Muhammad Fauzan Nur'ilham

1103204085

Robotika

TK-44-G7

Path Planning 8-19

Video 8

Pada robot ini, digunakan sensor posisi untuk membandingkan pembacaan saat ini dan pembacaan sebelumnya guna menentukan jarak linier yang telah ditempuh dari titik awalnya di grid sel 16 dimulai dari koordinat 15.0, -15.0. Dengan demikian, robot dapat melacak sejauh apa ia telah bergerak dari titik awal untuk mengetahui koordinatnya dan menentukan kotak mana yang sedang dihuni. Ini memungkinkan robot untuk terus bergerak dan mengunjungi setiap kotak.

Dalam kodingan, terdapat implementasi kontrol robot dalam lingkungan simulasi menggunakan Webots. Ini mencakup kontrol gerakan dan navigasi sederhana untuk robot yang dilengkapi dengan sensor jarak, sensor posisi roda, kamera, dan unit inersia (IMU).

Berikut adalah penjelasan singkat beberapa bagian kodingan:

1. **Inisialisasi dan Pengaturan Robot:**
    - Mendeklarasikan dan menginisialisasi variabel seperti timestep, sensor jarak, sensor posisi roda, kamera, IMU, dan motor roda.
    - Menetapkan konstanta dan parameter robot seperti radius roda, sirkumferensi roda, unit encoder, kecepatan maksimum, dan jarak antar roda.
2. **Update Fungsi Sensor dan Kontrol Gerakan:**
    - Fungsi **get_p_sensors_vals** dan **get_d_sensors_vals** mengembalikan nilai sensor posisi roda dan jarak dalam satuan inci.
    - Fungsi **move** menggerakkan robot sejumlah inci tertentu menggunakan motor roda.
3. **Fungsi Rotasi dan Pemutaran:**
    - Fungsi **rotate** memungkinkan robot berputar sejumlah derajat dalam waktu tertentu.
    - Fungsi **turn_left** dan **turn_right** digunakan untuk memutar robot ke kiri atau kanan.
4. **Fungsi Pemantauan dan Pembaruan Robot:**
    - Fungsi **update_robot** memantau dan memperbarui status robot, termasuk posisi, orientasi, dan sel terkait navigasi.
    - Fungsi **get_robot_x_y** menghitung posisi baru robot berdasarkan perbedaan sensor posisi roda.
    - Fungsi **get_current_grid_cell** menentukan sel grid saat ini berdasarkan posisi x dan y.

- Fungsi **update_direction** mengupdate arah hadap robot berdasarkan nilai sensor IMU.

5. **Navigasi ke Sel Tertentu:**

    - Fungsi **move_to_cell** digunakan untuk memindahkan robot ke sel tertentu dalam grid dengan mempertimbangkan posisi relatifnya dan mengatur arah hadapnya.

    - Fungsi **main** menggunakan serangkaian pergerakan dan navigasi untuk memindahkan robot ke sel yang ditentukan.

Kodingan ini memanfaatkan fungsi dan variabel untuk menggerakkan robot dalam grid sel, di mana robot berpindah dari satu sel ke sel berikutnya. Setiap sel yang dikunjungi ditandai dengan 'X' pada array **visited_cells**. Tujuannya bisa beragam, seperti menjelajahi seluruh grid atau mencapai sel tertentu sesuai dengan skenario simulasi.

Video 9

Sama seperti tutorial sebelumnya, robot ini juga mengalami kebisingan pada unit pengukuran inersia dan sensor jaraknya. Untuk mengatasi hal ini, diperlukan solusi agar nilai koordinat x dan y dapat ditambahkan berdasarkan arah yang dituju. Sebagai solusi, saya menggunakan kamus Python, **dirs = {"North": 0, "West": 0, "East": 0, "South": 0}**. Setiap langkah gerak akan menambahkan arah sesuai dengan nilai IMU, dan saya mengasumsikan bahwa robot bergerak ke arah dengan hitungan tertinggi. Pendekatan ini memecahkan masalah arah yang dituju dari satu sel ke sel lainnya dan dengan demikian menyelesaikan masalah koordinatnya juga. Menyetel ulang kamus untuk setiap arah baru yang dicoba juga penting untuk mencegah estimasi arah yang miring.

Meskipun kedua kodingan hampir sama, ada beberapa perbedaan di antara keduanya, seperti:

1. **Kemiripan:**

    - Keduanya memiliki fungsionalitas serupa untuk pergerakan robot, pembacaan sensor, dan navigasi berbasis grid.

    - Struktur keseluruhan kode, seperti mengimpor pustaka, menginisialisasi robot, dan mendefinisikan fungsi, mirip.

2. **Perbedaan:**

    - Ada perbedaan dalam nama variabel, komentar, dan detail khusus dalam fungsi-fungsi tersebut, tetapi logika inti dan struktur sepertinya tetap.

    - Potongan kode kedua memperkenalkan fitur tambahan seperti navigasi berbasis grid dan penandaan sel (dikunjungi atau tidak) yang tidak ada dalam potongan pertama.

Secara ringkas, kedua potongan kode tersebut memiliki struktur dan tujuan yang sama, tetapi terdapat perbedaan dalam detail spesifik dan fitur tambahan yang diperkenalkan pada potongan kode kedua. Potongan kode kedua memperkenalkan beberapa fitur tambahan yang signifikan untuk mendukung navigasi dan pemetaan posisi robot berdasarkan grid dalam lingkungan simulasi yang lebih kompleks.

Pertama-tama, navigasi berbasis grid diimplementasikan dengan membagi lingkungan menjadi sel-sel, dan robot melacak pergerakannya antara sel-sel ini. Selain itu, fitur penandaan sel muncul dengan penggunaan array **visited_cells**, yang memungkinkan robot untuk menandai sel mana yang sudah dikunjungi dan sel mana yang belum. Terdapat pula penyesuaian orientasi dan rotasi, di mana fungsi-fungsi seperti **turn_left** dan **turn_right** serta **rotate** digunakan untuk mengatur perubahan orientasi dan rotasi robot.

Navigasi ke sel berikutnya diimplementasikan melalui fungsi **move_to_cell**, yang memandu robot ke sel berikutnya berdasarkan grid. Selama proses ini, status grid terus dipantau, dan setiap pergerakan robot diiringi dengan penandaan status sel yang sudah dikunjungi. Struktur data tambahan, seperti **n_rc** yang mencocokkan indeks dan kolom grid, digunakan untuk memetakan posisi robot ke grid sel. Secara keseluruhan, potongan kode kedua ditambahkan dengan fitur-fitur ini untuk menangani tugas-tugas navigasi yang melibatkan pemantauan posisi robot dalam konteks grid pada simulasi yang lebih kompleks.

Video 10

Pada tutorial ini, skrip Webots menunjukkan cara menggunakan trilaterasi di setiap sel untuk menentukan lokasi robot di peta grid. Robot tersebut melakukan rotasi 360 derajat di setiap sel untuk mendeteksi keempat silinder dan menggunakan sistem persamaan dan trilaterasi untuk menentukan posisi robot di peta grid.

Beberapa fitur tambahan pada potongan kode kedua termasuk:

1. **Navigasi Berbasis Grid:**
    - Terdapat variabel **visited_cells** yang mencatat sel mana yang telah dikunjungi.
    - Pembaruan status sel (**'.'** untuk belum dikunjungi, **'X'** untuk sudah dikunjungi) dilakukan saat robot bergerak melalui fungsi **move_to_cell**.
2. **Penandaan Sel:**
    - Fungsi **move_to_cell** ditambahkan untuk menggerakkan robot menuju sel tertentu dan memperbarui penandaan sel yang telah dikunjungi di **visited_cells**.
3. **Trilateration:**
    - Potongan kode kedua memperkenalkan fungsi-fungsi terkait trilaterasi (**full_rotate**, **get_abcdef**, **trilateration**) untuk mengestimasi posisi robot berdasarkan deteksi objek berwarna menggunakan kamera.
    - Objek berwarna (silinder kuning, merah, hijau, biru) dikenali, dan jaraknya digunakan untuk melakukan trilaterasi.
4. **Navigasi ke Sel Berikutnya:**
    - Potongan kode kedua memiliki loop utama (**main**) yang memberikan robot tugas untuk mengunjungi setiap sel dalam urutan tertentu.

- Robot akan menghentikan tugasnya jika seluruh sel telah dikunjungi.

5. **Fungsi-Fungsi Tambahan:**
    - Fungsi-fungsi seperti **update_robot**, **get_current_grid_cell**, **update_direction**, **print_visited_cells**, dan lainnya ditambahkan untuk memantau status robot, mengupdate posisi, dan menyediakan informasi tambahan.
6. **Kontrol Gerak:**
    - Fungsi-fungsi seperti **move**, **stop_motors**, **rotate**, **turn_left**, dan **turn_right** memberikan kontrol gerak pada robot untuk mencapai tujuan.
7. **Sensor dan Aktuator:**
    - Inisialisasi dan penggunaan sensor jarak (**front_ds**, **left_ds**, **right_ds**), sensor posisi (**left wheel sensor**, **right wheel sensor**), kamera (**camera1**), dan motor (**left wheel motor**, **right wheel motor**) ditambahkan pada potongan kode kedua.

Potongan kode kedua ini mengintegrasikan beberapa fitur tambahan untuk mendukung navigasi robot dalam lingkungan simulasi yang lebih kompleks. Selain itu, trilaterasi digunakan untuk memperkirakan posisi robot berdasarkan deteksi objek berwarna, yang merupakan peningkatan signifikan dalam fungsionalitas robot dalam pemetaan grid.

Video 11

Pada kodingan ini, terdapat tambahan fitur Webots yang memperkenalkan noise pada kedua sensor jarak, membuatnya lebih sulit bagi robot untuk memperkirakan seberapa jauh jaraknya dari setiap silinder. Untuk mengatasi ini, digunakan jarak relatif kamera yang diratakan dengan nilai sensor jarak depan, kemudian dirata-ratakan untuk setiap silinder guna mendapatkan radius dari robot ke setiap silinder. Hal ini meningkatkan akurasi pembacaan sensor.

Berikut adalah beberapa kelebihan fitur pada kodingan ini dibandingkan dengan tutorial sebelumnya:

1. **Navigasi Berbasis Grid dan Posisi Robot:**
    - Penggunaan variabel **robot_pose** untuk menyimpan posisi robot dalam koordinat global (x, y), nomor sel (n), dan orientasi (theta).
    - Pembaruan posisi robot dilakukan dalam fungsi **update_robot**, yang mencakup pembacaan sensor dan trilateration untuk perkiraan posisi.
2. **Sensor dan Aktuator:**
    - Inisialisasi dan penggunaan sensor jarak (**front_ds**, **left_ds**, **right_ds**), sensor posisi (**left wheel sensor**, **right wheel sensor**), kamera (**camera1**), dan motor (**left wheel motor**, **right wheel motor**).
    - Pembaruan nilai sensor dan pose robot dilakukan dalam fungsi-fungsi seperti **print_measurements**, **get_p_sensors_vals**, **get_d_sensors_vals**, dan **update_robot**.

3. **Trilaterasi:**
    - Fungsi-fungsi **full_rotate**, **get_abcdef**, dan **trilateration** digunakan untuk melakukan trilaterasi berdasarkan deteksi objek berwarna (cilinder kuning, merah, hijau, biru) menggunakan kamera.
    - Perkiraan posisi robot diperbarui dengan melakukan trilaterasi dalam fungsi **update_robot** ketika parameter **trilat=True**.
4. **Navigasi ke Sel Berikutnya:**
    - Fungsi **move_to_cell** mengatur pergerakan robot ke sel berikutnya berdasarkan nomor sel tujuan dengan pertimbangan orientasi dan posisi saat ini.
5. **Penandaan Sel dan Pemantauan Sel yang Dikunjungi:**
    - Penggunaan list **visited_cells** untuk memantau sel mana yang telah dikunjungi (ditandai dengan 'X').
    - Fungsi **print_visited_cells** untuk mencetak status sel yang telah dikunjungi.
6. **Rotasi dan Kontrol Gerak:**
    - Fungsi-fungsi **rotate**, **turn_left**, dan **turn_right** digunakan untuk mengatur rotasi robot.
    - Kontrol gerak melibatkan pengaturan kecepatan motor dalam fungsi-fungsi **move** dan **stop_motors**.
7. **Penghentian Otomatis:**
    - Fungsi **check_if_robot_should_stop** memeriksa apakah seluruh sel sudah dikunjungi, dan jika ya, program dihentikan.
8. **Optimisasi Perjalanan:**
    - Dalam fungsi **main**, algoritma dipertimbangkan untuk mempercepat perjalanan robot dengan melompati sel yang telah dikunjungi sebelumnya.

Secara keseluruhan, kodingan ini menunjukkan peningkatan dalam pemantauan, navigasi, dan estimasi posisi robot, menjadikannya lebih sesuai untuk tugas simulasi di lingkungan grid dengan elemen tambahan seperti trilaterasi dan kontrol yang lebih canggih.

Video 12

Pada tutorial ini, disediakan skrip yang memandu robot melalui labirin dengan menggunakan algoritma dan filter partikel. Robot mampu mendeteksi dinding di sekitarnya, melakukan estimasi pengukuran pada setiap partikel dalam setiap sel, dan menggerakkan dirinya berdasarkan bobot sel setelah dinormalisasi. Beberapa kelebihan fitur pada kode baru dibandingkan dengan kode sebelumnya dapat diidentifikasi:

1. **Particle Filter Implementation:**

- Kode mengimplementasikan filter partikel untuk estimasi posisi robot. Filter partikel memanfaatkan metode statistik untuk memodelkan ketidakpastian, memberikan perkiraan yang lebih akurat tentang posisi sejati robot.
- Partikel digunakan untuk memodelkan variasi posisi dan orientasi robot dalam lingkungan.

2. **Penanganan Gerakan dan Pembaruan Posisi:**
    - Kode efektif mengelola gerakan robot dan pembaruan posisi berdasarkan pembacaan dari sensor posisi roda dan sensor inersia.
    - Robot dipindahkan dan diputar sesuai dengan gerakan fisiknya, sementara posisi dan orientasi robot diperbarui.
3. **Pemodelan Sensor dan Pengukuran:**
    - Pemodelan sensor jarak diimplementasikan dengan menambahkan noise pada pembacaan sensor, mensimulasikan ketidakpastian dalam pengukuran.
    - Pengukuran dari sensor jarak digunakan untuk mengestimasi probabilitas keberadaan dinding di sekitar robot.
4. **Visualisasi Data:**
    - Kode memberikan informasi terstruktur, termasuk pemetaan lingkungan, status sel yang dikunjungi, dan informasi partikel filter.
    - Visualisasi data membantu pemahaman yang lebih baik tentang interaksi robot dengan lingkungan.
5. **Resampling Partikel:**
    - Algoritma resampling partikel digunakan untuk mengatasi masalah degenerasi, memberikan bobot pada partikel yang memiliki probabilitas yang lebih signifikan.
    - Partikel diperbarui secara dinamis berdasarkan pergerakan dan pengukuran, meningkatkan akurasi estimasi.
6. **Implementasi Algoritma Gerak dan Pembaruan:**
    - Kode menyertakan algoritma untuk menggerakkan robot dan memperbarui posisi, mempertimbangkan noise gerakan dan ketidakpastian sensor.
    - Pembaruan berbasis pengukuran dan gerakan dilakukan dengan mempertimbangkan lingkungan sekitar robot.
7. **Optimisasi dan Pembersihan Kode:**
    - Kode telah dioptimalkan dan dibersihkan untuk memperjelas alur logika, membuatnya lebih mudah dipahami.

Fitur-fitur ini membuat kode lebih canggih dan efektif dalam memodelkan pergerakan serta estimasi posisi robot di dalam lingkungan labirin.

Video 13

Pada tutorial ini, disediakan skrip yang menggunakan filter partikel dengan 80 partikel untuk membimbing robot dalam menyelesaikan labirin. Algoritma yang komprehensif digunakan untuk menilai dinding-dinding di sekitar robot, mengidentifikasi langkah-langkah yang tersedia (kiri, depan, atau kanan), dan melakukan estimasi pengukuran pada partikel berdasarkan dinding-dinding yang diperoleh. Skrip kemudian menghitung bobot sel, mengnormalisasi mereka, dan meresampel partikel sesuai dengan bobot yang dinormalisasi tersebut. Robot bergerak secara strategis, memberi prioritas pada belokan ke kiri, diikuti oleh gerakan ke depan, dan kemudian belokan ke kanan. Jika tidak ada opsi yang tersedia, robot berbelok ke selatan, dan selanjutnya, ke timur, untuk menavigasi labirin. Model pengukuran memperkenalkan faktor kebisingan sebesar 25% pada pembacaan sensor, sementara model gerakan mencakup probabilitas pergerakan ke depan sebesar 90% dan kemungkinan tetap diam sebesar 10% ketika pergerakan dimungkinkan. Proses ini diulang hingga setiap sel dalam labirin dikunjungi, menunjukkan algoritma yang tangguh untuk tugas penyelesaian labirin.

Beberapa fitur utama yang diimplementasikan dalam kodingan ini melibatkan:

1. **Penanganan Informasi Posisi dan Orientasi:**
    - Kodingan memadukan informasi posisi dan orientasi robot dalam sel grid labirin.
2. **Penggunaan Partikel untuk Estimasi Posisi Robot:**
    - Partikel digunakan sebagai representasi kemungkinan posisi robot dalam labirin.
3. **Pemrosesan Data Sensor dan Kontrol Robot:**
    - Algoritma resampling digunakan untuk memperbarui partikel berdasarkan hasil pengukuran sensor dan kontrol robot.
4. **Pengukuran Noise, Model Pergerakan Robot, dan Pembaruan Matriks Labirin:**
    - Kodingan mencakup pengenalan noise pada pengukuran sensor, model pergerakan robot, dan pembaruan matriks labirin dengan informasi dinding yang dilihat oleh robot.

Semua fitur ini bekerja bersama untuk memungkinkan robot menjelajahi labirin, mengidentifikasi posisi dan orientasinya menggunakan filter partikel.

Video 14

Tutorial ini mengilustrasikan cara robot dapat melacak lokasinya di dunia dengan mengetahui lokasi awalnya dan membangun representasi dunia dengan mengumpulkan informasi setiap kali mengunjungi sel. Saat robot mengunjungi sel, ia mendapatkan informasi tentang dinding dan memperbarui susunan 2D sebelum membuat keputusan berikutnya. Informasi ini penting untuk

menentukan ketersediaan dinding di lokasi tertentu dan untuk menghindari gerakan yang tidak dapat dilakukan.

Kodingan ini merupakan implementasi kontroler untuk robot dalam simulasi labirin menggunakan Webots. Robot dilengkapi dengan sensor jarak, sensor posisi, kamera, dan unit inersia. Tujuannya adalah secara otonom memetakan labirin dan mencatat dinding-dinding yang ditemui selama eksplorasi. Kodingan menggabungkan inisialisasi robot dan sensor, konfigurasi parameter seperti radius roda, dan pemetaan labirin menggunakan beberapa fungsi. Prioritasnya adalah kunjungan ke sel-sel labirin yang belum pernah dikunjungi sebelumnya.

Fitur kodingan ini mencakup:

1. **Inisialisasi Robot dan Sensor:**
    - Pembuatan instance robot dan inisialisasi sensor-sensor.
    - Konfigurasi parameter seperti radius roda dan timestep simulasi.
2. **Visualisasi:**
    - Penggunaan kamera untuk mengambil informasi visual dari sekitar robot dan mengenali objek.
    - Aktivasi unit inersia (IMU) untuk mendapatkan informasi orientasi robot.
3. **Pengaturan Kecepatan Motor:**
    - Inisialisasi dan pengaturan kecepatan motor sesuai kebutuhan.
4. **Variabel dan Konstanta Robot:**
    - Penyimpanan variabel dan konstanta seperti radius roda dan kecepatan maksimum robot.
5. **Representasi Labirin:**
    - Labirin diwakili dalam bentuk matriks, dan konfigurasi labirin serta pemetaan menjadi bagian integral dari kodingan.
6. **Algoritma Pemetaan Labirin:**
    - Pemetaan labirin mengutamakan kunjungan ke sel-sel yang belum pernah dikunjungi sebelumnya.
7. **Fungsi-fungsi Utilitas:**
    - Konversi satuan, perhitungan waktu pergerakan, dan pemutakhiran posisi robot.
8. **Navigasi dan Kontrol Gerak Robot:**
    - Penanganan arah hadap robot dalam menghadapi empat arah utama: Utara, Barat, Selatan, dan Timur.

Semua fitur dan komponen disusun secara sistematis untuk mencapai tujuan utama: pemetaan labirin secara otonom.

Video 15

Tutorial ini mencakup pengendalian robot dalam simulasi labirin menggunakan Webots. Berikut adalah beberapa fitur utama dari program ini:

1. **Pemantauan Sensor:**
    - Program menggunakan berbagai sensor, termasuk sensor jarak (**DistanceSensor**), sensor posisi (**PositionSensor**), kamera (**Camera**), dan unit inersia (**InertialUnit**).
    - Informasi dari sensor-sensor ini digunakan untuk membuat keputusan terkait pergerakan dan pemetaan.
2. **Gerakan Robot:**
    - Robot dapat bergerak maju, berputar, dan menghentikan motor-motornya menggunakan instruksi seperti **move**, **rotate**, dan **stop_motors**.
    - Kecepatan dan durasi pergerakan diatur berdasarkan pengukuran sensor dan pengaturan tertentu.
3. **Pemetaan Labirin:**
    - Program berusaha memetakan labirin dengan mengenali tembok dan mengidentifikasi sel-sel yang telah dikunjungi.
    - Pemetaan dilakukan berdasarkan informasi dari sensor jarak dan orientasi robot.
4. **Navigasi Pintar:**
    - Program dirancang untuk melakukan navigasi yang efisien dan pintar di dalam labirin.
    - Faktor-faktor yang dipertimbangkan termasuk tembok yang ada, arah pergerakan robot, dan sel-sel yang telah dikunjungi.
5. **Orientasi Robot:**
    - Sensor unit inersia (IMU) digunakan untuk menentukan orientasi global robot.
    - Informasi ini digunakan untuk memperbarui posisi dan arah robot di dalam labirin.
6. **Penanganan Posisi dan Arah:**
    - Robot menyimpan informasi tentang posisi, orientasi, dan sel-sel yang telah dikunjungi.
    - Program berusaha memastikan bahwa robot dapat memetakan seluruh labirin dan menghindari jalur yang sudah dilalui sebelumnya.
7. **Pengendalian Alur Program:**

- Program menggunakan perulangan untuk terus beroperasi sampai kondisi tertentu terpenuhi, seperti pemetaan selesai atau tugas selesai.

Terdapat pula penekanan pada penggunaan kamus untuk mengetahui arah utama robot, dan program berusaha untuk membuat keputusan berdasarkan informasi sensor untuk mencapai tujuan pemetaan labirin secara efisien dan akurat.

Video 16

Tutorial ini membahas solusi SLAM (Simultaneous Localization and Mapping) yang diimplementasikan menggunakan Webots. Berikut adalah beberapa fitur Webots SLAM yang memberikan kelebihan dalam pengembangan algoritma SLAM:

1. **Simulasi Realistik:**
    - Webots menyediakan simulasi lingkungan robotik yang sangat realistik, memungkinkan pengembang untuk menguji dan mengembangkan algoritma SLAM dalam berbagai skenario dan kondisi tanpa memerlukan akses fisik ke robot atau lingkungan fisik.
2. **Debugging dan Visualisasi:**
    - Alat visualisasi yang kuat dalam Webots memfasilitasi pemantauan proses SLAM secara real-time. Ini mempermudah pengembang untuk melakukan debugging dan memahami interaksi algoritma SLAM dengan lingkungan simulasi.
3. **Reproduksi dan Pembuktian Konsep:**
    - Kemampuan untuk mengulangi pengujian dan pengembangan dengan mudah di lingkungan simulasi memungkinkan pengembang untuk membuktikan konsep SLAM tanpa batasan sumber daya atau waktu yang mungkin ditemui di lingkungan fisik.
4. **Dukungan untuk Sensor dan Actuator:**
    - Webots mendukung berbagai sensor dan aktuator yang umumnya digunakan dalam konteks SLAM, termasuk sensor jarak, kamera, sensor inersia (IMU), dan motor. Ini memungkinkan simulasi perangkat keras yang sesuai dengan robot fisik yang akan diimplementasikan.
5. **Pemrograman dan Pengujian Off-line:**
    - Pengembang dapat memprogram dan menguji algoritma SLAM secara off-line, tanpa perlu mengakses robot fisik atau lingkungan fisik. Hal ini memungkinkan fokus pengembangan sebelum mengujinya di lingkungan fisik.
6. **Dokumentasi dan Materi Edukasi:**
    - Webots menyediakan dokumentasi yang baik dan materi edukasi yang membantu pengguna memahami konsep dasar SLAM dan implementasinya menggunakan simulator.

Perlu dicatat bahwa, meskipun simulasi dapat memberikan banyak keuntungan dalam pengembangan dan pengujian, pengujian akhir pada robot fisik tetap diperlukan untuk memastikan bahwa algoritma SLAM dapat berfungsi di dunia nyata dengan semua ketidakpastian dan variabilitas yang melekat.

Video 17

Webots SLAM with Noise merujuk pada simulasi di lingkungan Webots yang memasukkan elemen kebisingan atau noise yang umumnya ditemui dalam sensor-sensor robotik di dunia nyata. Beberapa aspek utama terkait dengan Webots SLAM with Noise melibatkan:

1. **Sensor Noise:**
    - Webots memungkinkan pengguna untuk mensimulasikan noise pada sensor-sensor yang umumnya digunakan oleh robot, seperti lidar, kamera, dan sensor jarak. Noise tersebut menciptakan ketidakpastian dalam pengukuran jarak, sudut, dan jenis data sensor lainnya.
2. **Pemodelan Kebisingan:**
    - Pengguna dapat mengatur parameter untuk memodelkan jenis kebisingan yang mungkin muncul pada sensor. Ini mencakup noise sistem, noise lingkungan, atau noise yang dihasilkan oleh sensor itu sendiri. Pemodelan ini memberikan pengembang kontrol atas sifat kebisingan yang diinginkan.
3. **Akurasi Pengukuran:**
    - Dengan memasukkan noise pada pengukuran sensor, lingkungan simulasi Webots SLAM with Noise memberikan situasi yang lebih realistis untuk menguji algoritma SLAM. Pengembang dapat mengukur sejauh mana algoritma mereka dapat menangani ketidakpastian dalam data sensor dan masih dapat melakukan lokalizasi dan pemetaan dengan baik.
4. **Evaluasi Kinerja:**
    - Kehadiran noise memungkinkan pengembang untuk lebih akurat mengevaluasi kinerja algoritma SLAM dalam kondisi mendekati dunia nyata. Ini melibatkan penilaian sejauh mana robot dapat mempertahankan estimasi posisi yang tepat dan membangun peta yang akurat dalam kehadiran noise sensor.
5. **Strategi Koreksi dan Kalibrasi:**
    - Di lingkungan simulasi Webots SLAM with Noise, pengembang dapat menguji dan mengoptimalkan strategi koreksi dan kalibrasi untuk mengurangi dampak noise pada sensor. Ini termasuk teknik seperti filtrasi data atau kalibrasi sensor untuk meningkatkan akurasi pengukuran.

Penting untuk dicatat bahwa, meskipun simulasi dengan noise di Webots memberikan platform yang berguna untuk pengembangan dan pengujian awal algoritma SLAM, pengujian akhir di lingkungan fisik

tetap diperlukan untuk memvalidasi kinerja algoritma di dunia nyata dengan semua kompleksitas dan ketidakpastian yang mungkin terjadi.

Video 18

Perencanaan jalur di dunia dengan peta yang diketahui adalah elemen kritis dalam pengembangan sistem robotika yang dapat beroperasi secara otonom dalam lingkungan yang telah terpetakan. Berikut adalah langkah-langkah dan konsep utama terkait perencanaan jalur:

1. **Representasi Peta:**
    - Peta lingkungan dapat direpresentasikan dalam bentuk grid 2D atau 3D, yang mencakup informasi tentang rintangan, ruang terbuka, dan fitur lainnya. Peta ini dapat dibuat sebelumnya atau diperbarui secara real-time oleh robot menggunakan sensor-sensor seperti lidar, kamera, atau sensor jarak.
2. **Titik Awal dan Tujuan:**
    - Proses dimulai dengan menetapkan titik awal dan tujuan dalam peta. Titik awal adalah posisi awal robot, sedangkan titik tujuan adalah lokasi yang ingin dicapai.
3. **Algoritma Pencarian:**
    - Algoritma pencarian digunakan untuk menjelajahi peta dan menemukan jalur optimal antara titik awal dan tujuan. Algoritma yang umum digunakan meliputi Dijkstra, A* (A-star), atau algoritma lainnya. Pemilihan algoritma tergantung pada kebutuhan spesifik dan karakteristik lingkungan.
4. **Fungsi Biaya:**
    - Fungsi biaya diperhitungkan selama pencarian jalur untuk mengevaluasi keinginan dari setiap jalur. Fungsi biaya dapat mencakup faktor-faktor seperti jarak, waktu traversing, atau konsumsi energi. Tujuannya adalah menemukan jalur keseluruhan dengan biaya terendah.
5. **Penghindaran Rintangan:**
    - Penghindaran rintangan penting untuk memastikan bahwa robot dapat mengelilingi atau menghindari rintangan di sekitarnya. Ini melibatkan penyesuaian jalur untuk menghindari rintangan yang mungkin ada dalam peta.
6. **Lingkungan Dinamis:**
    - Dalam lingkungan dinamis, perencanaan jalur mungkin memerlukan penyesuaian untuk mengakomodasi perubahan kondisi seiring waktu. Hal ini dapat melibatkan pembaruan real-time terhadap peta atau perencanaan jalur yang adaptif.
7. **Penghalusan Jalur:**

- Teknik penghalusan jalur dapat diterapkan setelah jalur awal ditemukan. Ini membantu mengoptimalkan trayektori agar lebih memungkinkan dan halus bagi robot untuk diikuti, mengurangi kemungkinan gerakan yang kasar atau tiba-tiba.

8. **Otonomi dan Kolaborasi:**

    - Perencanaan jalur menjadi krusial untuk mencapai otonomi dalam robotika. Dalam situasi kolaborasi dengan manusia, algoritma perencanaan jalur juga dapat mempertimbangkan preferensi dan keselamatan manusia.

Penting untuk mencatat bahwa perencanaan jalur adalah bidang penelitian yang terus berkembang, dan berbagai algoritma dan teknik terus diperkenalkan untuk meningkatkan efisiensi, keamanan, dan keakuratan perencanaan jalur robotika.

Video 19

Proses perencanaan jalur di dalam lingkungan yang memiliki peta yang dikenal namun juga terdapat ketidakpastian atau "noise" melibatkan penyusunan strategi yang matang untuk mengatasi berbagai tantangan yang muncul akibat ketidakpastian dan gangguan dalam lingkungan tersebut. Sumber ketidakpastian, seperti ketidakakuratan pada sensor, pergerakan objek yang tidak dapat diprediksi, atau bahkan perubahan kondisi lingkungan seiring berjalannya waktu, menjadi faktor-faktor kunci yang perlu diperhatikan. Walaupun kita memiliki pemahaman yang baik tentang peta lingkungan, menangani ketidakpastian ini menjadi esensial untuk merancang jalur navigasi yang tidak hanya kokoh dan handal tetapi juga mampu menghadapi berbagai kemungkinan yang dapat mengganggu proses navigasi secara efisien.