Nama : Indra Andriansyah Dody Misnadin
NIM : 1103200005
Kelas : TK-44-G7

Lecture 8: Playlist Path Planning 8-19

Tutorial 8

Pada robot ini, penggunaan sensor posisi dimanfaatkan untuk mengukur perbedaan antara pembacaan saat ini dan pembacaan sebelumnya. Hal ini digunakan untuk menghitung jarak linier yang sudah ditempuh oleh robot sejak titik awalnya di dalam grid seluas 16 dengan koordinat awal 15.0, -15.0. Dengan informasi ini, robot dapat terus melacak perpindahannya dan menentukan koordinatnya, memungkinkan untuk mengidentifikasi kotak di mana robot berada. Proses ini memungkinkan robot untuk bergerak terus-menerus dan mengunjungi setiap kotak dalam lingkungan grid tersebut.

Selanjutnya, dalam kode ini terdapat bagian skrip yang bertujuan mengimplementasikan kendali robot di dalam lingkungan simulasi menggunakan platform Webots. Sepertinya, ini adalah implementasi kontrol gerakan dan navigasi yang sederhana untuk robot yang dilengkapi dengan berbagai sensor, termasuk sensor jarak, sensor posisi roda, kamera, dan unit inersia (IMU). Dengan bantuan sensor-sensor ini, robot dapat diatur untuk bergerak dan berorientasi dalam lingkungan simulasi, memungkinkan penentuan posisi dan navigasi yang akurat.

Penjelasan singkat untuk beberapa bagian kodingan:

- **Inisialisasi dan Pengaturan Robot:**
    - Deklarasi dan inisialisasi variabel, seperti timestep, sensor jarak, sensor posisi roda, kamera, IMU, dan motor roda.
    - Penetapan konstanta dan parameter robot, seperti radius roda, sirkumferensi roda, unit encoder, kecepatan maksimum, dan jarak antar roda.

- **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 sejauh sejumlah inci tertentu dengan menggunakan motor roda.

- **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.

- **Fungsi Pemantauan dan Pembaruan Robot:**
    - Fungsi `update_robot` digunakan untuk memantau dan memperbarui status robot, termasuk posisi, orientasi, dan sel lain yang berkaitan dengan 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.

- **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 telah ditentukan.

Dengan menggunakan serangkaian fungsi dan variabel, kodingan ini dirancang untuk memungkinkan navigasi robot dalam grid sel. Prosesnya melibatkan pergerakan robot dari satu sel ke sel berikutnya, dan setiap sel yang telah dikunjungi ditandai dengan 'X' pada array `visited_cells`. Tujuannya mungkin berkisar antara menjelajahi seluruh grid atau mencapai sel tertentu, sesuai dengan skenario simulasi yang diinginkan. Dengan menggunakan pendekatan ini, kodingan memberikan kontrol yang cukup fleksibel untuk mengarahkan pergerakan robot sesuai dengan kebutuhan simulasi yang spesifik.

Tutorial 9

Dalam mengatasi masalah kebisingan pada unit pengukuran inersia (IMU) dan sensor jarak, pendekatan yang diambil melibatkan penyesuaian nilai koordinat x dan y berdasarkan arah yang dituju. Solusinya melibatkan penggunaan kamus Python, yaitu `dirs = {"North": 0, "West": 0, "East": 0, "South": 0}`. Setiap langkah gerakan akan menambahkan nilai arah sesuai dengan nilai IMU, dan asumsi dibuat bahwa robot bergerak ke arah yang memiliki hitungan tertinggi.

Dengan melakukan ini, masalah arah yang dituju dari satu sel ke sel lainnya dapat diatasi, dan hal ini juga membantu dalam menyelesaikan masalah koordinat. Penting untuk menyetel ulang kamus untuk setiap arah baru yang dicoba dilalui, ini bertujuan untuk mencegah estimasi arah yang miring dan memastikan akurasi yang lebih baik dalam menentukan arah sebenarnya yang diambil oleh robot.

Secara keseluruhan, pendekatan ini memberikan solusi yang efektif untuk mengatasi kebisingan pada pengukuran sensor, mengoptimalkan informasi dari IMU, dan memastikan akurasi pergerakan robot dalam menentukan arahnya di dalam grid sel.

Analisis perbandingan kodingan keduanya:

Kemiripan:

1. Fungsionalitas Serupa:
    a. Keduanya memiliki fungsionalitas yang serupa untuk mengendalikan pergerakan robot, membaca sensor, dan melakukan navigasi berbasis grid.

2. Struktur Kode Keseluruhan:
    a. Struktur keseluruhan kode mirip, termasuk mengimpor pustaka, menginisialisasi robot, dan mendefinisikan fungsi-fungsi utama.

Perbedaan:

1. Nama Variabel, Komentar, dan Detail Khusus:
    a. Ada perbedaan dalam nama variabel, komentar, dan detail spesifik dalam fungsi-fungsi masing-masing. Namun, logika inti dan struktur umum sepertinya tetap konsisten.

2. Fitur Tambahan pada Kodingan Kedua:
    a. Potongan kode kedua memperkenalkan fitur tambahan, seperti navigasi berbasis grid dan penandaan sel (dikunjungi atau tidak). Fitur ini tidak ada dalam potongan kode pertama.

Meskipun ada perbedaan-perbedaan detail, kesamaan dalam fungsionalitas utama dan struktur menunjukkan bahwa keduanya mungkin merupakan variasi dari implementasi yang serupa. Potongan kode kedua mungkin merupakan pengembangan dari potongan

pertama dengan penambahan fitur-fitur tertentu untuk keperluan navigasi dan pelacakan posisi robot yang lebih canggih dalam grid sel.

Secara singkat, kedua potongan kode memiliki struktur dan tujuan yang mirip, namun perbedaan detail dan fitur tambahan membedakan keduanya. Potongan kode kedua mengenalkan fitur tambahan yang signifikan untuk mendukung navigasi dan pemetaan posisi robot berbasis grid dalam simulasi. Beberapa fitur ini melibatkan navigasi berbasis grid dengan membagi lingkungan menjadi sel-sel, pelacakan pergerakan robot antara sel-sel, dan penandaan sel yang sudah dikunjungi menggunakan array `visited_cells`. Terdapat juga penyesuaian orientasi dan rotasi menggunakan fungsi-fungsi seperti `turn_left`, `turn_right`, dan `rotate` untuk mengatur perubahan orientasi dan rotasi robot.

Dalam implementasi navigasi ke sel berikutnya, fungsi `move_to_cell` bertanggung jawab membimbing robot menuju sel berikutnya berdasarkan grid. Selama proses ini, status grid terus dipantau, dan setiap pergerakan robot diikuti dengan penandaan status sel yang sudah dikunjungi menggunakan array `visited_cells`. Struktur data tambahan, seperti `n_rc` yang mencocokkan indeks dan kolom grid, digunakan untuk memetakan posisi robot ke grid sel.

Pada intinya, potongan kode kedua diperkaya dengan fitur-fitur ini untuk menangani tugas-tugas navigasi yang melibatkan pemantauan posisi robot dalam konteks grid pada simulasi yang lebih kompleks. Ini memberikan kemampuan robot untuk secara cerdas bergerak di sepanjang grid, menavigasi dengan tepat, dan melacak sel mana yang telah dikunjungi. Dengan demikian, potongan kode tersebut dapat digunakan untuk simulasi yang memerlukan navigasi dan pemetaan posisi robot yang lebih canggih dalam lingkungan yang lebih kompleks.

Tutorial 10

Pada tutorial ini, terdapat skrip Webots yang menunjukkan penggunaan trilaterasi di setiap sel untuk menentukan lokasi robot pada peta grid. Robot ini dirancang untuk berputar 360 derajat di setiap sel guna mendeteksi keempat silinder yang mungkin ada dalam lingkungan simulasi. Setelah mendeteksi silinder, sistem persamaan dan teknik trilaterasi digunakan untuk menghitung dan menentukan posisi akurat robot pada peta grid.

Proses ini dapat dianggap sebagai teknik triangulasi yang memanfaatkan informasi jarak relatif antara robot dan keempat silinder. Dengan memperoleh data ini, robot dapat menghitung posisinya secara akurat di dalam grid. Seluruh skrip Webots dirancang untuk mendemonstrasikan konsep dan implementasi trilaterasi sebagai metode untuk menentukan posisi robot dengan memanfaatkan informasi sensor di lingkungan simulasi Webots.

Berikut beberapa fitur tambahan yang memperluas fungsionalitas dan kemampuan robot:

A. Navigasi Berbasis Grid:
    a. Penggunaan variabel `visited_cells` untuk 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`.
    c.
B. Penandaan Sel:
    a. Fungsi `move_to_cell` ditambahkan untuk menggerakkan robot menuju sel tertentu dan memperbarui penandaan sel yang telah dikunjungi di `visited_cells`.
    b.
C. Trilateration:
    a. Pengenalan fungsi-fungsi terkait trilaterasi (`full_rotate`, `get_abcdef`, `trilateration`) untuk mengestimasi posisi robot berdasarkan deteksi objek berwarna menggunakan kamera.
    b. Identifikasi objek berwarna (silinder kuning, merah, hijau, biru) dan penggunaan jaraknya untuk melakukan trilaterasi.
    c.
D. Navigasi ke Sel Berikutnya:
    a. Loop utama (`main`) memberikan robot tugas untuk mengunjungi setiap sel dalam urutan tertentu.
    b. Robot akan berhenti jika seluruh sel telah dikunjungi.
    c.
E. 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.
    b.

F. Kontrol Gerak:
    a. Fungsi-fungsi seperti `move`, `stop_motors`, `rotate`, `turn_left`, dan `turn_right` memberikan kontrol gerak pada robot untuk mencapai tujuan.
    b.
G. 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 penambahan fitur pada lingkungan simulasi Webots yang memperkenalkan noise pada kedua sensor jarak. Kehadiran noise tersebut memberikan tantangan dalam mengestimasi jarak antara robot dan setiap silinder. Untuk mengatasi hal ini, dilakukan pendekatan dengan menggunakan informasi jarak relatif dari kamera dan menggabungkannya dengan nilai sensor jarak depan. Selanjutnya, dilakukan proses merata-ratakan sejumlah nilai ini untuk setiap silinder dengan tujuan memperoleh estimasi radius yang lebih akurat dari robot ke masing-masing silinder. Pendekatan ini bertujuan untuk meningkatkan akurasi pembacaan sensor. Selain itu, terdapat modifikasi pada fungsi trilaterasi, dimana kini digunakan informasi dari tiga silinder terjauh dari robot untuk menghindari penggunaan jarak yang terlalu dekat dengan silinder. Langkah-langkah ini secara keseluruhan diimplementasikan untuk meningkatkan ketepatan pembacaan sensor dan estimasi posisi robot dalam menghadapi noise pada lingkungan simulasi.

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

A. 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.
    c.
B. 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`.
    c.
C. 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`.
    c.
D. 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.
    b.
E. 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.

c.

F. 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`.
    c.
G. Penghentian Otomatis:
    a. Fungsi `check_if_robot_should_stop` memeriksa apakah seluruh sel sudah dikunjungi, dan jika ya, program dihentikan.
    b.
H. Optimisasi Perjalanan:
    a. Dalam fungsi `main`, algoritma dipertimbangkan untuk mempercepat perjalanan robot dengan melompati sel yang telah dikunjungi sebelumnya.

Secara keseluruhan, kodingan ini menunjukkan peningkatan yang signifikan dalam hal pemantauan, navigasi, dan estimasi posisi robot. Peningkatan tersebut membuatnya lebih sesuai untuk tugas simulasi di lingkungan grid yang melibatkan elemen tambahan seperti trilaterasi dan kontrol yang lebih canggih. Dengan fitur-fitur seperti penggunaan variabel `robot_pose` untuk menyimpan informasi posisi secara lebih rinci, penggunaan sensor dan aktuator yang lebih kompleks, implementasi trilaterasi untuk memperbaiki pembacaan sensor, dan navigasi yang lebih cerdas ke sel berikutnya, kodingan ini dapat mengatasi tugas-tugas yang lebih kompleks dalam simulasi lingkungan grid. Peningkatan ini menunjukkan kemampuan yang lebih baik dalam menghadapi tantangan dan kompleksitas yang mungkin muncul dalam konteks simulasi yang lebih realistis.

Tutorial 12

Pada tutorial ini, terdapat skrip yang memandu robot melalui labirin menggunakan algoritma dan filter partikel. Robot mengumpulkan informasi tentang dinding di sekitarnya, melakukan estimasi pengukuran pada setiap partikel dalam setiap sel, dan menggerakkan dirinya berdasarkan bobot sel setelah dinormalisasi. Model pengukuran mengaplikasikan 5% noise pada sensor, sementara model gerak memungkinkan partikel maju 90% waktu dan tetap diam 10% waktu ketika dimungkinkan. 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 digunakan untuk memodelkan variasi posisi dan orientasi robot dalam lingkungan.
2. Penanganan Gerakan dan Pembaharuan 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.

Tutorial 13

Pada tutorial ini, terdapat skrip yang memanfaatkan filter partikel dengan menggunakan 80 partikel untuk membimbing robot dalam menyelesaikan labirin. Skrip ini mengimplementasikan algoritma komprehensif yang mengevaluasi dinding-dinding di sekitar robot, mengidentifikasi langkah-langkah yang tersedia (kiri, depan, atau kanan), dan melakukan estimasi pengukuran pada partikel berdasarkan dinding-dinding yang terdeteksi.

Langkah-langkah utama termasuk perhitungan bobot setiap partikel, normalisasi bobot, dan meresample partikel sesuai dengan bobot yang telah dinormalisasi. 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 akan berbelok ke selatan, dan selanjutnya, ke timur, untuk menavigasi labirin.

Model pengukuran diperkenalkan dengan menerapkan 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 kehandalan algoritma untuk menyelesaikan tugas penavigasian di dalam labirin.

Dalam kodingan ini, diterapkan algoritma Particle Filter pada simulasi robot untuk mengeksplorasi labirin. Beberapa fitur utama yang membedakannya dari 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.

Penggunaan partikel bertujuan merepresentasikan kemungkinan posisi robot dalam labirin. 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 terlihat oleh robot. Semua fitur ini bekerja bersama-sama untuk memungkinkan robot mengeksplorasi labirin, mengidentifikasi posisi dan orientasinya dengan menggunakan filter partikel.

Tutorial 14

Pada tutorial ini, ditunjukkan bagaimana robot dapat melacak lokasinya di dunia dengan mengetahui lokasi awalnya dan mengumpulkan potongan-potongan informasi dari lingkungannya untuk membentuk representasi yang berguna dari dunia yang dijelajahi. Setiap kali robot mengunjungi sel, ia memperoleh informasi tentang dinding dan menambahkannya ke susunan 2D SEBELUM membuat keputusan berikutnya. Informasi ini menjadi krusial karena membantu robot mengetahui lokasi dinding dan menghindari gerakan yang tidak dapat dilakukan.

Kemudian, pada kodingan implementasi kontroler untuk robot dalam simulasi labirin menggunakan Webots. Robot dilengkapi dengan sensor jarak, sensor posisi, kamera, dan unit inersia. Tujuan utamanya adalah melakukan pemetaan labirin secara otonom dan mencatat dinding-dinding yang ditemui selama eksplorasi. Kode ini memanfaatkan inisialisasi robot dan sensor, konfigurasi parameter seperti radius roda, serta strategi pemetaan labirin melalui sejumlah fungsi.

Strategi pemetaan didesain untuk memprioritaskan kunjungan ke sel-sel labirin yang belum pernah dikunjungi sebelumnya. Struktur kodingan ini didesain dengan baik untuk keterbacaan dan pemeliharaan, membuatnya mudah dipahami dan dikembangkan. Inisialisasi robot, termasuk pembuatan instance robot, pengambilan timestep dari simulasi, inisialisasi sensor dan motor, semuanya dicakup dalam kodingan ini.

Fitur visual menjadi elemen kunci dalam kodingan ini, di mana kamera diaktifkan untuk mengumpulkan informasi visual dari sekitar robot dan melakukan pengenalan objek. Untuk mendapatkan informasi orientasi robot, kodingan ini juga menggunakan unit inersia (IMU). Pengaturan kecepatan motor menjadi fokus berikutnya, dengan mengakses handler motor untuk mengatur kecepatan dan posisi target motor sesuai kebutuhan.

Berbagai variabel dan konstanta yang berkaitan dengan robot disimpan, seperti radius roda dan kecepatan maksimum robot. Representasi labirin diwakili dalam bentuk matriks, dan konfigurasi labirin serta pemetaan menjadi bagian penting dari kodingan. Algoritma pemetaan labirin dirancang untuk memberikan prioritas 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 integral 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 membahas solusi SLAM (Simultaneous Localization and Mapping), suatu permasalahan kompleks dalam robotika di mana robot harus secara simultan menentukan lokasinya dan membuat peta dari lingkungannya. Dalam konteks tutorial ini, solusi SLAM diimplementasikan dengan menggunakan landmark sebagai titik referensi untuk memperkirakan lokasi robot pertama kali (lokalisasi) dan kemudian menyusun bagian-bagian dari peta dunia seiring berjalannya waktu (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 realtime. 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.

Tutorial 17

Webots SLAM with Noise merujuk pada simulasi di lingkungan Webots yang mencakup elemen kebisingan atau noise yang sering dihadapi oleh sensor-sensor robotik di dunia nyata. Dalam konteks Simultaneous Localization and Mapping (SLAM), sensor-sensor seperti lidar, kamera, dan sensor jarak seringkali terpengaruh oleh 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 lokalisasi 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.

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, dan mencakup informasi tentang rintangan, ruang terbuka, dan fitur lainnya. Proses perencanaan jalur ini memungkinkan robot untuk membuat keputusan cerdas tentang langkah-langkah yang harus diambil untuk mencapai tujuan dengan mempertimbangkan kendala dan karakteristik lingkungan yang telah terpetakan sebelumnya.

Proses perencanaan jalur dimulai dengan menetapkan titik awal dan tujuan dalam peta yang diketahui. 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. Ini dapat melibatkan pemantauan terus-menerus terhadap lingkungan dan penyesuaian jalur secara dinamis untuk mengatasi hambatan baru atau perubahan dalam peta.

Teknik penghalusan jalur dapat digunakan untuk meningkatkan jalur yang telah ditemukan sebelumnya. Hal ini dilakukan dengan membuat jalur menjadi lebih halus dan mudah dilalui oleh robot.

Dalam robotika, perencanaan jalur merupakan hal yang penting untuk mencapai otonomi. Robot yang dilengkapi sensor dapat secara mandiri 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 dilakukan 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

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.