Nama : Ardhien Fadhillah Suhartono

NIM : 1103204137

Week 8 Path Planning 8 – 19

Tutorial 8

Dimana pada robot ini menggunakan sensor posisi untuk mencari perbedaan pembacaan saat ini dan pembacaan sebelumnya untuk menentukan jarak linier yang telah ditempuh dari titik awalnya di sel grid 16 mulai dari koordinat 15.0, -15.0. Dari sini, robot dapat melacak seberapa jauh ia telah bergerak dari titik awal untuk melacak koordinatnya guna menentukan di kotak mana ia berada, dan dapat terus bergerak hingga mengunjungi setiap kotak.

Lalu pada kodingan merupakan bagian dari skrip untuk mengimplementasikan kontrol robot dalam lingkungan simulasi menggunakan Webots. Ini tampaknya merupakan implementasi kontrol gerakan dan navigasi sederhana untuk robot yang dilengkapi dengan sensor jarak, sensor posisi roda, kamera, dan unit inersia (IMU).

Berikut penjelasan singkat untuk beberapa bagian kodingan:

1. Inisialisasi dan Pengaturan Robot:
    a. Mendeklarasikan dan menginisialisasi beberapa variabel, seperti waktu langkah (timestep), sensor jarak, sensor posisi roda, kamera, IMU, dan motor roda.
    b. Menetapkan beberapa konstanta dan parameter robot, seperti radius roda, sirkumferensi roda, unit encoder, kecepatan maksimum, dan jarak antar roda.
2. Update Fungsi Sensor dan Kontrol Gerakan:
    a. Fungsi `get_p_sensors_vals` dan `get_d_sensors_vals` mengembalikan nilai sensor posisi roda dan jarak dalam satuan inci.
    b. Fungsi `move` menggerakkan robot sejauh sejumlah inci tertentu dengan menggunakan motor roda.
3. Fungsi Rotasi dan Pemutaran:
    a. Fungsi `rotate` memungkinkan robot berputar sejumlah derajat dalam waktu tertentu.
    b. Fungsi `turn_left` dan `turn_right` digunakan untuk memutar robot ke kiri atau kanan.
4. Fungsi Pemantauan dan Pembaruan Robot:
    a. Fungsi `update_robot` digunakan untuk memantau dan memperbarui status robot, termasuk posisi, orientasi, dan sel lain yang berkaitan dengan navigasi.
    b. Fungsi `get_robot_x_y` menghitung posisi baru robot berdasarkan perbedaan sensor posisi roda.
    c. Fungsi `get_current_grid_cell` menentukan sel grid saat ini berdasarkan posisi x dan y.
    d. Fungsi `update_direction` mengupdate arah hadap robot berdasarkan nilai sensor IMU.
5. Navigasi ke Sel Tertentu:
    a. Fungsi `move_to_cell` digunakan untuk memindahkan robot ke sel tertentu dalam grid dengan mempertimbangkan posisi relatifnya dan mengatur arah hadapnya.
    b. Fungsi `main` menggunakan serangkaian pergerakan dan navigasi untuk memindahkan robot ke sel yang telah ditentukan.

Kodingan ini menggunakan beberapa fungsi dan variabel untuk melakukan navigasi robot dalam grid sel. Robot bergerak dari satu sel ke sel berikutnya, dan setiap sel yang dikunjungi ditandai dengan 'X' pada array `visited_cells`. Tujuannya mungkin untuk membuat robot menjelajahi seluruh grid atau mencapai sel tertentu sesuai dengan skenario simulasi yang diinginkan.

Tutorial 9

Sama seperti tutorial sebelumnya akan tetapi robot ini juga mengalami kebisingan pada unit pengukuran inersia dan sensor jaraknya, karena robot ini terus melacak bagaimana cara menambahkan perbedaan nilai pada koordinat x dan y berdasarkan arah yang dituju, perlu ada solusi bagi IMU untuk melakukannya. menentukan apakah ia bergerak ke Utara, Timur, Selatan, atau Barat. Saya memecahkan masalah ini dengan membuat kamus python dirs = {"North": 0, "West": 0, "East": 0, "South": 0}, untuk setiap langkah gerak saya akan menambahkan arah sesuai dengan Nilai IMU, saya kemudian akan berasumsi bahwa robot sedang menuju ke arah yang memiliki hitungan tertinggi, melakukan hal ini memecahkan masalah saya tentang arah mana yang dituju dari satu sel ke sel lainnya dan dengan demikian memecahkan masalah koordinatnya juga! Menyetel ulang kamus untuk setiap arah baru yang coba dilalui juga sangat penting untuk mencegah estimasi arah yang miring.

Untuk kodingan keduanya hamper sama tetapi ada juga perbedaan diantara kedunnya seperti:

1. Kemiripan
    a. Keduanya memiliki fungsionalitas serupa untuk pergerakan robot, pembacaan sensor, dan navigasi berbasis grid.
    b. Struktur keseluruhan kode, seperti mengimpor pustaka, menginisialisasi robot, dan mendefinisikan fungsi, mirip.
2. Perbedaan
    a. Ada perbedaan dalam nama variabel, komentar, dan detail khusus dalam fungsi-fungsi tersebut, tetapi logika inti dan struktur sepertinya tetap.
    b. Potongan kode kedua memperkenalkan fitur tambahan seperti navigasi berbasis grid dan penandaan sel (dikunjungi atau tidak) yang tidak ada dalam potongan pertama.

Secara ringkas, dua 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. 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.

Tutorial 10

Pada tutorial kali ini terdapat skrip Webots yang mendemonstrasikan cara menggunakan trilaterasi di setiap sel untuk menentukan lokasi robot di peta grid. Robot ini berputar 360 derajat di setiap sel untuk mendeteksi keempat silinder dan menggunakan sistem persamaan dan trilaterasi untuk menentukan posisi robot di peta grid.

Dimana tedapat beberapa fitur tambahan seperti:

1. Navigasi Berbasis Grid:
    a. Pada potongan kode kedua, terdapat variabel `visited_cells` yang mencatat sel mana yang telah dikunjungi.
    b. Pembaruan status sel (`'.'` untuk belum dikunjungi, `'X'` untuk sudah dikunjungi) dilakukan saat robot bergerak melalui fungsi `move_to_cell`.
2. Penandaan Sel:
    a. Fungsi `move_to_cell` ditambahkan untuk menggerakkan robot menuju sel tertentu dan memperbarui penandaan sel yang telah dikunjungi di `visited_cells`.
3. Trilateration:
    a. Potongan kode kedua memperkenalkan fungsi-fungsi terkait trilaterasi (`full_rotate`, `get_abcdef`, `trilateration`) untuk mengestimasi posisi robot berdasarkan deteksi objek berwarna menggunakan kamera.
    b. Objek berwarna (silinder kuning, merah, hijau, biru) dikenali, dan jaraknya digunakan untuk melakukan trilaterasi.
4. Navigasi ke Sel Berikutnya:
    a. Potongan kode kedua memiliki loop utama (`main`) yang memberikan robot tugas untuk mengunjungi setiap sel dalam urutan tertentu.
    b. Robot akan menghentikan tugasnya jika seluruh sel telah dikunjungi.
5. Fungsi-Fungsi Tambahan:
    a. 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:
    a. Fungsi-fungsi seperti `move`, `stop_motors`, `rotate`, `turn_left`, dan `turn_right` memberikan kontrol gerak pada robot untuk mencapai tujuan.
7. Sensor dan Aktuator:
    a. 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.

Tutorial 11

Sama seperti sebelumnya terdapat tambahan Webots yang memperkenalkan noise di kedua sensor jarak yang membuatnya lebih sulit untuk memperkirakan seberapa jauh jarak robot dari setiap silinder. Untuk menyiasatinya, saya menggunakan jarak relatif kamera, meratakannya dengan nilai sensor jarak depan, lalu merata-ratakan banyak nilai tersebut untuk setiap silinder untuk mendapatkan radius dari robot ke setiap silinder. Hal ini memungkinkan pembacaan sensor yang lebih akurat. Selain itu, saya juga mengubah fungsi trilaterasi saya untuk menggunakan 3 silinder terjauh dari robot untuk menghindari penggunaan jarak yang terlalu dekat dengan silinder untuk mendapatkan pembacaan yang akurat.

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

1. Navigasi Berbasis Grid dan Posisi Robot:
    a. Penggunaan variabel `robot_pose` untuk menyimpan posisi robot dalam koordinat global (x, y), nomor sel (n), dan orientasi (theta).
    b. Pembaruan posisi robot dilakukan dalam fungsi `update_robot`, yang mencakup pembacaan sensor dan trilateration untuk perkiraan posisi.
2. Sensor dan Aktuator:
    a. 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`).
    b. 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:
    a. Fungsi-fungsi `full_rotate`, `get_abcdef`, dan `trilateration` digunakan untuk melakukan trilaterasi berdasarkan deteksi objek berwarna (cilinder kuning, merah, hijau, biru) menggunakan kamera.
    b. Perkiraan posisi robot diperbarui dengan melakukan trilaterasi dalam fungsi `update_robot` ketika parameter `trilat=True`.
4. Navigasi ke Sel Berikutnya:
    a. 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:
    a. Penggunaan list `visited_cells` untuk memantau sel mana yang telah dikunjungi (ditandai dengan 'X').
    b. Fungsi `print_visited_cells` untuk mencetak status sel yang telah dikunjungi.
6. Rotasi dan Kontrol Gerak:
    a. Fungsi-fungsi `rotate`, `turn_left`, dan `turn_right` digunakan untuk mengatur rotasi robot.
    b. Kontrol gerak melibatkan pengaturan kecepatan motor dalam fungsi-fungsi `move` dan `stop_motors`.
7. Penghentian Otomatis:
    a. Fungsi `check_if_robot_should_stop` memeriksa apakah seluruh sel sudah dikunjungi, dan jika ya, program dihentikan.
8. Optimisasi Perjalanan:

a. Dalam fungsi `main`, algoritma dipertimbangkan untuk mempercepat perjalanan robot dengan melompati sel yang telah dikunjungi sebelumnya.

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

Tutorial 12

Pada tutorial ini terdapat script yang memandu robot melalui labirin menggunakan algoritma dan filter partikel. Robot mendapatkan dinding sekitarnya, melakukan estimasi pengukuran pada setiap partikel dalam setiap sel, dan menggerakkan dirinya berdasarkan bobot sel setelah dinormalisasi. Model pengukuran menerapkan 5% noise pada sensor, sementara model gerak memungkinkan partikel maju 90% waktu dan tetap 10% waktu ketika memungkinkan. Setelah setiap pergerakan, robot kembali menghadap ke Utara dan mengulangi proses hingga setiap sel dikunjungi.

Berikut adalah beberapa kelebihan fitur pada kode yang baru dibandingkan dengan kode sebelumnya:

1. Particle Filter Implementation:
    a. Kode ini mengimplementasikan filter partikel untuk estimasi posisi robot. Filter partikel adalah metode statistik yang memungkinkan pemodelan ketidakpastian dan memberikan perkiraan yang lebih baik tentang posisi sejati robot.
    b. Partikel yang digunakan untuk memodelkan variasi posisi dan orientasi robot dalam lingkungan.
2. Penanganan Gerakan dan Pembaruan Posisi:
    a. Kode mengelola gerakan robot dan pembaruan posisi berdasarkan pembacaan dari sensor posisi roda dan sensor inersia.
    b. Robot dipindahkan dan diputar sesuai dengan gerakan fisiknya, dan posisi serta orientasi robot diperbarui.
3. Pemodelan Sensor dan Pengukuran:
    a. Pemodelan sensor jarak diimplementasikan dengan menambahkan noise pada pembacaan sensor untuk mensimulasikan ketidakpastian dalam pengukuran.
    b. Pengukuran dari sensor jarak digunakan untuk mengestimasi probabilitas keberadaan dinding di sekitar robot.
4. Visualisasi Data:
    a. Kode mencetak informasi yang terstruktur, termasuk pemetaan lingkungan, status sel yang dikunjungi, dan informasi partikel filter.
    b. Visualisasi data membantu pemahaman yang lebih baik tentang cara robot berinteraksi dengan lingkungan.
5. Resampling Partikel:
    a. Algoritma resampling partikel digunakan untuk mengatasi masalah degenerasi dan memberikan bobot pada partikel yang lebih signifikan probabilitasnya.
    b. Partikel yang diperbarui secara dinamis berdasarkan pergerakan dan pengukuran, sehingga meningkatkan akurasi estimasi.
6. Implementasi Algoritma Gerak dan Pembaruan:

    a. Kode menyertakan algoritma untuk menggerakkan robot dan memperbarui posisi, mempertimbangkan noise gerakan dan ketidakpastian sensor.
    b. Pembaruan berbasis pengukuran dan gerakan dilakukan dengan mempertimbangkan lingkungan sekitar robot.
7. Optimisasi dan Pembersihan Kode:
    a. Kode telah dioptimalkan dan dibersihkan untuk memperjelas alur logika dan mempermudah pemahaman.

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

Tutorial 13

Pada turial kali ini mempunyai skrip yang menggunakan filter partikel dengan 80 partikel untuk membimbing robot dalam menyelesaikan labirin. Ini menggunakan algoritma komprehensif yang 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 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.

Pada kodingan ini mengimplementasikan algoritma Particle Filter pada simulasi robot untuk menjelajahi labirin. Beberapa fitur utama dibandingkan dengan kodingan sebelumnya melibatkan penanganan informasi posisi dan orientasi robot dalam sel grid, penggunaan partikel untuk memodelkan estimasi posisi robot, dan pemrosesan data sensor untuk memperbarui matriks labirin. Partikel digunakan untuk merepresentasikan kemungkinan posisi robot dalam labirin, dan algoritma resampling digunakan untuk memperbarui partikel berdasarkan hasil pengukuran sensor dan kontrol robot. Selain itu, kodingan ini mencakup pengukuran noise, model pergerakan robot, dan pembaruan matriks labirin dengan informasi dinding yang dilihat oleh robot. Semua fitur ini bekerja sama untuk memungkinkan robot menjelajahi labirin dan mengidentifikasi posisi serta orientasinya dengan menggunakan filter partikel.

Tutorial 14

Pada tutorial kali ini menunjukkan bagaimana robot dapat melacak lokasinya di dunia hanya dengan mengetahui lokasi awalnya di dunia dan dapat mengumpulkan potongan-potongan dari apa yang diamati menjadi representasi berguna dari dunia yang dialaminya. Hal penting yang tidak saya sebutkan adalah bagaimana setiap kali robot mengunjungi sel, ia mendapatkan dinding dan menambahkannya ke susunan 2D SEBELUM membuat keputusan berikutnya, ia memerlukan informasi tersebut untuk mengetahui bahwa dinding berada di lokasi tertentu dan gerakan tertentu tidak dapat dilakukan.

Lalu pada kodingan yang merupakan implementasi kontroler untuk robot dalam simulasi labirin menggunakan Webots. Dalam simulasi ini, robot dilengkapi dengan sensor jarak, sensor posisi, kamera, dan unit inersia. Tujuannya adalah untuk secara otonom memetakan labirin dan mencatat dinding-dinding yang ditemui selama eksplorasi. Kode ini mengandalkan inisialisasi robot dan sensor, konfigurasi parameter seperti radius roda, serta pemetaan labirin menggunakan sejumlah fungsi. Proses pemetaan dilakukan dengan strategi yang memprioritaskan kunjungan ke sel-sel labirin yang belum pernah dikunjungi sebelumnya. Secara keseluruhan, struktur kodingan ini didesain untuk keterbacaan dan pemeliharaan, membuatnya lebih mudah dipahami dan dikembangkan. Selain itu, kodingan ini mencakup inisialisasi robot, termasuk pembuatan instance robot dan pengambilan timestep dari simulasi. Pada tahap ini, sensor-sensor dan motor-motor robot juga diinisialisasi.

Fitur visual tidak terlupakan, dengan kamera yang diaktifkan untuk mengambil informasi visual dari sekitar robot dan pengenalan objek pada kamera. Untuk memperoleh informasi orientasi robot, kodingan ini juga mengaktifkan unit inersia (IMU). Pengaturan kecepatan motor menjadi fokus berikutnya, di mana handler motor diperoleh dan kecepatan serta posisi target motor diatur sesuai kebutuhan. Beberapa variabel dan konstanta robot disimpan, seperti radius roda dan kecepatan maksimum robot. Labirin sendiri diwakili dalam bentuk matriks, dan konfigurasi labirin serta pemetaan menjadi bagian integral dari kodingan. Algoritma pemetaan labirin mengutamakan kunjungan ke sel-sel yang belum pernah dikunjungi sebelumnya. Fungsi-fungsi utilitas, seperti konversi satuan, perhitungan waktu pergerakan, dan pemutakhiran posisi robot, juga diterapkan. Navigasi dan kontrol gerak robot, bersama dengan pemantauan dan pembaruan posisi robot, membentuk bagian krusial dari kodingan ini.

Seluruhnya, kodingan ini mencakup aspek-aspek spesifik, seperti penanganan arah hadap robot dalam menghadapi empat arah utama, yakni Utara, Barat, Selatan, dan Timur. Semua fitur dan komponen disusun secara sistematis dalam kodingan ini untuk mencapai tujuan utama: pemetaan labirin secara otonom.

Tutorial 15

Pada tutorial kali ini mirip dengan video Lab 4 tugas 1 sebelumnya, dimana ini memiliki 10% noise pada sensor jarak, unit pengukuran inersia, dan sensor posisi. Saya menjelaskan bagaimana metode kamus saya digunakan untuk mengetahui arah utama robot.

Kode yang terdapat pada tutorial merupakan program untuk mengendalikan robot pada lingkungan simulasi menggunakan Webots. Program ini mungkin ditulis dalam bahasa pemrograman Python dan menggunakan modul-modul dari Webots API (contohnya, `controller`). Robot bergerak di dalam grid yang merepresentasikan labirin, dan tujuannya adalah memetakan labirin tersebut. Beberapa fitur utama dari program ini melibatkan pemantauan sensor dan tindakan yang diambil berdasarkan informasi tersebut:

1. Pemantauan Sensor:
    a. Program ini menggunakan berbagai sensor, seperti sensor jarak (`DistanceSensor`), sensor posisi (`PositionSensor`), kamera (`Camera`), dan unit inersia (`InertialUnit`). Informasi dari sensor-sensor ini digunakan untuk mengambil keputusan mengenai pergerakan dan pemetaan.

2. Gerakan Robot:
    a. 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:
    a. Program ini mencoba memetakan labirin dengan mengenali tembok dan mengidentifikasi sel-sel yang telah dikunjungi. Pemetaan dilakukan berdasarkan sensor jarak dan informasi orientasi robot.
4. Navigasi Pintar:
    a. Program berusaha untuk melakukan navigasi yang efisien dan pintar di dalam labirin dengan mempertimbangkan tembok yang ada, arah pergerakan robot, dan sel-sel yang telah dikunjungi.
5. Orientasi Robot:
    a. Program menggunakan sensor unit inersia (IMU) untuk menentukan orientasi global robot. Informasi ini digunakan untuk memperbarui posisi dan arah robot di dalam labirin.
6. Penanganan Posisi dan Arah:
    a. Robot menyimpan informasi tentang posisi, orientasi, dan sel-sel yang telah dikunjungi. Program berusaha untuk memastikan bahwa robot dapat memetakan seluruh labirin dan menghindari jalur yang sudah dilalui sebelumnya.
7. Pengendalian Alur Program:
    a. Program menggunakan perulangan untuk terus beroperasi sampai kondisi tertentu terpenuhi, seperti pemetaan selesai atau tugas selesai.

Tutorial 16

Tutorial ini menjelaskan solusi SLAM (lokalisasi dan pemetaan simultan) saya. SLAM adalah masalah yang sangat sulit dalam robotika karena robot perlu mencari tahu di mana letaknya tanpa harus membuat peta. Cara kami menyiasatinya dalam video ini pada dasarnya adalah dengan menggunakan landmark untuk memperkirakan lokasi dunia pertama kali (lokalisasi) dan kemudian kami mulai menyatukan bagian-bagian dunia setelah itu (pemetaan).

Berikut adalah beberapa fitur Webots SLAM yang dapat memberikan kelebihan dibandingkan dengan pengembangan langsung pada perangkat fisik:

1. Simulasi Realistik:
    a. Webots menyediakan simulasi lingkungan robotik yang sangat realistik, yang memungkinkan pengguna untuk menguji dan mengembangkan algoritma SLAM dalam berbagai skenario dan kondisi tanpa memerlukan akses fisik ke robot atau lingkungan fisik.
2. Debugging dan Visualisasi:
    a. Webots menyediakan alat visualisasi yang kuat untuk melihat proses SLAM secara real-time. Ini memudahkan pengembang untuk melakukan debugging dan memahami bagaimana algoritma SLAM berinteraksi dengan lingkungan simulasi.
3. Reproduksi dan Pembuktian Konsep:

    a. Pengguna dapat mengulangi pengujian dan pengembangan dengan mudah tanpa harus khawatir tentang batasan sumber daya atau waktu di lingkungan fisik. Ini memungkinkan pengembang untuk secara efisien menguji dan membuktikan konsep SLAM mereka.
4. Dukungan untuk Sensor dan Actuator:
    a. Webots mendukung berbagai sensor dan aktuator yang umumnya digunakan dalam konteks SLAM, seperti sensor jarak (distance sensors), kamera, sensor inersia (IMU), dan motor. Ini memungkinkan pengguna untuk mensimulasikan perangkat keras yang sesuai dengan robot fisik mereka.
5. Pemrograman dan Pengujian Off-line:
    a. Pengembang dapat memprogram dan menguji algoritma SLAM secara off-line tanpa ketergantungan pada ketersediaan robot fisik atau lingkungan fisik. Ini memungkinkan pengembang untuk memfokuskan pengembangan mereka sebelum menguji di lingkungan fisik.
6. Dokumentasi dan Materi Edukasi:
    a. Webots menyediakan dokumentasi yang baik dan materi edukasi yang membantu pengguna memahami konsep-konsep dasar SLAM dan cara mengimplementasikannya dengan menggunakan simulator.

Penting untuk dicatat bahwa sementara Webots memberikan kelebihan ini, pengujian akhir pada robot fisik tetap diperlukan untuk memastikan bahwa algoritma SLAM dapat berfungsi dengan baik di dunia nyata dengan semua ketidakpastian dan variabilitas yang melekat.

Tutorial 17

Webots SLAM with Noise mengacu pada simulasi di lingkungan Webots yang mencakup elemen-elemen kebisingan atau noise yang biasanya ditemukan dalam sensor-sensor robotik di dunia nyata. Dalam konteks SLAM, sensor-sensor seperti lidar, kamera, dan sensor jarak biasanya terkena noise yang dapat mempengaruhi akurasi pengukuran dan estimasi posisi robot.

Berikut adalah beberapa aspek utama terkait Webots SLAM with Noise:

1. Sensor Noise:
    a. Webots memungkinkan pengguna untuk menyimulasikan noise pada sensor-sensor yang digunakan oleh robot. Misalnya, sensor lidar dapat memiliki noise yang menciptakan ketidakpastian dalam pengukuran jarak dan sudut. Ini mencerminkan kondisi sebenarnya di lapangan di mana sensor sering kali tidak memberikan pengukuran yang sempurna.
2. Pemodelan Kebisingan:
    a. Pada tingkat tinggi, pengguna dapat mengatur parameter untuk memodelkan jenis kebisingan yang mungkin muncul pada sensor. Ini bisa termasuk noise sistem, noise lingkungan, atau noise yang dihasilkan oleh sensor itu sendiri. Pemodelan ini memungkinkan pengembang untuk menguji seberapa baik algoritma SLAM mereka dapat menangani ketidakpastian yang diakibatkan oleh noise.
3. Akurasi Pengukuran:
    a. Dengan memasukkan noise pada pengukuran sensor, Webots SLAM with Noise memberikan lingkungan yang lebih realistis untuk menguji algoritma SLAM. Pengembang

dapat mengukur sejauh mana algoritma mereka dapat mengatasi ketidakpastian dalam data sensor dan tetap dapat melakukan lokalizasi dan pemetaan dengan baik.

4. Evaluasi Kinerja:
    a. Dengan kehadiran noise, pengembang dapat secara lebih akurat mengevaluasi kinerja algoritma SLAM dalam kondisi yang mendekati dunia nyata. Ini mencakup melihat sejauh mana robot dapat mempertahankan estimasi posisi yang tepat dan membangun peta yang akurat dalam kehadiran noise sensor.
5. Strategi Koreksi dan Kalibrasi:
    a. Dalam lingkungan Webots SLAM with Noise, pengembang dapat menguji 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 mencatat bahwa simulasi dengan noise di Webots adalah langkah awal penting dalam pengembangan dan pengujian algoritma SLAM, tetapi pengujian akhir di lingkungan fisik tetap diperlukan untuk memvalidasi kinerja algoritma di dunia nyata yang penuh ketidakpastian dan noise yang kompleks.

Tutorial 18

Perencanaan jalur di dunia dengan peta yang diketahui melibatkan penentuan jalur optimal atau memungkinkan bagi sebuah robot atau agen untuk bergerak dari titik awal ke titik tujuan dalam suatu lingkungan yang telah terpetakan sebelumnya. Pada dasarnya, robot memahami struktur lingkungannya melalui representasi peta, yang dapat berupa grid 2D atau 3D, mencakup informasi tentang rintangan, ruang terbuka, dan fitur lainnya.

Proses dimulai dengan menetapkan titik awal dan tujuan dalam peta yang diketahui tersebut. Algoritma pencarian, seperti Dijkstra, A* (A-star), atau variasi lainnya, kemudian digunakan untuk menjelajahi peta dan menemukan jalur paling efisien berdasarkan kriteria tertentu, seperti jarak atau biaya. Fungsi biaya diperhitungkan untuk mengevaluasi keinginan dari setiap jalur, mempertimbangkan faktor-faktor seperti jarak, waktu traversing, atau konsumsi energi, dengan tujuan menemukan jalur keseluruhan dengan biaya terendah. Penting untuk mempertimbangkan penghindaran rintangan, bahkan dalam lingkungan yang sudah terpetakan, agar robot dapat mengelilingi rintangan dan tetap berada pada jalur optimal. Di lingkungan dinamis, perencanaan jalur mungkin memerlukan penyesuaian untuk mengakomodasi perubahan kondisi seiring waktu.

Teknik penghalusan jalur dapat diterapkan setelah jalur awal ditemukan, membantu mengoptimalkan t
rayektori agar lebih memungkinkan dan halus bagi robot untuk diikuti. Dalam implementasi robotika, perencanaan jalur menjadi krusial dalam mencapai otonomi, di mana robot yang dilengkapi sensor dapat secara otonom mengidentifikasi lokasinya dalam peta yang diketahui dan mengeksekusi jalur yang telah direncanakan. Selain itu, dalam situasi kolaborasi dengan manusia, algoritma perencanaan jalur juga dapat mempertimbangkan preferensi dan keselamatan manusia. Hal ini bertujuan untuk memastikan bahwa jalur robot tidak hanya efisien secara teknis, tetapi juga sosial, sehingga dapat diterima oleh manusia dan menghindari potensi tabrakan atau gangguan.

Tutorial 19

Path planning in a world with a known map with noise involves strategizing to overcome challenges presented by uncertainties and disturbances in the environment, often referred to as "noise." Noise can emanate from various sources, including inaccuracies in sensors, unpredictable movements of objects, or alterations in the environment over time. Despite having a familiar map of the environment, addressing these uncertainties becomes crucial for devising paths that facilitate robust and dependable navigation.

Dalam perencanaan jalur di dunia dengan peta yang sudah dikenal dan terdapat noise, terlibat dalam penyusunan strategi untuk mengatasi tantangan yang muncul dari ketidakpastian dan gangguan di lingkungan, yang sering disebut sebagai "noise." Gangguan dapat berasal dari berbagai sumber, termasuk ketidakakuratan sensor, pergerakan objek yang tidak terduga, atau perubahan di lingkungan seiring waktu. Meskipun memiliki peta yang dikenal di lingkungan tersebut, penanganan ketidakpastian ini menjadi krusial untuk merancang jalur yang memfasilitasi navigasi yang kuat dan dapat diandalkan.