

- Servo 360
- Servo 180
- Komponen 3D print
- Rotary encoder
- Elektromagnet
- Limit switch
- Power supply

2.12.2 Bahan

- SolidWorks, untuk melakukan pemodelan robot
- Matlab, simulink, simscape, untuk membuat simulasi robot dan tuning PID
- Visual Studio Code, untuk menjalankan machine learning

BAB III PROSEDUR

Hal pertama yang dilakukan dalam perancangan robot sorting adalah menentukan end effector yang akan digunakan. Pilihan end effector yang dipertimbangkan mencakup *gripper* dan *suction*. Namun, pilihan akhir jatuh pada *end effector* berupa elektromagnet karena dinilai lebih efisien dan praktis baik dari segi pembuatan maupun penggunaan. Setelah *end effector* ditentukan, langkah berikutnya adalah memilih algoritma *machine learning* yang sesuai. Algoritma yang dipilih meliputi OpenCV untuk *image processing* dan pengukuran dimensi benda, serta algoritma Rectangle Packing untuk mengatur tata letak barang agar lebih efisien.

Selanjutnya, pemilihan komponen robot dilakukan dengan memperhatikan kinerja dan performa sistem secara keseluruhan. Komponen utama yang digunakan mencakup servo motor, lengan robot, dan sensor. Servo motor terdiri dari dua jenis: servo 180° (positional) untuk pergerakan terbatas dengan sudut rotasi tertentu, dan servo 360° (continuous) yang digunakan pada basis robot untuk memungkinkan rotasi penuh. Servo 360° ini dilengkapi dengan rotary encoder yang berfungsi untuk membaca rotasi dengan presisi yang cukup tinggi sehingga dapat mencapai sudut yang diinginkan.

Lengan robot dirancang agar memiliki bobot seringan mungkin untuk menghindari beban berlebih pada servo, memastikan servo dapat beroperasi dengan optimal. Perancangan lengan juga mempertimbangkan pusat massa dan daya tahan material untuk menjaga stabilitas dan keandalan sistem. Sensor rotary encoder diintegrasikan pada servo base untuk memberikan kontrol yang lebih akurat terhadap sudut rotasi.

Dengan langkah-langkah ini, perancangan robot dilakukan secara sistematis sehingga setiap komponen dan algoritma dapat bekerja secara harmonis untuk mendukung efisiensi dan performa robot yang maksimal.

Robot ini memanfaatkan teknologi 3D design dalam perancangannya sehingga dapat menciptakan sebuah struktur robot yang kokoh. Part untuk robot ini dirancang menggunakan software SolidWorks dan setelah banyak revisi rancangan yang dilakukan hasil akhir bentuk dari robot ini seperti pada gambar di bawah ini.

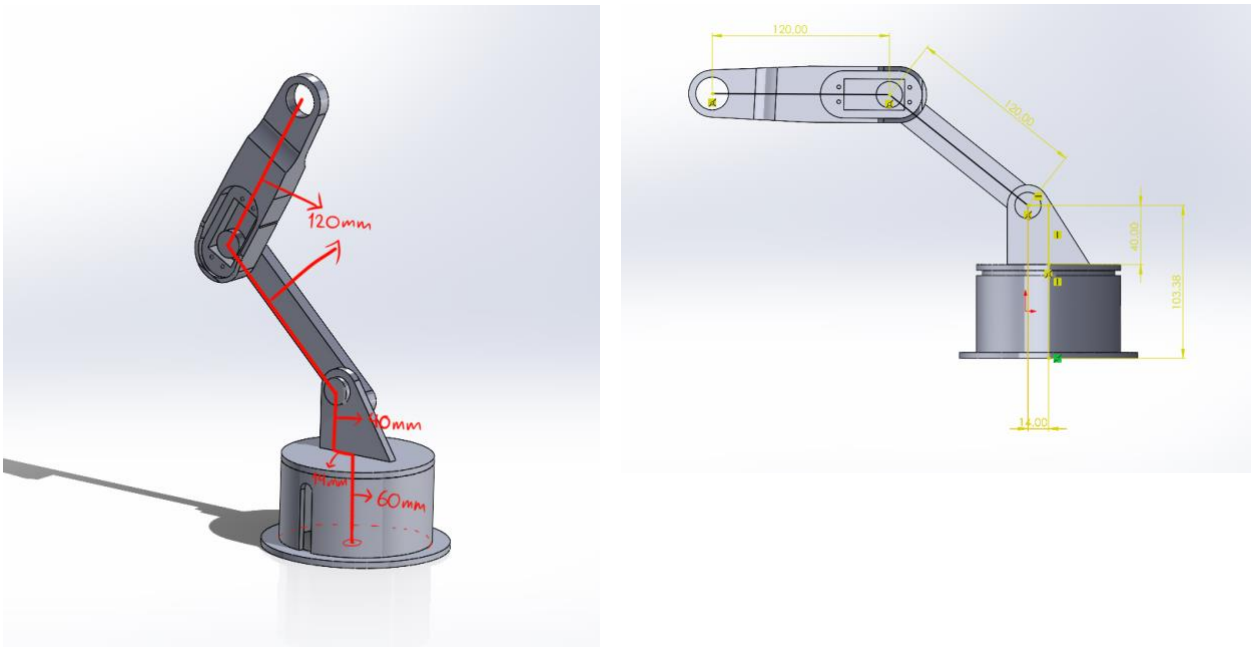


Figure 2. Robot Dimensions & Design

Berdasarkan gambar di atas, penempatan servo yang digunakan dalam robot ini adalah pada kedua sendi robot yang merupakan lengan dari robot tersebut yang menggunakan servo 180 dan pada alas dari robot yang menggunakan servo 360 yang bertujuan untuk mengatur arah robot tersebut menghadap untuk saat mengambil barang.

Untuk mengidentifikasi koordinat dan ukuran benda berbentuk silinder yang akan diambil, sebuah kamera digunakan dan diletakkan tepat di atas area penyimpanan awal benda. Dengan menggunakan kamera tersebut, algoritma computer vision yang didukung oleh OpenCV diterapkan. OpenCV memungkinkan identifikasi benda berdasarkan fitur visual seperti bentuk melingkar, diameter, dan posisi koordinat. Setelah koordinat dan ukuran benda berhasil dideteksi, algoritma

Rectpack dapat disesuaikan untuk menyusun tata letak benda-benda berbentuk lingkaran secara efisien dalam suatu area terbatas. Rectpack, meskipun umumnya dirancang untuk objek berbentuk persegi panjang, dapat dimodifikasi atau digabungkan dengan pendekatan lain untuk menangani benda berbentuk bulat. Kombinasi algoritma ini memungkinkan sistem untuk mengidentifikasi, mengatur, dan memanfaatkan ruang penyimpanan dengan lebih optimal sesuai dengan bentuk benda.

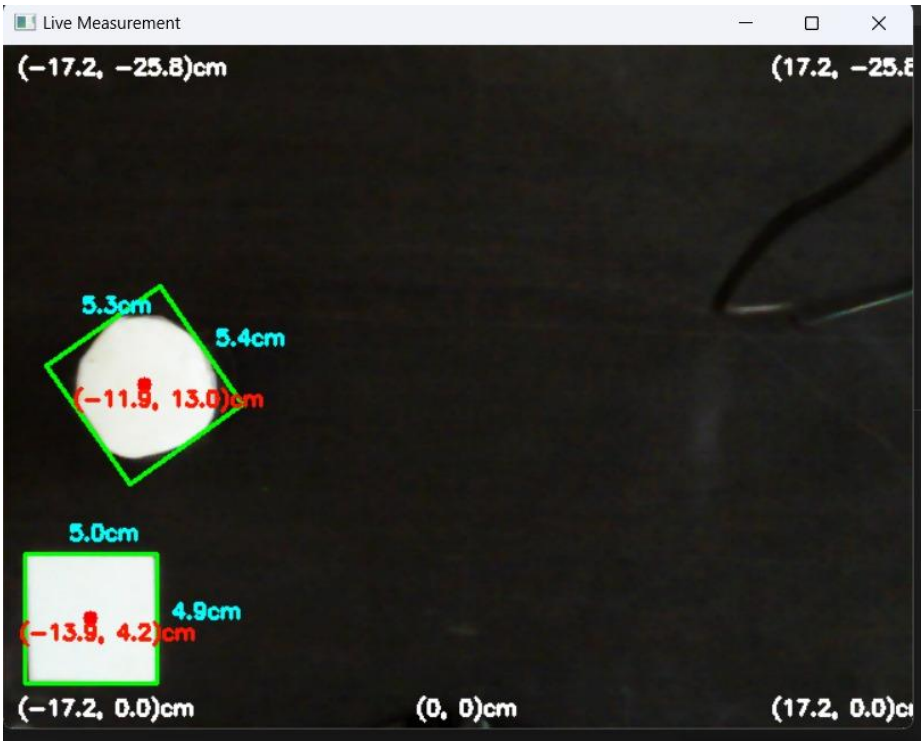


Figure 3. Live Camera Measurement (OpenCV)

Pada gambar di atas, kita dapat melihat cara kerja OpenCV, pertama, OpenCV memanfaatkan kemampuan *edge detection*, untuk mengetahui batas-batas objek yang terdeteksi dan memberikan ukuran ukuran. Kemudian kita menggunakan kemampuan pemetaan koordinat/piksel untuk mendeteksi lokasi/koordinat objek silinder. Kubus yang terlihat di kiri bawah gambar diatas berfungsi sebagai titik referensi yang diatur secara manual, untuk memastikan bahwa untuk memastikan skala pengukuran sudah akurat.

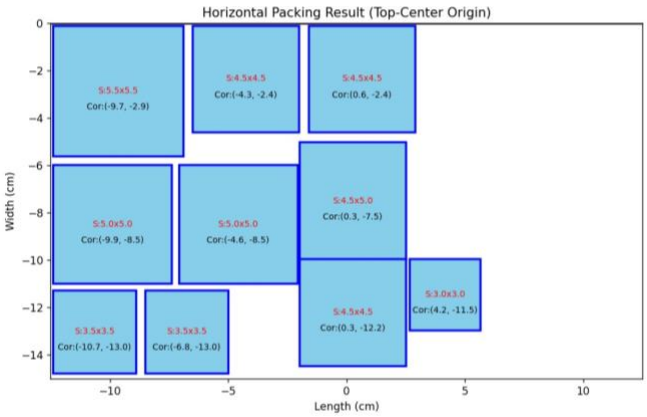


Figure 4. Rectpack result

Setelah deteksi ukuran dan posisi objek silinder dilakukan oleh OpenCV, variabel-variabel tersebut kemudian dimasukkan ke dalam algoritma *2D-bin packing* yang dikenal sebagai *rectpack*. Algoritma packing ini kemudian memproses silinder, yang telah ditafsirkan ulang sebagai kotak dan atau persegi panjang, dan menghasilkan cara pengemasan yang optimal berdasarkan perhitungannya. Hasil dari algoritma ini kemudian ditampilkan secara visual di layar kepada *user* melalui penggunaan *library* matplotlib. Memungkinkan pengguna untuk melihat koordinat silinder setelah di packing.

Sebelum dilakukan pengerjaan yang lebih lanjut untuk robot, dilakukan simulasi gerak robot yang dilakukan dalam matlab, simulink, simscape, dan ROS.

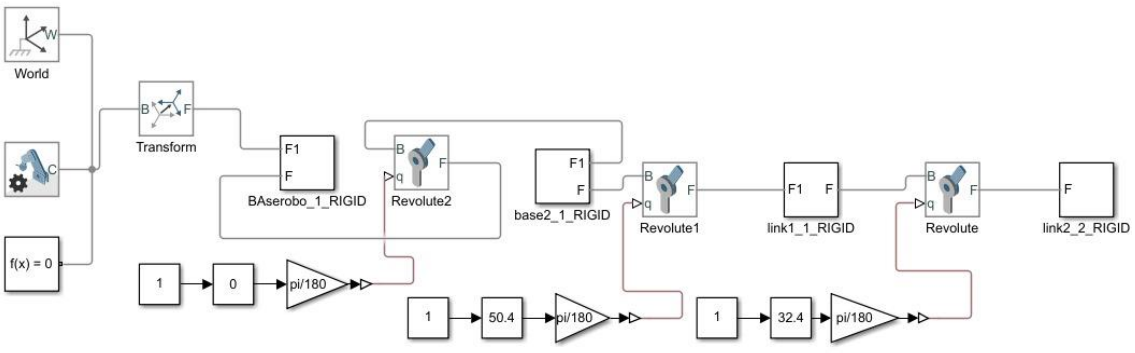


Figure 5. Model Simscape Robot

Model Simscape yang ditampilkan merupakan simulasi untuk robot 3 link yang dirancang menggunakan komponen rigid dan sambungan revolute. Sistem ini dimulai dengan kerangka referensi dunia (World Frame) yang menjadi acuan utama posisi dan orientasi robot dalam ruang 3D. Transformasi digunakan untuk menghubungkan dunia ke basis robot, memungkinkan integrasi

gerakan dengan lingkungan simulasi. Robot terdiri dari tiga sambungan revolute, masing-masing memungkinkan gerakan rotasi yang merepresentasikan derajat kebebasan (DOF) pada setiap joint. Sambungan ini menggerakkan komponen rigid yang membentuk struktur fisik robot, seperti basis, link 1, dan link 2. Input kontrol diberikan dalam bentuk nilai rotasi, yang kemudian dikonversi dari derajat ke radian agar sesuai dengan perhitungan simulasi. Setiap gerakan dihitung berdasarkan hubungan antar joint dan link, menghasilkan visualisasi kinematika robot, termasuk pergerakan ujung robot (end-effector). Secara keseluruhan, model ini memungkinkan analisis kinematika dan validasi desain robot 3 link sebelum diterapkan secara fisik.

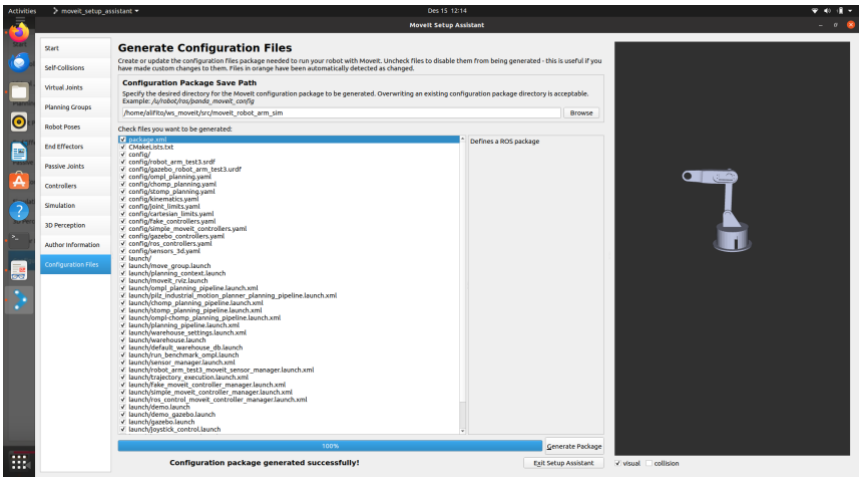


Figure 6. Model ROS

Model robot pada gambar di atas telah diimpor ke MoveIt dari SolidWorks menggunakan ekstensi SW2URDF yang memungkinkan pengguna untuk mengekspor model SolidWorks sebagai file URDF untuk ROS. Robot ini memiliki konfigurasi 3-DOF dengan tiga joint revolute: dua pada lengan dan satu pada pangkalan (base). Joint pada lengan memungkinkan rotasi hingga 180 derajat, memberikan fleksibilitas untuk melakukan tugas seperti mengambil, meletakkan, atau bermanuver di dalam ruang kerja semi-sferis. Sementara itu, joint pada pangkalan mendukung rotasi penuh 360 derajat, memungkinkan robot untuk beroperasi di seluruh area melingkar tanpa perlu reposisi.

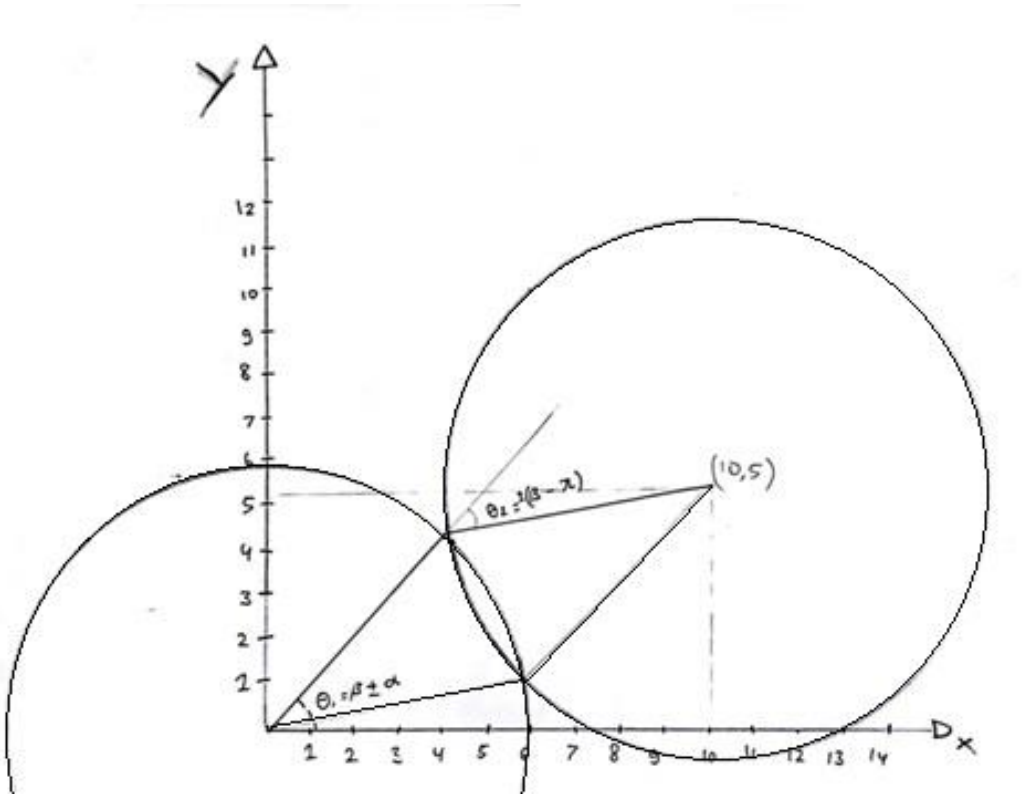
Konfigurasi ini membuat robot sangat cocok untuk aplikasi yang membutuhkan fleksibilitas dan jangkauan gerak yang luas, seperti manipulasi objek, penyortiran, atau tugas perakitan sederhana. Paket MoveIt yang dihasilkan menyediakan kemampuan perencanaan kinematika, pemeriksaan tabrakan, dan perencanaan lintasan, sehingga robot dapat melakukan pergerakan secara efisien dan aman baik dalam simulasi maupun lingkungan dunia nyata.

Berikut ini merupakan kalkulasi dan aplikasi dari teori *robotic kinematics & dynamics* yang kami gunakan dalam proyek ini:

1. Kinematics

   Robot yang digunakan merupakan robot dengan 3-DOF, terdiri dari 2-link planar arm dan satu servo motor 360 derajat yang berfungsi untuk memutar base robot. Oleh karena itu, analisis akan dimulai dengan menghitung sudut pada 2-link planar arm (θ1 dan θ2). Selanjutnya, sudut pada joint ketiga (θ3) akan dihitung menggunakan fungsi arctan untuk menentukan rotasi pada base robot.

   Perhitungan 2 link arm :



$$\alpha = \cos^{-1}\left(\frac{l_1^2 + c^2 - l_2^2}{2l_1 c}\right)$$

$$\beta = \cos^{-1}\left(\frac{l_1^2 + l_2^2 - c^2}{2l_1 l_2}\right)$$

L1 = Panjang link 1

L2 = Panjang link 2

$c = \sqrt{a^2 + b^2}$

$$\phi = \tan^{-1}\left(\frac{a}{b}\right)$$
$$= \tan^{-1}\left(\frac{y - l_3 \sin\theta}{x - l_3 \cos\theta}\right)$$

Kemudian sudut ini dapat dicari dengan cara :

$$\theta_1 = \phi \pm \alpha$$
$$\theta_2 = \pm(\beta - \pi)$$

$$c = \sqrt{10^2 + 5^2} = \sqrt{100 + 25} = 5\sqrt{5}$$

$$\alpha = cos^{-1}\left(\frac{12^2 + \sqrt{125}^2 - 12^2}{2\,(12)(12)}\right) = 62$$

$$\beta = cos^{-1}\left(\frac{12^2 + 12^2 - \sqrt{125}^2}{2\,(12)(12)}\right) = 55$$

$$\theta = tan^{-1}\left(\frac{5}{10}\right) = 26$$

$$\theta 1 = \beta \pm \alpha$$

$$\theta 2 = \pm(\beta - \pi)$$

$$\theta 1 = 26 \pm 62 = 88, -36$$

$$\theta 2 = \pm(55 - 180) = \pm(-125)$$

2. Dynamics

   Dalam proyek ini, dinamika digunakan untuk menganalisis dan menentukan kekuatan maksimum yang dapat dicapai oleh lengan robot.

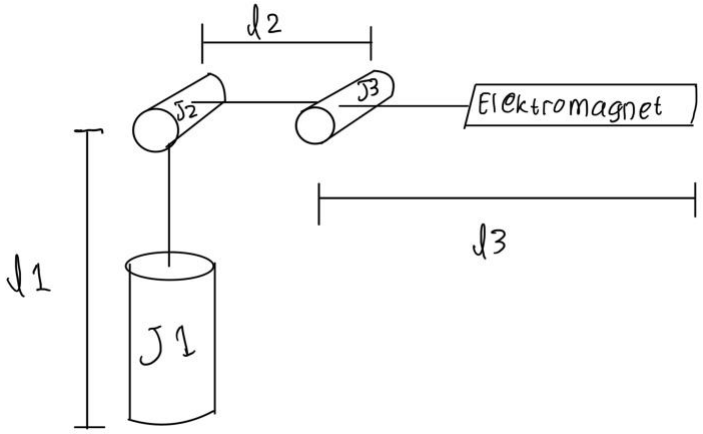


Figure 7. Robot Dynamics

Dinamika dalam proyek ini dimulai dengan perhitungan pada sendi ketiga dari robot, yang berfungsi sebagai lengan untuk mengangkat benda. Oleh karena itu, torsi pada lengan ketiga perlu dihitung. Titik pusat massa dari benda yang diangkat berada pada jarak l_3 dari sumbu rotasi sendi ketiga.

Ketika sudut lengan berada pada 0 derajat, jarak titik pusat massa benda adalah l_3. Namun, jika lengan membentuk sudut tertentu, maka jarak efektifnya menjadi $l_3 cos(x)$. Gaya gravitasi yang bekerja pada benda ditentukan oleh massa benda (*Mbenda*) dikalikan dengan percepatan gravitasi (g), yang dirumuskan sebagai $Wbenda = Mbenda * g$

Akibatnya, torsi total yang dihasilkan oleh sendi ketiga pada benda yang diangkat dapat dirumuskan sebagai:

$$\tau = l_3 cos(x) \cdot Wbenda$$

Titik pusat massa dari lengan 3 berada pada sendi ketiga dan end-effector (elektromagnet) yang terletak pada jarak $\frac{l_3}{2}$ dari sumbu rotasi. Ketika sudut xx tidak bernilai nol, posisi titik pusat massa

lengan 3 berada pada jarak $\frac{l_3}{2}$cos(x). Dengan demikian, torsi pada lengan 3 dapat dirumuskan sebagai:

$$torque_2 = (l_3 cos(x) \cdot Wbenda) + (\frac{l_3}{2} cos(x).Wlink3)$$

Untuk sendi kedua, diperlukan perhitungan torsi yang meliputi beban dari benda, lengan 3, dan lengan 2. Setelah semua torsi di atas dihitung, torsi total pada sendi kedua dapat dinyatakan sebagai:

$$torque_2 = ((l_2 cos(x) + l_3 cos(x)) \cdot Wbenda) + (l_2 cos(x) + \frac{l_3}{2} cos(x)) \cdot Wlink_3) + (l_2 cos(x) \cdot Wjoint_3) + (\frac{l_2}{2} cos(x) \cdot Wlink_2)$$

Pada sendi pertama (l_1), torsi bernilai nol karena lengan robot yang digunakan tidak memiliki torsi yang bekerja pada sendi ini. Hal ini disebabkan oleh fakta bahwa semua gaya menuju ke sendi 1 melalui sendi 2, sehingga Torque1=0

$$torque_1 = 0$$

Kemudian berikut ini merupakan aplikasi dari rumus yang telah dipaparkan ke dalam robot sorting gudang

Link 2,3 = 12cm (0.12m)

Massa benda = 50 gram (0.05kg)

Massa link 3 =47gram (0.047kg)

Massa link 2 = 35gram (0.035kg)

Joint 2,3 = 55 gram (0.055kg)

$torque_3 = (0.12cos(0) \cdot 0.05) + (\frac{0.12}{2} cos(0) \cdot 0.047)$

$torque_3 = 0.006 + 0.00282 = 0.00882$Nm

$torque_2 = (0.12cos(0) + 0.12cos(0)) \cdot 0.05) + (0.12cos(0) + \frac{0.12}{2}cos(0)) \cdot 0.047)$

$+ (0.12cos(0) \cdot 0.055) + (\frac{0.12}{2}cos(0) \cdot 0.035)$

$torque_2 = 0.126 + 0.12282 + 0.0066 + 0.0021 = 0.25752Nm$

Perhitungan di atas merupakan besarnya torsi yang bekerja pada sendi2 dan sendi3.

Control System

Dikarenakan PID akan diaplikasikan ke dalam sistem robot, yaitu pada bagian servo 360° yang terdapat pada alas, maka harus dicari tahu transfer function dari komponen tersebut. Oleh karena itu, perlu diturunkan dari diagram servo yang didapatkan dari publikasi berjudul "Physical Modeling and Parameters Identification of the MG995 Servomotor" [14].

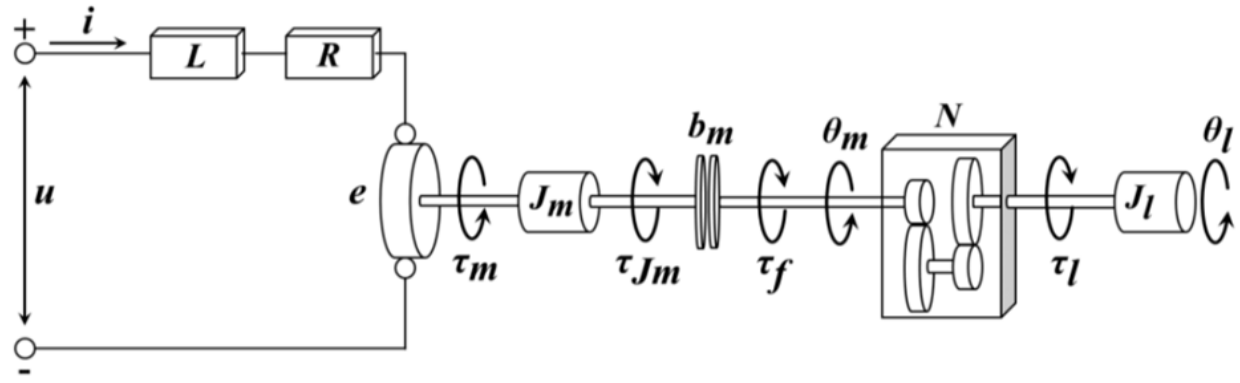


Figure 8. Diagram servo MG995

Dari diagram di atas, didapatkan beberapa variabel. Di mana u adalah tegangan yang diberikan ke terminal motor, i adalah arus listrik, R adalah hambatan motor, L adalah induktansi, dan e adalah gaya gerak listrik balik. Unsur mekanisnya adalah τm yang merupakan torsi yang dihasilkan motor, Jm adalah momen inersia motor, bm adalah koefisien gesek viskos, τf adalah torsi gesek viskos, θm adalah sudut putaran poros motor, N adalah rasio roda gigi gearbox, Jl adalah momen inersia beban yang dipasang pada poros keluaran servo, τl adalah torsi yang diterapkan pada beban pada poros keluaran dan, terakhir, θl adalah sudut rotasi poros keluaran servo.

Dari sumber yang sama, juga didapatkan nilai-nilai untuk variabel-variabel tersebut.

Variable (symbol)	Value	Variable (symbol)	Value

Motor Polar Moment of Inertia (Jm)	6.3173*(10^{-7}) kg·m²	Gear ratio (N)	275.6923
Viscous Friction Coefficient (bm)	2.49797*(10^{-7}) Nms	Gearbox Efficiency Factor (μ)	0.81156
Motor Resistance (R)	2.5 Ω	Torque Constant (Kt)	5.8857*(10^{-3}) N·m/A
Inductance (L)	0	Load Polar Moment of Inertia (Jl)	0.00216 kg·m²

L ≈ 0 karena dinamika kelistrikan jauh lebih cepat dibandingkan dengan dinamika mekanis, dikarenakan konstanta waktu R/L jauh lebih kecil dibandingkan konstanta waktu mekanis.

Untuk mencari nilai Jl di atas, perlu dilakukan tahap-tahap berikut:

1. Simplify bentuk dari robot

   Untuk mempermudah proses kalkulasi, bentuk lengan robot disederhanakan menjadi balok.

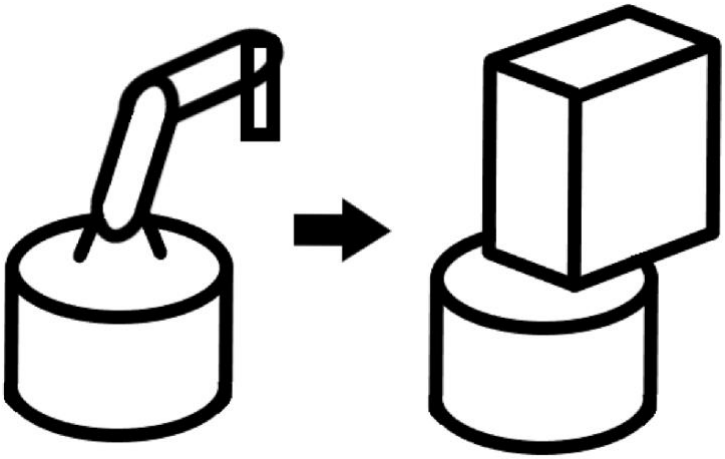


Figure 9. Gambaran simplifikasi bentuk lengan robot untuk mempermudah kalkulasi

2. Formula yang digunakan

$$J = \frac{1}{3}mL^2$$

3. Tentukan nilai dari barang: Mass (m) = 450g, Length (L) = 12cm
4. Kalkulasi

$$J = \frac{1}{3} \times 0.45 \times (0.12)^2$$

$$J = \frac{1}{3} \times 0.45 \times 0.0144$$

$$J = \frac{1}{3} \times 0.00648$$

$$J = 0.00216\,\mathrm{kg \cdot m^2}$$

Untuk mencari transfer function dari sistem, perlu dilakukan seperti berikut:

1. Persamaan keseimbangan torsi motor servo

$$J_l\ddot{\theta}_l = \eta N \left[K_t i - b_m\dot{\theta}_m - J_m\ddot{\theta}_m\right]$$

2. Daya pada poros keluaran berkaitan dengan daya poros masukan yang dikurangi dengan efisiensi gearbox

$$P_m = P_l \implies \dot{\theta}_m = N\dot{\theta}_l \implies \ddot{\theta}_m = N\ddot{\theta}_l$$

3. Terapkan power loss ke persamaan torsi

$$J_l\ddot{\theta}_l = \eta N \left[K_t i - b_m N\dot{\theta}_l - J_m N\ddot{\theta}_l\right]$$

4. Ubah persamaan sebelumnya menggunakan transformasi Laplace, manipulasi aljabar, dll.

$$(J_m\eta N^2 + J_\theta)s^2\Theta_1(s) + b_m\eta N^2 s\Theta_1(s) = \eta N K_t I(s)$$

$$\downarrow$$

$$\frac{\Theta_l(s)}{I(s)} = \frac{N\eta K_t}{\left(J_l + J_m\eta N^2\right)s^2 + \left(b_m\eta N^2\right)s}$$

5. Persamaan tegangan motor

$$u = Ri + L\frac{di}{dt} + \frac{N\dot{\theta}_l}{K_\omega}$$

6. Transformasi Laplace persamaan tegangan motor

$$U(s) = (R + Ls)\, I(s) + \frac{Ns}{K_\omega}\Theta_l(s)$$

7. Gabungkan persamaan 6 ke persamaan 4

$$P(s) = \frac{\Theta_l(s)}{U(s)} = \frac{\eta N K_t}{\{(R + Ls)\,[(J_m \eta N^2 + J_l)\, s + b_m \eta N^2] + \eta N^2 K_t^2\}\, s}$$

Perlu dilakukan perubahan pada bentuk persamaan menjadi seperti berikut agar mudah diproses dalam MATLAB:

$$P(s) = \frac{\Theta_l(s)}{U(s)} = \frac{\eta N K_t}{\{(R + Ls)[(J_m \eta N^2 + J_l)s + b_m \eta N^2] + \eta N^2 K_t^2\}s}$$

$$= \frac{\eta N K_t}{\{(R + Ls)(J_m \eta N^2 + J_l)s + (R + Ls) b_m \eta N^2 + \eta N^2 K_t^2\}s}$$

$$= \frac{\eta N K_t}{\{(R J_m \eta N^2 + R J_l)s + (L J_m \eta N^2 + L J_l)s^2 + R b_m \eta N^2 + L b_m \eta N^2 s + \eta N^2 K_t^2\}s}$$

$$= \frac{\eta N K_t}{\{L(J_m \eta N^2 + J_l)s^3 + [R(J_m \eta N^2 + J_l) + L b_m \eta N^2]s^2 + (R b_m \eta N^2 + \eta N^2 K_t^2)s\}}$$

Setelah didapatkan *transfer function* di atas, substitusi nilai-nilai yang diketahui ke dalam fungsi, sehingga mendapatkan fungsi berikut

$$\frac{1.31687148}{0.09957849517s^2 + 2.175332158s}$$

Karena sistem di atas masih dalam bentuk *continuous,* maka perlu diproses menggunakan ZOH agar menjadi *discrete.*

Zero-Order Hold (ZOH) adalah metode yang banyak digunakan untuk merekonstruksi sinyal secara real time. Dalam metode rekonstruksi penahan orde nol, sinyal kontinu direkonstruksi dari sampelnya dengan menahan sampel tertentu selama jangka waktu tertentu hingga sampel berikutnya diterima. Oleh karena itu, ZOH menghasilkan perkiraan langkah.

Berikut adalah gambaran dari proses rekonstruksi ulang sinyal menggunakan ZOH:

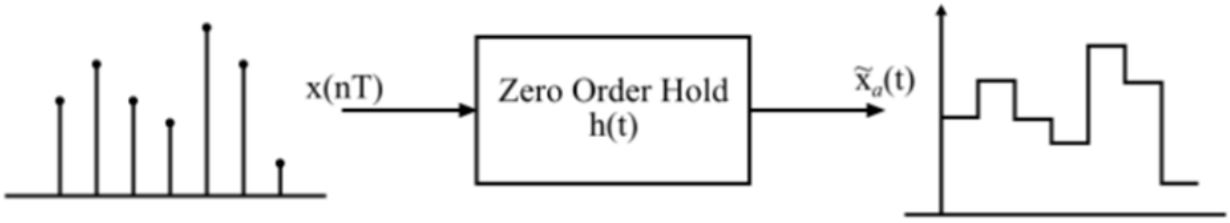


Figure 10. Proses cara kerja ZOH

Berikut adalah proses konversi menggunakan MATLAB:

```
20 -    num2 = [1.31687148];
21 -    den2 = [0.09957849517 2.175332158 0];
22 -    G2 = tf(num2, den2);
23
24      % Gs to Gz Conversion (To discrete time system)
25 -    T = input('Type period T ');  % Input sampling interval.
26 -    G2z = c2d(G2,T,'zoh')         % Convert G1 (s) in cascade with z.o.h. to G(z) and display.
```

```
Command Window
>> TesControl
Type period T 0.1

G2z =

   0.03594 z + 0.01778
  ----------------------
  z^2 - 1.113 z + 0.1125

Sample time: 0.1 seconds
Discrete-time transfer function.

fx >>
```

Figure 11. Proses konversi transfer function dari kontinuous menjadi diskrit menggunakan MATLAB

Sehingga didapatkan perubahan seperti berikut

$$\frac{1.31687148}{0.09957849517s^2+2.175332158s} \Rightarrow \frac{0.03594z+0.01778}{z^2-1.113+0.1125}$$

Setelah mendapatkan sistem *discrete* di atas, dapat dikeluarkan root locus dari fungsi tersebut untuk menentukan nilai *gain* untuk mendapatkan *damping* yang diinginkan. Seperti dalam gambar di bawah, dapat dilihat bahwa ada dua titik yang dipilih. Yang berwarna kuning adalah nilai *gain* yang perlu digunakan untuk mendapatkan *damping* sebesar 0.49. Yang berwarna hijau adalah nilai *gain* yang perlu digunakan untuk mendapatkan *damping* sebesar 1, yaitu saat *gain* bernilai 5.47.

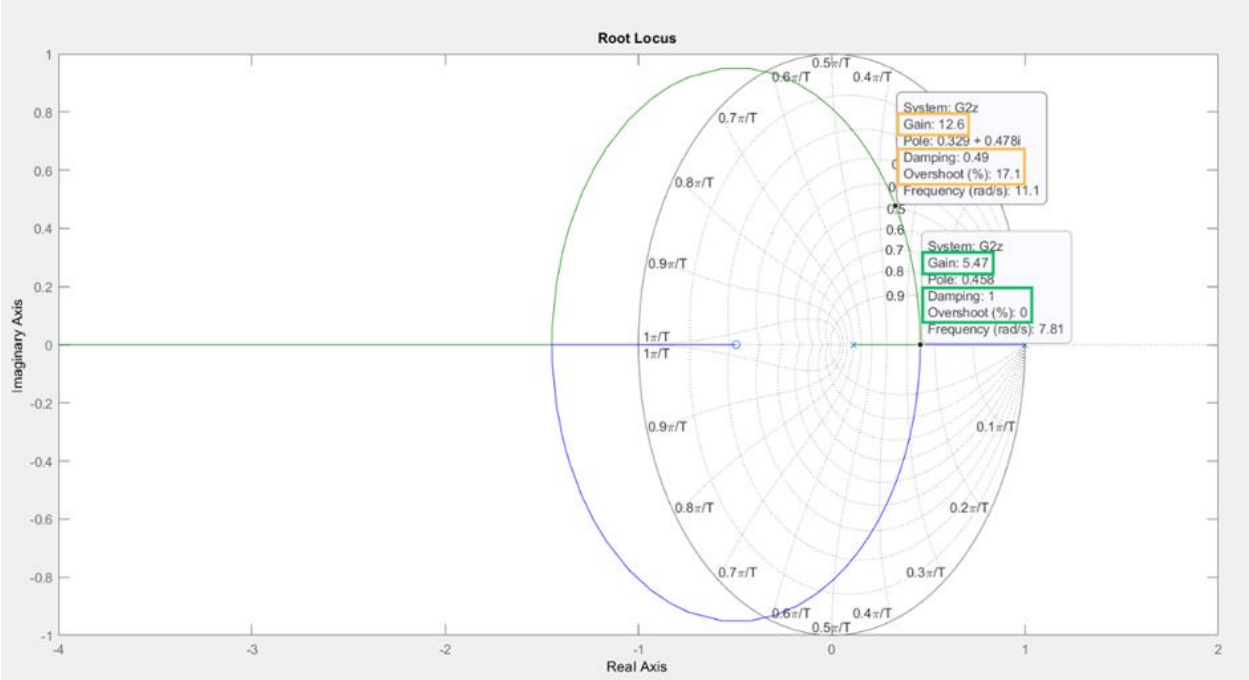


Figure 12. Hasil plot root locus dari discrete transfer function

Berikut adalah sistem yang digunakan untuk mensimulasikan PID pada servo

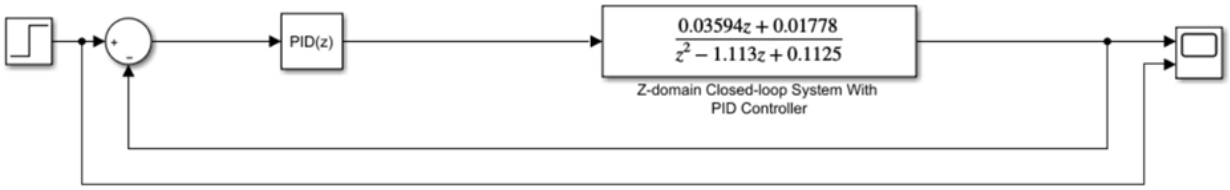


Figure 13. Sistem dengan PID

Menggunakan sistem di atas, dapat diaplikasikan nilai yang didapatkan dari root locus. Namun untuk kepentingan projek ini, hanya akan digunakan nilai *gain* yang menghasilkan *damping* sebesar 1. Kemudian, dipasangkan nilai yang diinginkan sebesar 60. Waktu pengambilan sampel dibuat setiap 0.1 detik. Hasil dari sistem tersebut adalah seperti berikut:

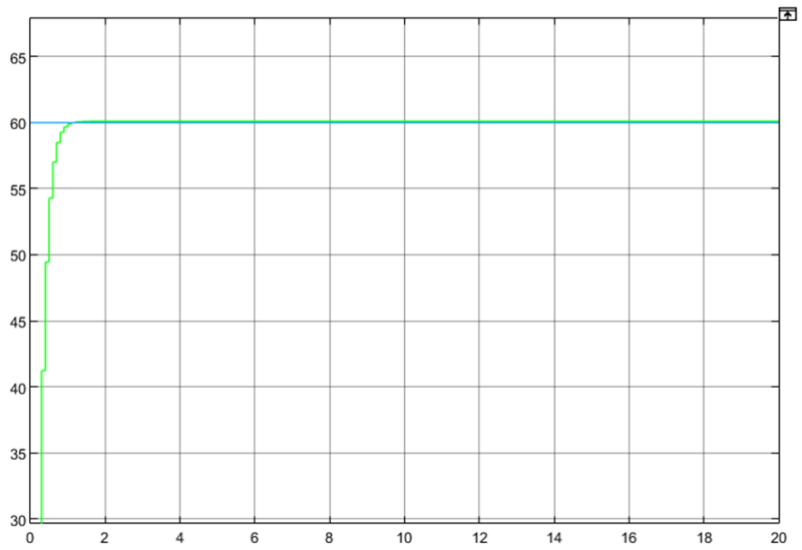


Figure 14. Hasil plot sistem dengan PID

Dapat dilihat bahwa sistem mencapai kestabilan di nilai yang diinginkan dalam waktu kurang dari 1 detik. Namun untuk sistem ini, juga perlu ditambahkan saturasi. Alasan digunakannya saturasi adalah sebagai pembatas voltase input dengan batas terendah sebesar -12 dan batas tertinggi sebesar +12 untuk membatasi tegangan *input* agar tidak membebani servo yang berisiko merusak servo. Waktu pengambilan sampel masih dibuat setiap 0.1 detik. Berikut adalah sistem dengan PID dan saturasi:

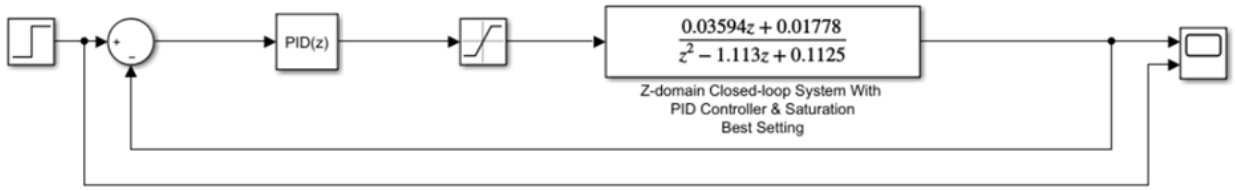


Figure 15. Sistem dengan PID dan saturasi

Menggunakan sistem di atas, didapatkan hasil di bawah saat menggunakan parameter PID dan input yang sama seperti sistem sebelumnya.

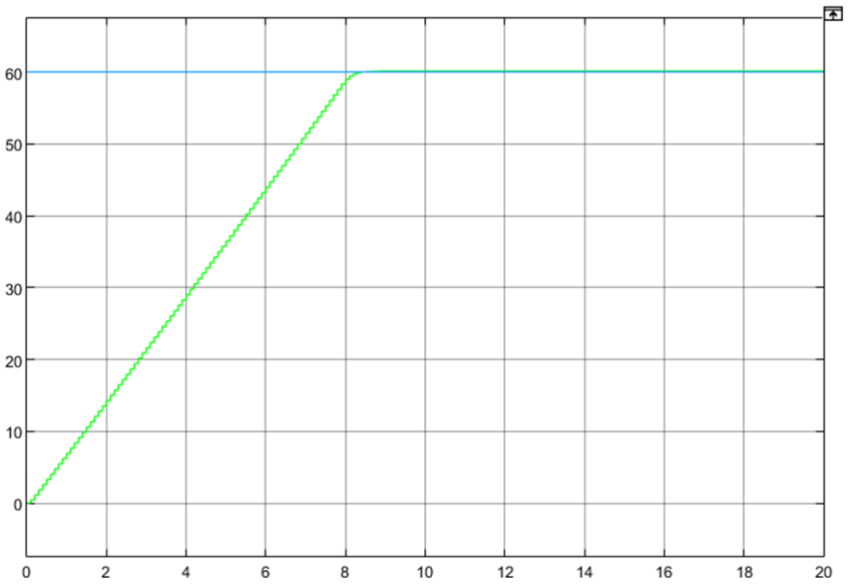


Figure 16. Hasil plot sistem dengan PID dan saturasi

Flowchart

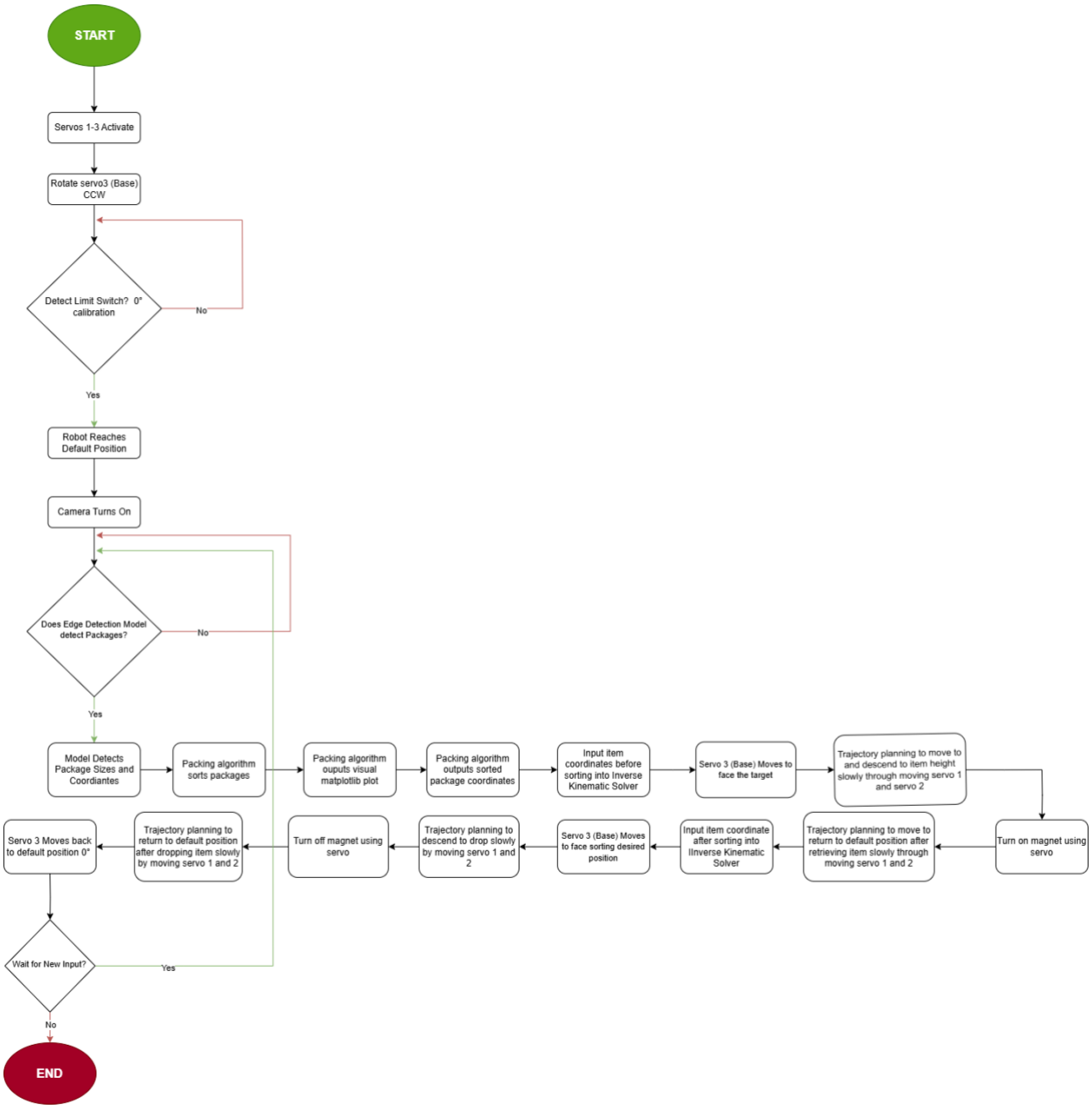


Figure 17. Diagram Flowchart

Diagram di atas menunjukkan *flowchart* seluruh sistem kami. Dimulai dengan aktivasi ketiga servo, setelah itu servo *base* bergerak hingga mencapai *limit switch*. Setelah mencapai *limit switch*, servo akan berhenti, mencapai posisi *default*. Setelah itu, kamera menyala dan mencari objek, ukuran dan variabel koordinat objek yang terdeteksi dimasukkan ke dalam algoritma *packing* rectpack. Algoritma ini kemudian mengeluarkan koordinat yang ideal untuk pengemasan setiap objek yang terdeteksi menurut perhitungannya.

Setelah proses *packing* selesai, koordinat objek sebelum *packing* di input ke dalam *Inverse Kinematics Solver* dan servo bergerak sesuai dengan *trajectory planning* untuk mengambil objek supaya lengan robot dan objek tersebut tidak bertabrakan dengan apa pun. *Inverse Kinematics Solver* kemudian diberi koordinat objek yang sama setelah *packing* dan servo-servo kemudian bergerak sesuai dengan *trajectory planning* untuk memindah objek ke lokasi pengemasan.

BAB IV HASIL DAN PEMBAHASAN

4.1 Hasil Pembuatan Robot

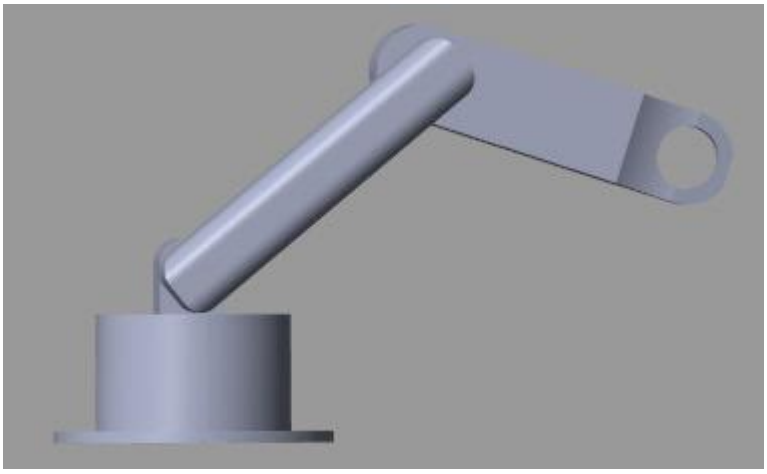
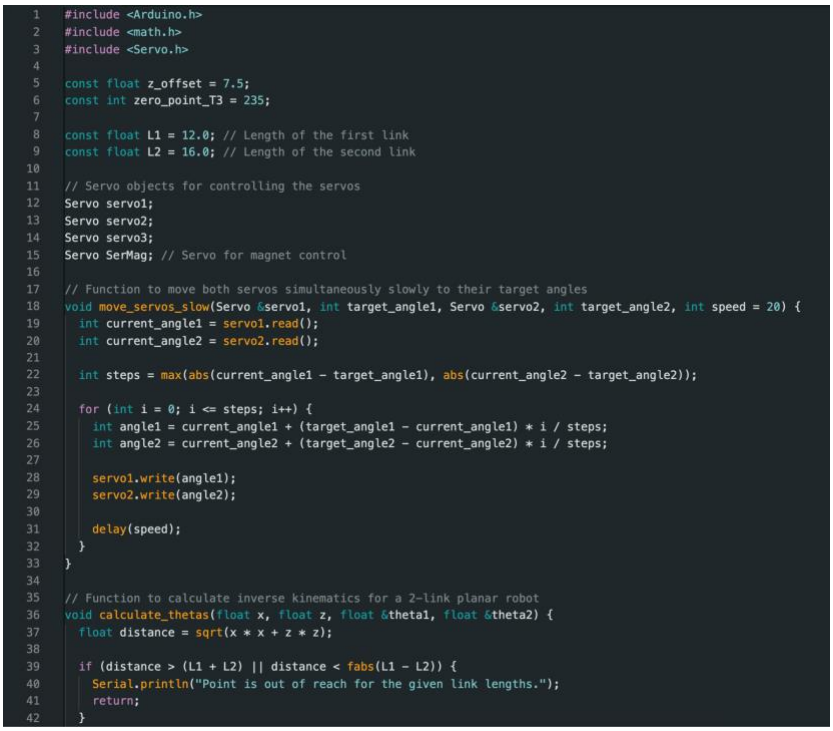


Figure 18. Hasil Simulasi Simscape

Pada hasil simulasi menggunakan model Simscape untuk robot 3 link, end-effector belum dapat dimodelkan karena keterbatasan dalam merepresentasikan elektromagnetik pada lingkungan Simscape. Namun, berdasarkan pengujian yang telah dilakukan, struktur lengan, sendi, servo, dan end-effector telah berfungsi dengan baik secara fisik. End-effector elektromagnetik mampu mengangkat objek dengan akurat, tetapi tidak dapat beroperasi dalam jangka waktu lama karena panas yang dihasilkan dapat menghilangkan sifat magnet penghantarnya. Untuk sendi yang menggunakan servo 180°, pengujian menunjukkan hasil yang akurat, di mana setiap servo mampu menjalankan trajectory planning dengan tepat sesuai dengan nilai theta yang dihasilkan melalui metode machine learning.

Servo pada alas, yang menggunakan servo 360°, juga menunjukkan performa yang cukup akurat. Namun, terdapat keterbatasan teknis di mana servo harus kembali ke posisi 0° sebelum menuju sudut

tujuan. Hal ini disebabkan oleh keterbatasan pada perangkat, khususnya pada feedback dari rotary encoder, yang memiliki limitasi dalam pergerakan dua arah. Dari sisi struktur lengan, pengujian menunjukkan bahwa lengan telah memenuhi spesifikasi yang diinginkan. Lengan mampu menahan torsi yang dihasilkan selama operasi dengan baik, sehingga mendukung stabilitas dan akurasi gerakan secara keseluruhan. Hasil ini menunjukkan bahwa sistem robot secara keseluruhan telah bekerja sesuai harapan, meskipun ada beberapa batasan yang memerlukan perhatian lebih lanjut untuk pengembangan di masa depan.



```
#include <Arduino.h>
#include <math.h>
#include <Servo.h>

const float z_offset = 7.5;
const int zero_point_T3 = 235;

const float L1 = 12.0; // Length of the first link
const float L2 = 16.0; // Length of the second link

// Servo objects for controlling the servos
Servo servo1;
Servo servo2;
Servo servo3;
Servo SerMag; // Servo for magnet control

// Function to move both servos simultaneously slowly to their target angles
void move_servos_slow(Servo &servo1, int target_angle1, Servo &servo2, int target_angle2, int speed = 20) {
  int current_angle1 = servo1.read();
  int current_angle2 = servo2.read();

  int steps = max(abs(current_angle1 - target_angle1), abs(current_angle2 - target_angle2));

  for (int i = 0; i <= steps; i++) {
    int angle1 = current_angle1 + (target_angle1 - current_angle1) * i / steps;
    int angle2 = current_angle2 + (target_angle2 - current_angle2) * i / steps;

    servo1.write(angle1);
    servo2.write(angle2);

    delay(speed);
  }
}

// Function to calculate inverse kinematics for a 2-link planar robot
void calculate_thetas(float x, float z, float &theta1, float &theta2) {
  float distance = sqrt(x * x + z * z);

  if (distance > (L1 + L2) || distance < fabs(L1 - L2)) {
    Serial.println("Point is out of reach for the given link lengths.");
    return;
  }
```

Figure 19. kode robot part1

```
  float alpha = acos((L1 * L1 + distance * distance - L2 * L2) / (2 * L1 * distance));
  float beta = acos((L1 * L1 + L2 * L2 - distance * distance) / (2 * L1 * L2));
  float phi = atan2(z, x);

  theta1 = phi + alpha;
  theta2 = M_PI - beta;

  theta1 = degrees(theta1);
  theta2 = degrees(theta2);
}

float calculate_theta3_with_zones(float x, float z) {
  if (z >= 0) {
    z = z + 8;
    if (z == 0) {
      Serial.println("Invalid input for theta3 calculation: z + 8 cannot be zero.");
      return -999;
    }
  } else {
    z = z - 8;
    if (z == 0) {
      Serial.println("Invalid input for theta3 calculation: z - 8 cannot be zero.");
      return -999;
    }
  }

  float theta3_rad = atan2(x, z);
  float theta3_deg = degrees(theta3_rad);

  if (x == 0 && z <= 0) {
    theta3_deg = -(zero_point_T3 - theta3_deg);
  } else if (x > 0 && z <= 0) {
    theta3_deg = -(180 - theta3_deg);
  } else {
    theta3_deg = -(theta3_deg + zero_point_T3);
  }
```

Figure 20. kode robot part2

```

  if (theta3_deg >= -245 && theta3_deg <= -235) {
    theta3_deg -= 1;
  } else if (theta3_deg < -245) {
    theta3_deg -= 14;
  } else {
    // pass
  }

  return theta3_deg;
}

void setup() {
  Serial.begin(9600);
  while (!Serial) {
    // Wait for the serial port to connect.
  }

  servo1.attach(9);
  servo2.attach(10);
  servo3.attach(11);
  SerMag.attach(12); // Attach the magnet servo

  servo1.write(130);
  servo2.write(90);

  SerMag.write(23);

  Serial.println("Please input x, z coordinates in the format: x,z");
}

void loop() {
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n');
    input.trim();

    // Sanitize input to remove any non-printable characters
    for (int i = 0; i < input.length(); i++) {
      if (!isPrintable(input[i])) {
        input.remove(i);
        i--;
      }
    }
```

Figure 21. kode robot part3

```
124   if (input.length() == 0) {
125     Serial.println("Invalid input. Please input x, z coordinates in the format: x,z");
126     return;
127   }
128
129   int comma = input.indexOf(',');
130
131   if (comma > 0) {
132     String x_str = input.substring(0, comma);
133     String z_str = input.substring(comma + 1);
134
135     float x = x_str.toFloat();
136     float z = z_str.toFloat();
137
138     if (z >= 0) {
139       z += z_offset;
140     } else {
141       z -= z_offset;
142     }
143
144     float theta3 = calculate_theta3_with_zones(x, z);
145
146     if (theta3 == -999) {
147       return;
148     }
149
150     Serial.print("Calculated Theta 3: ");
151     Serial.println(theta3, 2);
152
153
154     ////////////////////IF DOESN'T WORK, DELETE THIS PART//////////////////
155     servo1.write(130);
156     servo2.write(90);
157     Serial.println("Servo 1 and 2 default position");
158     ////////////////////IF DOESN'T WORK, DELETE THIS PART//////////////////
159
160
```

Figure 22. kode robot part4

```
161       Serial.println("Press Enter to continue with Theta 1 and Theta 2 calculations.");
162       while (Serial.available() == 0) {
163         // Wait for user input
164       }
165       Serial.read(); // Clear the input buffer
166
167       float distance = sqrt(x * x + z * z);
168       float target_y = 3;
169
170       for (float current_y = 18; current_y >= target_y; current_y -= 3) {
171         float theta1, theta2;
172         calculate_thetas(distance, current_y, theta1, theta2);
173
174         Serial.print("For y = ");
175         Serial.print(current_y);
176         Serial.print(", Calculated angles - Theta 1: ");
177         Serial.print(theta1, 2);
178         Serial.print(", Theta 2: ");
179         Serial.println(theta2, 2);
180
181         int servo1_angle = constrain(map(theta1, 0, 180, 0, 180), 0, 180);
182         int servo2_angle = constrain(map(theta2, 0, 180, 0, 180), 0, 180);
183
184         move_servos_slow(servo1, servo1_angle, servo2, servo2_angle, 30);
185         servo3.write(constrain(map(theta3, -235, 235, 0, 180), 0, 180));
186
187         Serial.println("Press Enter to continue.");
188         while (Serial.available() == 0) {
189           // Wait for user input
190         }
191         Serial.read(); // Clear the input buffer
192       }
193
194       SerMag.write(10);
195       delay(500);
196       SerMag.write(23);
197       delay(500);
198       Serial.println();
199       Serial.println("Magnet Toggled");
200       Serial.println();
201
```

Figure 23. kode robot part 5

```
202       for (float current_y = target_y; current_y <= 18; current_y += 3) {
203         float theta1, theta2;
204         calculate_thetas(distance, current_y, theta1, theta2);
205
206         Serial.print("For y = ");
207         Serial.print(current_y);
208         Serial.print(", Calculated angles - Theta 1: ");
209         Serial.print(theta1, 2);
210         Serial.print(", Theta 2: ");
211         Serial.println(theta2, 2);
212
213         int servo1_angle = constrain(map(theta1, 0, 180, 0, 180), 0, 180);
214         int servo2_angle = constrain(map(theta2, 0, 180, 0, 180), 0, 180);
215
216         move_servos_slow(servo1, servo1_angle, servo2, servo2_angle, 30);
217         servo3.write(constrain(map(theta3, -235, 235, 0, 180), 0, 180));
218
219         Serial.println("Press Enter to continue.");
220         while (Serial.available() == 0) {
221           // Wait
222         }
223         Serial.read(); // Clear the input buffer
224       }
225
226       move_servos_slow(servo1, 130, servo2, 90, 30);
227       delay(2000);
228       Serial.println("Servos reset to Theta 1 = 130, Theta 2 = 90.");
229       Serial.println("Please input x, z coordinates in the format: x,z");
230     } else {
231       Serial.println("Invalid input format. Please enter in the format: x,z");
232     }
233   }
234
235   delay(100);
```

Figure 24. kode robot part6

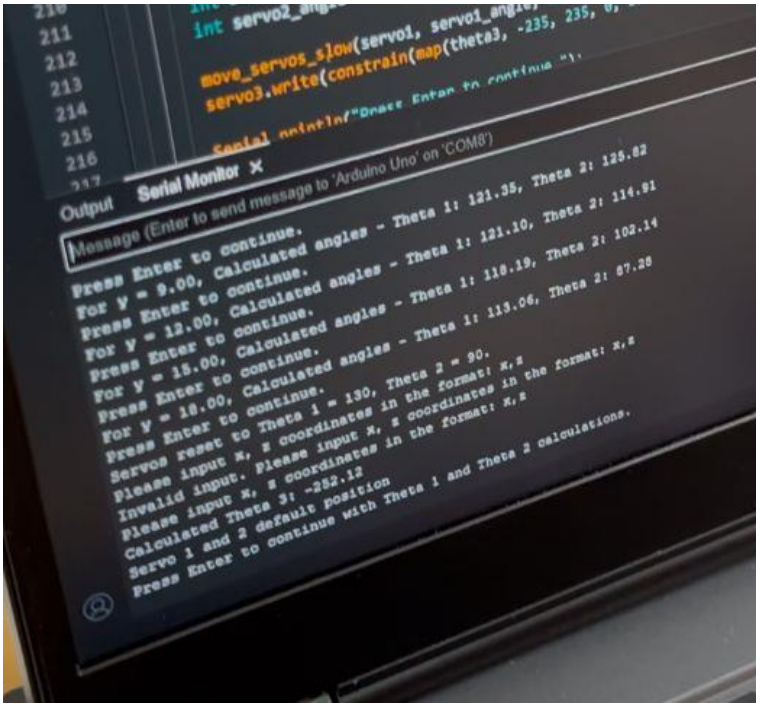


Figure 25. Hasil Kode yang telah dijalankan

Kode di atas merupakan implementasi untuk menghitung sudut pergerakan setiap sendi robot dengan menggunakan metode *trajectory planning* dan *inverse kinematics*. Proses ini bertujuan untuk menentukan nilai sudut θ (theta) yang optimal bagi servo agar dapat bergerak secara presisi. Nilai sudut ini digunakan untuk menggerakkan sendi robot secara bersamaan guna mengambil objek yang telah ditentukan berdasarkan koordinat hasil prediksi dari model *machine learning*.

Hasil dari kode ini menunjukkan keberhasilan integrasi antara perencanaan trayektori dan metode kinematika terbalik, sehingga robot dapat bergerak dengan akurasi tinggi untuk mengeksekusi tugasnya. Analisis ini membuktikan bahwa sistem mampu menghitung sudut gerakan dengan tepat, sesuai dengan kebutuhan operasi pengambilan barang berdasarkan posisi yang telah ditentukan.

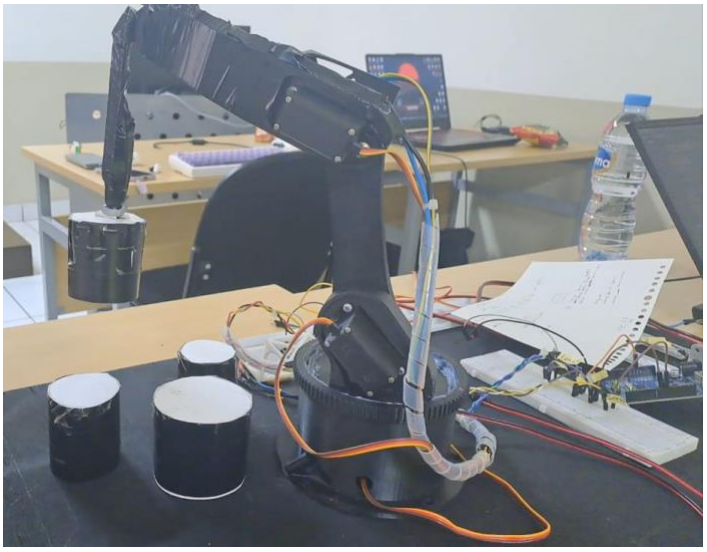


Figure 26. Robot yang bekerja

4.2 Hasil Machine Learning

Parameter eksperimen untuk menguji akurasi pixel mapping OpenCV:

- Tinggi Kamera dari alas : 43 cm

Trial Number	Expected	Predicted	Error
1	10.8 cm	10.9 cm	0.926%
2	5.6 cm	5.2 cm	7.142%
3	6.7 cm	5.8 cm	13.432%
4	11.7 cm	10.9 cm	6.838%
5	9.5 cm	8.3 cm	12.632%
6	21.5 cm	22.4 cm	4,186%
7	17.7 cm	18.5 cm	4.520%
8	8.7 cm	8.7 cm	0%
9	17.3 cm	18.4 cm	6.358%
10	7.2 cm	7.2 cm	0%

Mean Average Percentage Error : 5.603%

Model Accuracy : 94.397%

```
30
31  # Initialize webcam
32  cap = cv2.VideoCapture(1)  # Replace 0 with your camera ID or video file path
33
34  # Check if the camera is opened
35  if not cap.isOpened():
36      print("Error: Unable to access the camera.")
37      exit()
38
```

Figure 27. Turn on camera

```
# Preprocessing
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (9, 9), 0)
edged = cv2.Canny(blur, 50, 100)
edged = cv2.dilate(edged, None, iterations=1)
edged = cv2.erode(edged, None, iterations=1)

# Find contours
cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)

# Sort contours from left to right
(cnts, _) = contours.sort_contours(cnts)

# Filter small contours
cnts = [x for x in cnts if cv2.contourArea(x) > 100]

# Compute pixel-per-centimeter scale using the reference object
if cnts:
    ref_object = cnts[0]
    box = cv2.minAreaRect(ref_object)
    box = cv2.boxPoints(box)
    box = np.array(box, dtype="int")
    box = perspective.order_points(box)
    (tl, tr, _, _) = box
    pixel_dist = euclidean(tl, tr)
    pixel_per_cm = pixel_dist / ref_object_width_cm

    # Draw contours and dimensions
    for i, cnt in enumerate(cnts):
        box = cv2.minAreaRect(cnt)
        box = cv2.boxPoints(box)
        box = np.array(box, dtype="int")
        box = perspective.order_points(box)
        (tl, tr, br, bl) = box
        cv2.drawContours(frame, [box.astype("int")], -1, (0, 255, 0), 2)
```

Figure 28. pre-processing edge detection

```
# Calculate center of the bounding box in pixels
center_x_px = int((tl[0] + tr[0] + br[0] + bl[0]) / 4)
center_y_px = int((tl[1] + tr[1] + br[1] + bl[1]) / 4)

# Convert center coordinates to cm with bottom-center as origin
center_x_cm = (center_x_px - frame.shape[1] / 2) / pixel_per_cm  # X relative to center
center_y_cm = (frame.shape[0] - center_y_px) / pixel_per_cm  # Y relative to bottom

# Draw center dot
cv2.circle(frame, (center_x_px, center_y_px), 5, (0, 0, 255), -1)

# Measure dimensions
width = euclidean(tl, tr) / pixel_per_cm
height = euclidean(tr, br) / pixel_per_cm

# Add margin of 0.5 cm to each dimension
width += 0.5
height += 0.5

# Create a unique identifier for each object
object_id = (round_to_nearest_half(width), round_to_nearest_half(height),
             round_to_nearest_half(center_x_cm), round_to_nearest_half(center_y_cm))
```

Figure 29. pixel/coordinate mapping

```python
# Perform hyper-packing if sufficient rectangles are detected
if len(detected_rectangles) >= max_rectangles:
    print("\nPerforming hyper-packing of detected rectangles:")

    # Initialize the packer and disable rotation
    packer = rectpack.newPacker(rotation=False)

    # Define the margin size (gap between rectangles)
    margin = 0.2  # Adjust as needed (in cm)

    # Add all detected rectangles to the packer with margins
    for rect in detected_rectangles:
        adjusted_width = rect[0] + margin
        adjusted_height = rect[1] + margin
        rect_id = rect[4]
        # print(f" rect id = {rect_id}")
        packer.add_rect(adjusted_width, adjusted_height, rect_id)  # Add rectangles with the margin included

    # Add the container (bin) to the packer
    packer.add_bin(container_width, container_height)

    # Perform the packing process
    packer.pack()
```

Figure 30. perform packing

```python
148
149         # Show the frame
150         cv2.imshow("Live Measurement", frame)
151
152         # Break on 'q' key press
153         if cv2.waitKey(1) & 0xFF == ord('q'):
154             break
155
156         if len(detected_rectangles) >= max_rectangles:
157             time.sleep(3)
158             break
159
```

Figure 31. show camera frame

```python
327     # Set axis labels, limits, and title
328     ax.set_xlim(-container_height / 2, container_height / 2)
329     ax.set_ylim(-container_width, 0)
330     ax.set_title("Horizontal Packing Result (Top-Center Origin)")
331     ax.set_xlabel("Length (cm)")
332     ax.set_ylabel("Width (cm)")
333
334     # Maintain equal aspect ratio for proper visualization
335     plt.gca().set_aspect("equal", adjustable="box")
336     plt.show()
```

Figure 32. plot results of packing using matplotlib

4.2.1 ROS

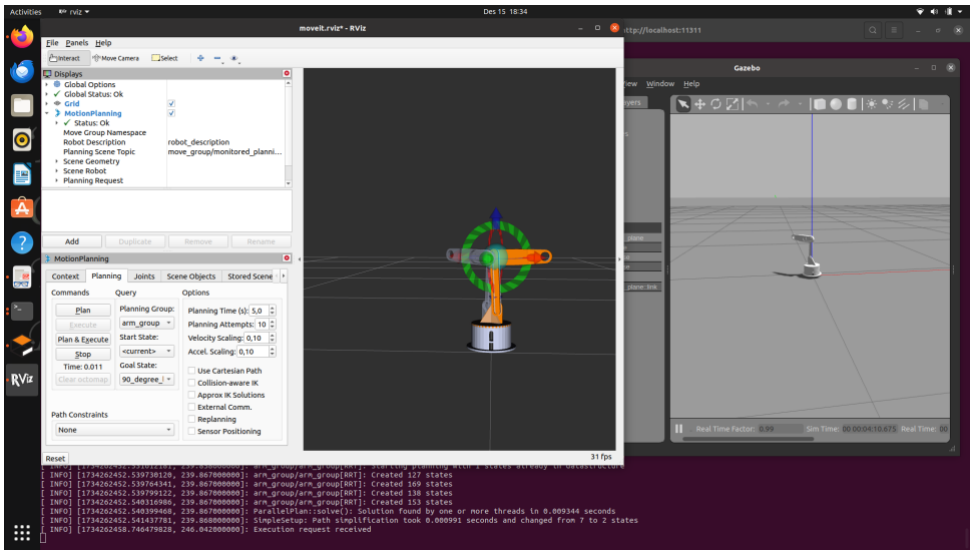


Figure 33. Model ROS di MoveIt dan Rviz

4.3 Hasil Control System

Seperti yang dijelaskan dalam prosedur, *transfer function* yang digunakan dalam sistem merupakan fungsi dalam bentuk diskrit seperti berikut:

$$\frac{0.03594z + 0.01778}{z^2 - 1.113 + 0.1125}$$

Berikut adalah sistem yang digunakan untuk mensimulasikan sistem kontrol servo 360° pada alas robot sudah diimplementasikan saturasi dan kontrol PID.

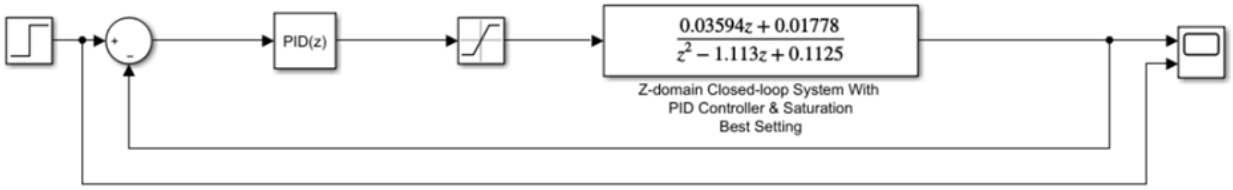


Figure 34. Sistem dengan PID dan saturasi

PID diimplementasikan berdasarkan *output* dari hasil *root locus*, yaitu nilai *gain* sebesar 5.47 untuk *proportional*. Kemudian untuk *integrator* dan *derivative* diberikan nilai 0. Saturasi dalam sistem diimplementasikan sebagai pembatas *input* agar terbatas di antara -12 dan +12. Terakhir, *sampling time* dibuat agar terjadi setiap 0.1 detik. Sehingga menghasilkan grafik seperti berikut:

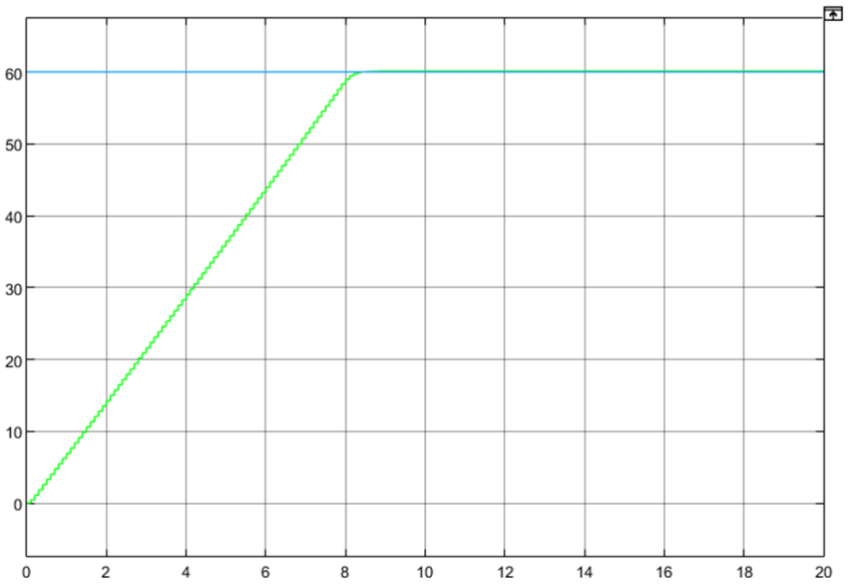


Figure 35. Hasil plot sistem dengan PID dan saturasi

Dapat dilihat bahwa hasil dari sistem tersebut mencapai nilai target secara stabil setelah sekitar 8 detik. Sistem tersebut memiliki waktu respon lebih lambat karena pembatasan sinyal kontrol dari saturasi yang menghalangi kontroler memberikan aksi koreksi besar saat error masih besar. Saturasi juga melindungi objek dari overload dengan membatasi input agar tetap dalam batas aman. Tanpa saturasi, kontroler dapat mengurangi error lebih cepat, tetapi berisiko melebihi batas kemampuan sistem.

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini berhasil mengembangkan sistem robot sorting yang mengintegrasikan teknologi machine learning, inverse kinematics, dan trajectory planning untuk meningkatkan efisiensi penyortiran barang di gudang. Sistem ini memanfaatkan algoritma computer vision (OpenCV) untuk mendeteksi dimensi dan posisi barang, serta algoritma Rectpack untuk mengoptimalkan tata letak barang. Hasil pengujian menunjukkan bahwa robot mampu menjalankan tugasnya dengan akurasi tinggi, dengan nilai rata-rata kesalahan prediksi posisi sebesar 5,603%, yang menunjukkan efektivitas integrasi teknologi ini.

Dari sisi mekanisme, robot tiga derajat kebebasan (3-DOF) yang dirancang menggunakan servo motor 360° dan 180° telah mampu beroperasi sesuai perhitungan kinematika dan dinamika.

Perencanaan trayektori serta implementasi PID pada servo 360° memberikan kontrol yang stabil, meskipun terdapat keterbatasan teknis seperti keharusan servo kembali ke posisi awal sebelum menuju sudut tertentu. Meskipun demikian, robot menunjukkan performa yang konsisten dalam mengangkat dan memindahkan objek berbentuk silinder menggunakan end-effector elektromagnetik.

Penelitian ini memberikan kontribusi nyata dalam meningkatkan efisiensi ruang, mempercepat proses penyortiran, dan mengurangi risiko kecelakaan kerja di gudang. Namun, beberapa keterbatasan, seperti sensitivitas sistem terhadap cahaya dan durasi penggunaan elektromagnet, memerlukan perhatian lebih lanjut. Dengan pengembangan tambahan, seperti peningkatan desain end-effector dan optimasi sistem deteksi visual, sistem ini memiliki potensi untuk diimplementasikan secara luas dalam industri pergudangan modern.

5.2 Saran

Dalam penelitian ini, tentu ada beberapa batasan masalah yang harus diberlakukan baik karena keterbatasan perlengkapan atau keterbatasan waktu. Pertama, barang-barang yang digunakan saat *testing* robot berbentuk silinder, sehingga orientasi barang tidak menjadi masalah. Kedua, dikarenakan panjang lengan robot yang tidak terlalu panjang, maka batas jangkauan robot menyesuaikan dengan kemampuan robot. Ketiga, dikarenakan sistem *machine learning* pada kamera yang bertujuan untuk mendeteksi posisi barang bekerja dengan cara “memisahkan” warna yang lebih terang dari latarnya yang berwarna hitam, sistem rawan mendeteksi “noise” yang bukan merupakan objek jika dijalankan dalam ruangan yang terang, sehingga cahaya harus diperhatikan saat praktek. Keempat, dikarenakan *end-effector* pada robot menggunakan elektromagnet, *end-effector* tidak boleh dinyalakan terlalu lama karena akan menyebabkan magnet menjadi panas dan kekuatan medan magnet menjadi berkurang. Kelima, masih berhubungan dengan magnet, posisi *end-effector* perlu dibantu agar menghadap ke bawah secara sempurna dikarenakan hanya digantung menggunakan kabel yang cukup kaku.

BAB VI REFERENSI

[1] Brosnan Property, “Warehouse Management: Top 6 Order Picking Problems and How to Solve Each,” 5 August 2023. [Online]. Available: https://www.globaltrademag.com/warehouse-management-top-6-order-picking-problems-and-how-to-solve-each/.

[2] Roberto Michel, “2018 Warehouse/Distribution Center Equipment Survey: Automation & Robotics Lead Robust Outlook,” Logistics Managements, 16 April 2018. [Online]. Available: https://www.logisticsmgmt.com/article/2018_warehouse_distribution_center_equipment_survey_automation_robotics_1#:~:text=For%20manufacturing%20respondents,expect%20a%20decrease..

[3] Iris Dynamics, “Forward and Inverse Kinematics: Explained,” Iris Dynamics, 12 April 2023. [Online]. Available: https://irisdynamics.com/articles/forward-and-inverse-kinematics.

[4] M. Toussaint, “Robotics Dynamics,” University of Stuttgart, Stuttgart, 2015.

[5] R. Nilsson, “Iris Dynamics,” *Master's Thesis,* 2009.

[6] rosrobotics learning, “Jacobian,” rosrobotics learning, [Online]. Available: https://www.rosroboticslearning.com/jacobian.

[7] geeksforgeeks, “What is OpenCV Library?,” geeksforgeeks, 15 April 2024. [Online]. Available: https://www.geeksforgeeks.org/opencv-overview/.

[8] David Colson, “Exploring rectangle packing algorithms,” David Colson, 10 Maret 2020. [Online]. Available: https://www.david-colson.com/2020/03/10/exploring-rect-packing.html.

[9] ni, “The PID Controller & Theory Explained,” ni, 11 Desember 2024. [Online]. Available: https://www.ni.com/en/shop/labview/pid-theory-explained.html.

[10] V. Kanade, “What Is a Robot Operating System (ROS)? Meaning, Working, Applications, and Benefits,” spiceworks, 13 February 2024. [Online]. Available: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-robot-operating-system/#:~:text=In%20essence%2C%20ROS%20operates%20by,%2C%20simulation%2C%20and%20hardware%20integration..

[11] moveit, “Moveit Concepts,” moveit, [Online]. Available: https://moveit.ai/documentation/concepts/#concepts.

[12] R. S. D. D. P. ADI SUCIPTO, “Gerak Robot Berkaki Dua menggunakan ROS dan RViz sebagai Visualisasi Interaktif,” ELKOMIKAL Jurnal, [Online]. Available: https://ejurnal.itenas.ac.id/index.php/elkomika/article/view/3834.

[13] The Construct, “[ROS in 5 mins] 028 – What is Gazebo simulation?,” The Construct, [Online]. Available: https://www.theconstruct.ai/ros-5-mins-028-gazebo-simulation/.

[14] L. Santana, L. Goes dan M. Maximo, “Artigo Servo,” *Physical Modeling and Parameters Identification of the MG995 Servomotor,* 2021.

LAMPIRAN

1. Working Log Akhir (Recap)

Nama Anggota	Recap pengerjaan project
Willsan A Jantho	● Membuat sistem feedback pada servo 360 menggunakan rotary encoder sehingga servo 360 dapat berputar dengan akurat menuju ke sudut theta yang diinginkan. ● Melakukan Research dan kalkulasi untuk Dinamika pada robot agar dapat mengetahui torsi yang bekerja pada setiap sendi atau *joint.* ● Membuat *item* yang diangkat oleh robot, dengan membuat cylinder yang diberikan paperclip agar dapat diangkat oleh elektromagnet yang merupakan end effector. ● Membuat sistem komunikasi anatara phyton (machine learning) dengan arduino menggunakan pyserial (tidak jadi digunakan). ● Menyusun laporan akhir dan menuliskan bagian Robot Kinematics & Dynamics pada laporan.
Raphael Nazareth	● Membuat perhitungan 2link arm dengan input xyz dari robot ● Membuat coordinate system machine learning input & output dengan merapihkan titik (0.0) pada pengambilan menggunakan kamera dan penyimpanan barang menggunakan hyperpack ● Membuat electromagnet dengan baut dan kabel ● Menyusun trajectory planning linear interpolation ● RnD offset system koordinat robot agar 2 link bekerja dengan baik ● RnD Speed motor servo 360° 20kg ● Membuat simscape simulation Kinematics yang disambungkan dengan solidworks di matlab ● Membuat PPT presentasi akhir untuk Robot Kinematics & dynamics
Samuel Jordan Widjaja	● Melakukan riset awal terkait cara kerja rotary encoder ● Membuat sistem untuk menentukan angle servo 360 menggunakan phytagoras ● Turut berpartisipasi RnD mencari offset system koordinat robot & speed motor servo 360° 20kg ● Pengaplikasiah sistem kontrol ke motor servo 360° ● Turut berpartisipasi dalam pembuatan laporan
Efran Leonard Putra Satria	● 3D Model Robot ● Bantu hitungan Inverse Kinematics ● Trajectory Planning ● Coding Arduino

Alifito Marciano Camil	● Membuat simulasi RKD di dalam ROS menggunakan package Moveit-Rviz dan Gazebo. ● Membantu dalam zoning dan RnD sedangkan penentuan offset base servo 360 supaya inverse kinematic solver akurat. ● Membantu dalam zoning dan RnD sedangkan penentuan offset 2-link arm untuk trajectory planning pergerakan arm. ● Membantu dalam konversi Inverse Kinematics Solver dari Python ke C++ supaya bisa bekerja di dalam Arduino. ● Membuat solver Inverse Kinematics pertama untuk bagian 2-link arm robot output hanya theta 1 dan 2 (tidak digunnakan di prototype akhir). ● Membantu edit kode Machine Learning di dalam python supaya output koordinat dapat di kontrol user dengan menekan tombol enter. ● Menulis bagian terkait dengan Machine Learning di dalam laporan tugas akhir.