Tugas 3 Sistem Robot Otonom: Simulasi Odometri Wheeled Mobile Robot Pioneer P3DX Menggunakan Coppelia Sim dan Bahasa Pemrograman Python Dengan Perantara ZMQ Remote API

Nama: Falah Amru Dikasmara
NRP: 5022211041
Mata Kuliah: Sistem Robot Otonom

1 Pendahuluan

1.1 Latar Belakang

Proyek ini adalah simulasi odometri pada wheeled mobile robot Pioneer P3DX yang dibuat menggunakan perangkat lunak coppelia sim edu dan bahasa pemrograman python. Dalam proyek ini, Program python berkomunikasi dengan perangkat lunak copelliasim menggunakan ZMQ Remote API. Proyek ini bertujuan sebagai sarana edukasi dalam memahami cara kerja metode odometri dalam memperkirakan posisi dan orientasi robot berdasarkan informasi dari gerakan roda berupa kecepatan sudut roda.

1.2 Tujuan

Adapun tujuan dari proyek ini adalah :

1. Mempelajari cara kerja metode odometri dalam memperkirakan posisi dan orientasi robot berdasarkan informasi dari gerakan roda berupa kecepatan sudut roda
2. Mempelajari cara mengimpelmentasikan metode odometri menggunakan bahasa pemrograman python.
3. Merancang program python yang dapat memperkirakan posisi dan orientasi robot berdasarkan informasi dari gerakan roda dan menampilkannya dalam bentuk grafik.
4. Membandingkan perkiraan posisi dan pose dari metode odometri terhadap posisi dan pose sebenarnya pada scene simulasi.
5. Membuat laporan sebagai bahan pembelajaran mahasiswa

1.3 Fitur Proyek

Pada proyek simulasi metode odometri ini, program diharapkan dapat secara otomatis membaca beberapa informasi dari scene simulasi yang dibutuhkan dalam melakukan metode odometri seperti jari-jari roda dan jarak antara pusat roda terhadap pusat robot, mencatat posisi dan pose sebenarnya dari robot selama simulasi berlangsung, melakukan proses odometri dan mencatat hasil perkiraan selama simulasi berlangsung, dan menampilkan posisi dan pose robot sebenarnya dan hasil perkiraan odometri dalam grafik yang sama saat simulasi dihentikan.

1.4 Software yang Dibutuhkan dan Fungsinya

Adapun beberapa software yang dibutuhkan untuk membuat proyek ini antara lain :

1. Python : Sebagai bahasa pemrograman dan interpreter
2. Coppelia Sim Edu : Sebagai software untuk membuat lingkungan simulasi yang terdiri atas robot P3DX dan dinding pembatas.
3. ZMQ Remote API : untuk membaca informasi dan memberi perintah pada lingkungan simulasi menggunakan program python.

1.5 Tantangan

Dalam pengerjaan proyek ini, terdapat beberapa tantangan yang dihadapi, antara lain :

1. Menentukan algortima yang tepat agar program dapat memenuhi tujuan proyek.
2. Menentukan perintah API CoppeliaSim yang tepat untuk melakukan tugas tertentu.

1.6 Kemampuan yang Dipelajari

1. Penggunanan CoppeliaSim untuk membuat lingkungan simulasi sederhana.
2. Pemrograman dalam bahasa python dan penggunaan ZMQ Remote API untuk membaca data dan memberi perintah pada lingkungan simulasi.
3. Cara kerja metode odometri dan cara mengimpelmentasikan metode tersebut menggunakan bahasa python.
4. Penggunaan GitHub untuk dokumentasi proyek.

2 Dasar Teori

2.1 Python

Python merupakan bahasa pemrograman yang diciptakan Guido van Rossum dan pertama kali dirilis pada tahun 1991. Python digolongkan sebagai bahasa pemrograman tingkat tinggi karena cara penulisannya lebih dekat dengan bahasa manusia dan lebih jauh dari bahasa mesin, sehingga memberikan beberapa kelebihan. Pengguna python tidak perlu mengatur detail teknis yang berhubungan dengan perangkat keras seperti alamat memori, register CPU, atau instruksi mesin, sehingga membuat baris perintah python lebih ringkas daripada bahasa pemrograman lain untuk tugas yang sama. Sintaks pada python juga mudah dipahami karena mirip dengan bahasa inggris. Python dapat dijalankan pada berbagai sistem operasi karena memiliki interpreter yang dapat menerjemahkan kode ke instruksi yang sesuai dengan sistem yang dipakai. Berbagai kelebihan tersebut membuat penggunaan python sangat populer bagi banyak orang dan mendorong dikembangkannya berbagai pustaka dan modul siap pakai untuk berbagai tugas, sehingga memungkinkan programmer fokus ke logika aplikasi daripada detail implementasi rendah.

2.2 CoppeliaSim

CoppeliaSim adalah sebuah perangkat lunak simulasi robotika yang banyak digunakan di bidang penelitian, pendidikan, dan pengembangan sistem otonom. Aplikasi ini memungkinkan pengguna membuat model robot, sensor, aktuator, serta lingkungan virtual yang realistis, kemudian menguji algoritma kendali atau kecerdasan buatan secara aman tanpa perlu langsung menggunakan perangkat keras. CoppeliaSim dilengkapi dengan mesin fisika untuk mensimulasikan dinamika gerakan, benturan, serta interaksi antar objek. Selain itu, software ini mendukung berbagai antarmuka pemrograman (API) seperti Python, C++, Java, MATLAB, dan ROS (Robot Operating System), sehingga memudahkan integrasi dengan proyek robotika nyata. Karena fleksibilitas dan kelengkapan fiturnya, CoppeliaSim menjadi salah satu platform simulasi robot paling populer untuk eksperimen akademik maupun industri sebelum implementasi di dunia nyata.

2.3 Application Programming Interface (API)

Application Programming Interface (API) adalah suatu antarmuka yang menyediakan sekumpulan aturan, fungsi, dan protokol yang memungkinkan satu perangkat lunak berkomunikasi dengan perangkat lunak lainnya. API berfungsi sebagai jembatan penghubung antara aplikasi dengan sistem operasi, pustaka, layanan, atau aplikasi eksternal sehingga pengembang tidak perlu memahami detail internal dari sistem yang digunakan. Dengan API, pertukaran data dan instruksi dapat dilakukan secara standar melalui mekanisme request dan response. Dalam praktiknya, API banyak digunakan, misalnya pada aplikasi cuaca yang mengambil data dari server penyedia layanan meteorologi. Keberadaan API mempermudah proses pengembangan perangkat lunak karena memungkinkan pemanfaatan fungsi dan layanan yang sudah ada, mendukung interoperabilitas antar sistem, serta mempercepat implementasi teknologi baru.

2.4 ZMQ Remote API

ZMQ Remote API adalah antarmuka komunikasi pada perangkat lunak CoppeliaSim yang dibangun di atas teknologi ZeroMQ (ZMQ) untuk memungkinkan interaksi antara simulasi dengan program eksternal. Melalui API ini, CoppeliaSim dapat dijalankan sebagai server yang menerima perintah dari program client yang ditulis dalam bahasa pemrograman seperti Python, C++, Java, atau MATLAB. Mekanisme kerjanya berbasis pertukaran pesan (message-based communication) yang bersifat asinkron, sehingga proses pengiriman instruksi dan penerimaan data dapat berjalan cepat, efisien, serta lintas platform. Dengan adanya ZMQ Remote API, pengguna tidak hanya dapat memulai dan menghentikan simulasi, tetapi juga mengontrol objek, menggerakkan robot, membaca sensor, dan melakukan integrasi dengan sistem lain seperti Robot Operating System (ROS). Keunggulan utama pendekatan ini adalah pemisahan antara logika kendali robot yang dijalankan di luar simulasi dengan lingkungan virtual yang ada di dalam CoppeliaSim, sehingga memberikan fleksibilitas tinggi, kemudahan integrasi, serta kondisi eksperimen yang lebih mendekati penerapan nyata pada sistem robotika.

2.5 Odometri Robot Mobile

Odometri merupakan metode untuk memperkirakan posisi dan orientasi (*pose*) robot berdasarkan informasi gerakan roda. Prinsip dasar odometri adalah mengintegrasikan kecepatan linier dan kecepatan sudut dari robot terhadap waktu, sehingga dapat diperoleh estimasi lintasan.

Pada robot *differential–drive*, odometri dihitung dari kecepatan sudut roda kanan (ω_r) dan kiri (ω_l) dengan jari–jari roda R serta setengah jarak antar roda L_{half}:

$$v_r = \omega_r R, \qquad v_l = \omega_l R$$

$$v = \frac{v_r + v_l}{2}, \qquad \omega = \frac{v_r - v_l}{2L_{\text{half}}}$$

dengan v adalah kecepatan translasi robot dan ω adalah kecepatan rotasi (yaw rate). Pose robot (x, y, θ) kemudian diperbarui dengan integrasi:

$$\begin{aligned} x(t + \Delta t) &= x(t) + v \cos\theta \, \Delta t, \\ y(t + \Delta t) &= y(t) + v \sin\theta \, \Delta t, \\ \theta(t + \Delta t) &= \theta(t) + \omega \, \Delta t. \end{aligned}$$

2.6 Ground Truth dan Proyeksi ke Kerangka Lokal

Dalam simulasi CoppeliaSim, *ground truth* (GT) adalah posisi dan orientasi aktual robot di dalam kerangka dunia (world frame). Agar dapat dibandingkan langsung dengan odometri (yang selalu diasumsikan mulai dari $(0, 0, 0)$), maka pose GT perlu diproyeksikan ke kerangka lokal awal robot. Transformasi ini dilakukan dengan translasi dan rotasi menggunakan matriks rotasi $R(-\theta_0)$, di mana θ_0 adalah orientasi awal robot:

$$\begin{bmatrix} x_{gt} \\ y_{gt} \end{bmatrix} = \begin{bmatrix} \cos\theta_0 & \sin\theta_0 \\ -\sin\theta_0 & \cos\theta_0 \end{bmatrix} \begin{bmatrix} x_w - x_0 \\ y_w - y_0 \end{bmatrix},$$

$$\theta_{gt} = \theta_w - \theta_0.$$

2.7 Matriks Rotasi

Matriks rotasi dua dimensi digunakan untuk memutar suatu vektor dalam bidang xy. Secara umum, untuk rotasi dengan sudut θ searah jarum jam atau berlawanan arah jarum jam, matriks yang digunakan adalah:

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Dalam kasus program ini, digunakan variasi matriks rotasi yang ekuivalen dengan $R(-\theta_0)$, yaitu:

$$\begin{bmatrix} \cos\theta_0 & \sin\theta_0 \\ -\sin\theta_0 & \cos\theta_0 \end{bmatrix}.$$

Fungsi utamanya adalah untuk mengubah koordinat dari kerangka dunia (world) menjadi koordinat relatif terhadap kerangka awal robot (local).

2.8 Normalisasi Sudut

Karena operasi integrasi dan pengurangan sudut dapat menghasilkan nilai di luar rentang $[-\pi, \pi)$, maka digunakan fungsi pembungkus sudut:

$$\theta_{\text{wrap}} = \text{atan2}(\sin\theta, \cos\theta).$$

Dengan cara ini, nilai orientasi robot selalu konsisten dalam rentang $[-180°, 180°)$.

2.9 Root Mean Square Error (RMSE)

Untuk mengukur perbedaan antara lintasan hasil odometri dengan lintasan ground truth, digunakan metrik RMSE. RMSE memberikan nilai rata-rata kuadrat error yang diakarkan:

$$\text{RMSE}_{xy} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}((x_o(k) - x_{gt}(k))^2 + (y_o(k) - y_{gt}(k))^2)}.$$

Semakin kecil nilai RMSE, semakin akurat estimasi posisi odometri dibandingkan dengan ground truth.

3 Alur Kerja Program Simulasi Odometri

Program ini dibuat untuk mensimulasikan dan mengevaluasi sistem odometri pada robot differential–drive (dalam hal ini Pioneer P3DX) dengan bantuan *CoppeliaSim* dan *Python*. Program ini memiliki beberapa kegunaan utama:

- Menghitung lintasan odometri berdasarkan model kinematika robot dan data kecepatan roda kanan–kiri yang dibaca dari simulator.
- Mengambil data *ground truth* (GT) posisi dan orientasi robot langsung dari CoppeliaSim untuk dijadikan acuan pembanding.
- Memproyeksikan lintasan GT ke dalam kerangka lokal awal robot, sehingga dapat dibandingkan langsung dengan hasil odometri.
- Menyediakan visualisasi dalam bentuk kurva $x(t)$, $y(t)$, yaw(t), lintasan x–y, serta kurva galat $e_x(t)$, $e_y(t)$, dan $e_\theta(t)$.
- Menghitung metrik kuantitatif berupa galat posisi (RMSE) dan galat orientasi maksimum.

Fitur ini bermanfaat baik untuk tujuan edukasi dalam memahami cara kerja odometri dan perbandingan dengan data simulasi.

Alur kerja program dapat dirangkum sebagai berikut:

1. **Inisialisasi dan Konfigurasi**
   (a) Mengimpor pustaka: `math`, `time`, `matplotlib.pyplot`, dan `RemoteAPIClient`.
   (b) Menentukan path objek di scene CoppeliaSim: basis robot, joint kanan, joint kiri, dan bentuk roda kanan.
   (c) Menetapkan mode `STEPPING` untuk mengontrol apakah Python yang memajukan simulasi (`sim.step()`) atau simulasi berjalan otomatis.
2. **Fungsi Utilitas**
   (a) `wrap_pi(a)`: membungkus sudut ke interval $[-\pi, \pi)$ menggunakan atan2$(\sin a, \cos a)$ untuk menghindari discontinuitas sudut.
   (b) `read_params_from_scene(sim)`: membaca parameter dari scene, meliputi:
      - R: jari–jari roda (dari diameter geometri).
      - L_{half}: setengah jarak antar roda (dari selisih posisi y joint kanan dan kiri).
      - dt_{scene}: time step simulasi.
      - Handle objek: joint kanan, joint kiri, dan basis.
   (c) `get_gt_pose2d(sim, base_h)`: mengambil pose ground truth dalam world frame (x, y, θ).
3. **Loop Odometri dan Ground–Truth**

(a) Ambil pose awal GT (x_0, y_0, θ_0) dan hitung $\cos\theta_0$, $\sin\theta_0$ untuk membentuk matriks rotasi.

(b) Masuk loop hingga simulasi berhenti:

i. Periksa status simulasi:
- **STOP**: jika `state` = `simulation_stopped`, keluar loop.
- **PAUSE**: jika `state` mengandung bit `simulation_paused`, program tidur singkat lalu lanjut iterasi.
- **RUN**: jika simulasi aktif, waktu dimajukan dengan `sim.step()` (stepping) atau polling (non-stepping).

ii. Hitung $\Delta t = t_{\text{now}} - t_{\text{prev}}$. Jika $\Delta t \leq 0$, iterasi dilewati.

iii. Baca kecepatan sudut roda kanan (ω_r) dan kiri (ω_l).

iv. Hitung kecepatan linear roda: $v_r = \omega_r R,\ v_l = \omega_l R$.

v. Hitung kecepatan translasi dan angular robot:

$$v = \frac{v_r + v_l}{2}, \qquad \omega = \frac{v_r - v_l}{2L_{\text{half}}}.$$

vi. Integrasi odometri:

$$\begin{aligned} x_o &\leftarrow x_o + v\cos\theta_o\,\Delta t, \\ y_o &\leftarrow y_o + v\sin\theta_o\,\Delta t, \\ \theta_o &\leftarrow \text{wrap\_pi}(\theta_o + \omega\,\Delta t). \end{aligned}$$

vii. Ambil GT dunia (x_w, y_w, θ_w), translasi relatif $(\Delta x, \Delta y)$, lalu proyeksikan ke frame lokal awal:

$$\begin{bmatrix} x_{gt} \\ y_{gt} \end{bmatrix} = \begin{bmatrix} \cos\theta_0 & \sin\theta_0 \\ -\sin\theta_0 & \cos\theta_0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}, \quad \theta_{gt} = \text{wrap\_pi}(\theta_w - \theta_0).$$

viii. Hitung error:

$$e_x = x_o - x_{gt}, \quad e_y = y_o - y_{gt}, \quad e_\theta = \text{wrap\_pi}(\theta_o - \theta_{gt}).$$

ix. Simpan histori waktu, ODO, GT, dan error.

4. **Plotting dan Evaluasi**

(a) Plot $x(t)$, $y(t)$, yaw(t), serta lintasan x–y untuk membandingkan ODO vs GT.

(b) Plot kurva error $e_x(t)$, $e_y(t)$, dan $e_\theta(t)$ pada subplot terpisah.

(c) Hitung metrik ringkas:

$$\text{RMSE}_{xy} = \sqrt{\frac{1}{N}\sum(e_x^2 + e_y^2)}, \qquad \max|e_\theta| = \max_k |e_\theta(k)|.$$

(d) Tampilkan grafik ke layar dan cetak metrik evaluasi di konsol.

4 Tutorial Penggunaan Program

1. **Persiapan**

(a) Pastikan **Python 3** sudah terpasang.

(b) Instal pustaka yang diperlukan:
- `pip install coppeliasim-zmqremoteapi-client`
- `pip install matplotlib`

(c) Siapkan **CoppeliaSim** dan pastikan bisa dijalankan.

2. **Penempatan File**

(a) Simpan **file scene** CoppeliaSim, **program Python**, dan **API ZMQ Remote** dalam satu folder agar mudah diakses.

(b) Pastikan path objek di scene sesuai dengan yang digunakan dalam program:
- `/PioneerP3DX`
- `/rightMotor`
- `/leftMotor`

- `/rightMotor/rightWheel`

3. **Menjalankan Simulasi**
   (a) Buka CoppeliaSim, lalu muat scene robot Pioneer P3DX.
   (b) Jalankan program Python dari terminal atau code editor.
   (c) Program akan otomatis:
   - Menghubungkan ke CoppeliaSim.
   - Memulai simulasi.
   - Membaca parameter roda dan konfigurasi robot.
   - Menghitung odometri dan membandingkan dengan data ground truth.

   (d) Untuk menghentikan program, tekan **Stop** pada simulasi di CoppeliaSim. Hal ini membuat perulangan di program Python berhenti secara otomatis.

4. **Hasil yang Ditampilkan**
   (a) Setelah simulasi berhenti, program akan menampilkan dua jendela grafik:
   - **Jendela 1 (ODO vs GT)**:
     – Grafik $x(t)$ odometri vs ground truth.
     – Grafik $y(t)$ odometri vs ground truth.
     – Grafik yaw(t) (dalam derajat) odometri vs ground truth.
     – Lintasan x–y (trajectory) ODO vs GT.

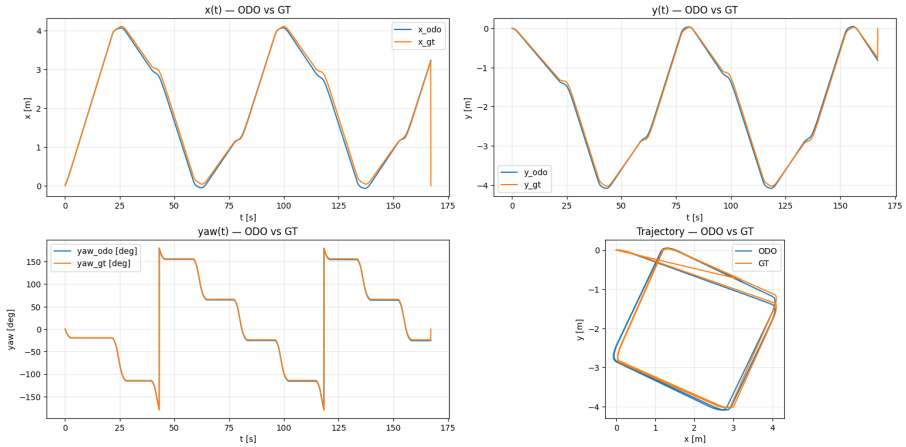

Figure 1: Odometri vs Ground Truth

   - **Jendela 2 (Error)**:
     – Kurva error $e_x(t)$.
     – Kurva error $e_y(t)$.
     – Kurva error $e_\theta(t)$ (dalam derajat).

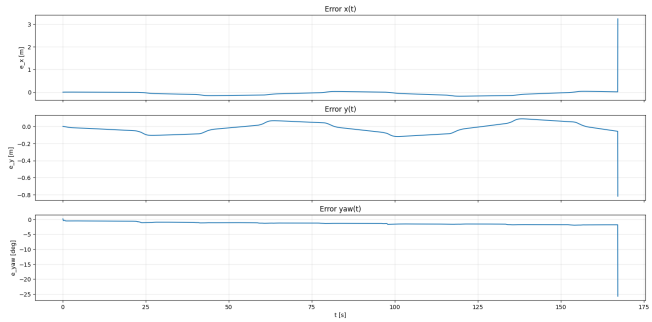

Figure 2: Error

   (b) Di konsol juga akan dicetak metrik ringkas:
   - RMSE posisi (RMSE$_{xy}$).
   - Galat orientasi maksimum (max $|e_\theta|$).

5 Kesimpulan

Berdasarkan implementasi dan pengujian yang telah dilakukan, dapat disimpulkan bahwa:

1. Program berhasil melakukan **koneksi** antara Python dan CoppeliaSim menggunakan ZMQ Remote API, sehingga proses simulasi dapat dikendalikan secara langsung dari kode Python.
2. Program mampu **membaca parameter fisik robot** (jari–jari roda, jarak antar roda, dan time step simulasi) secara otomatis dari scene CoppeliaSim.
3. Perhitungan **odometri differential–drive** dapat dijalankan secara real-time dengan integrasi numerik yang konsisten terhadap waktu simulasi.
4. Program berhasil mengambil data **ground truth** (GT) posisi dan orientasi robot dari CoppeliaSim, kemudian memproyeksikannya ke kerangka lokal awal untuk dibandingkan langsung dengan hasil odometri.
5. Visualisasi yang ditampilkan berupa **perbandingan lintasan ODO vs GT** serta **kurva galat** menunjukkan bahwa program dapat mengevaluasi performa odometri secara kuantitatif.

Secara keseluruhan, program ini telah berfungsi dengan baik untuk mensimulasikan, menghitung, dan mengevaluasi odometri robot differential–drive pada lingkungan CoppeliaSim, serta memberikan keluaran grafik dan metrik yang mendukung analisis performa.

- Link Github :

  https://github.com/amrufal/Odometri_Coppeliasim

- Linkedin :

  https://www.linkedin.com/posts/falah-amru-9a192a385_github-amrufalodometricoppeliasim-activity-737
  utm_source=share&utm_medium=member_desktop&rcm=ACoAAF7_5wkBHEm5eAEQ0al5sPz8ty6cgqgTe08