HASIL ANALISIS

A. COLAB

1. **Implementasi Filter Kalman untuk Estimasi Posisi Robot**

   Kalman Filter adalah algoritma yang menggunakan estimasi berbasis probabilitas untuk menggabungkan berbagai sumber informasi yang memiliki ketidakpastian. Biasanya digunakan untuk estimasi posisi robot dengan memanfaatkan sensor yang memiliki noise.

   Penjelasan Kode:

   Prediksi Posisi (Predicted State): Pada setiap langkah waktu, x_pred diperkirakan berdasarkan posisi sebelumnya. Ini berfungsi untuk memperbarui estimasi posisi berdasarkan model dinamika robot.

   Pembaruan Posisi dengan Pengukuran (Update Step): Setelah memperoleh pengukuran z dari sensor, Kalman Filter memperbarui estimasi posisi dengan menggunakan Gain Kalman (K) yang mengkombinasikan informasi dari prediksi dan pengukuran sensor. Gain Kalman ini didapat dari rasio ketidakpastian prediksi terhadap total ketidakpastian (prediksi + pengukuran).

   Gain Kalman (K): Gain Kalman memutuskan seberapa besar kontribusi pengukuran dan prediksi terhadap estimasi posisi akhir.

   Visualisasi: Setelah 100 langkah simulasi, posisi yang diestimasi digambarkan dalam plot untuk menunjukkan bagaimana Kalman Filter mengestimasi posisi dengan akurat meskipun ada noise di dalam pengukuran.

2. **Implementasi Filter Partikel untuk Estimasi Posisi Robot**

   Particle Filter adalah metode Monte Carlo yang digunakan untuk mengestimasi posisi robot ketika modelnya non-linear dan distribusi noise tidak diketahui. Filter Partikel menggunakan banyak partikel untuk merepresentasikan berbagai kemungkinan posisi robot.

   Penjelasan Kode:

   Inisialisasi Partikel: Partikel diinisialisasi secara acak dalam rentang posisi tertentu (misalnya, 0 hingga 10). Setiap partikel mewakili kemungkinan posisi robot.

   Prediksi Gerakan Partikel: Setiap partikel dipindahkan berdasarkan noise gerakan (misalnya, np.random.normal) untuk mensimulasikan ketidakpastian gerakan robot.

   Pembaruan Bobot: Bobot partikel dihitung berdasarkan seberapa dekat posisi partikel dengan pengukuran sensor z (jarak yang diukur). Bobot partikel dihitung menggunakan distribusi Gaussian.

   Resampling: Setelah pembaruan bobot, dilakukan resampling partikel, dimana partikel dengan bobot lebih tinggi lebih sering dipilih untuk mendominasi populasi partikel yang baru.

   Visualisasi: Estimasi posisi robot dihitung sebagai rata-rata posisi seluruh partikel dan divisualisasikan setelah 100 langkah simulasi.

3. **Implementasi Localization dengan Sensor IMU dan Lidar**

   Dalam implementasi ini, sensor IMU memberikan data orientasi (yaw), sedangkan Lidar memberikan informasi tentang jarak ke objek di sekitar robot. Dengan informasi ini, kita dapat memperkirakan posisi robot di ruang dua dimensi.

Penjelasan Kode:

Data IMU: Diperoleh sebagai perubahan yaw, yang digunakan untuk memperbarui orientasi robot (theta).

Data Lidar: Diperoleh sebagai jarak ke objek di depan robot, yang digunakan untuk memperbarui posisi robot dalam arah sumbu X dan Y berdasarkan orientasi robot (theta).

Perubahan Posisi dan Orientasi: Dengan menggunakan IMU (untuk perubahan yaw) dan Lidar (untuk jarak), kita menghitung posisi baru robot setiap kali langkah waktu dilakukan.

4. **Implementasi Simulasi Ekstensi Kalman Filter untuk Navigation**

Ekstensi Kalman Filter (EKF) adalah versi Kalman Filter yang dirancang untuk menangani sistem non-linear. Di dalam konteks robotik dan navigasi, robot sering kali bergerak di dunia dengan model non-linear, dan pengukuran sensor juga tidak selalu linier. EKF mengatasi masalah ini dengan melakukan **linearization** terhadap model sistem dan pengukuran.

Pada umumnya, EKF digunakan ketika model gerakan dan pengukuran robot tidak linier, seperti dalam hal perhitungan posisi dengan sensor Lidar atau GPS yang mengukur jarak atau koordinat.

Penjelasan Kode:

Model Gerakan Non-linear: Fungsi motion_model() digunakan untuk memprediksi posisi robot berdasarkan kecepatan linier (v) dan kecepatan sudut (w) menggunakan model gerakan non-linear.

Model Pengukuran Non-linear: Fungsi measurement_model() mengukur jarak robot dari titik asal menggunakan pengukuran Lidar (misalnya, jarak Euclidean ke origin).

Prediksi dan Pembaruan EKF: Fungsi ekf_update() melakukan langkah prediksi dan pembaruan EKF. Dalam prediksi, posisi robot dihitung berdasarkan model gerakan, sementara pada pembaruan, prediksi posisi ini diperbarui berdasarkan pengukuran sensor (Lidar dalam hal ini). Estimasi posisi diperbarui dengan gain Kalman.

Visualisasi: Setelah simulasi, posisi robot yang sebenarnya (true position) dan posisi estimasi (estimated position) digambarkan untuk menunjukkan perbandingan hasil yang didapat oleh EKF.

5. **Implementasi Particle Filter untuk Navigation**

Particle Filter (PF) adalah metode berbasis Monte Carlo yang dapat digunakan untuk estimasi posisi dalam sistem yang sangat non-linear dan dengan model yang lebih kompleks. PF mengandalkan populasi partikel untuk menggambarkan distribusi kemungkinan posisi robot.

Penjelasan Kode:

Inisialisasi Partikel: Partikel diinisialisasi secara acak dalam ruang dua dimensi untuk menggambarkan distribusi kemungkinan posisi robot. Setiap partikel memiliki posisi dan orientasi (x, y, theta).

Prediksi Partikel: Fungsi predict_particles() memperbarui posisi dan orientasi partikel berdasarkan kecepatan linier (v) dan sudut (w) dengan menambahkan noise acak untuk mensimulasikan ketidakpastian.

Update Bobot Partikel: Fungsi update_weights() memperbarui bobot masing-masing partikel berdasarkan kesesuaian antara posisi partikel dan pengukuran sensor (misalnya Lidar). Bobot dihitung menggunakan distribusi Gaussian.

Resampling Partikel: Fungsi resample() melakukan resampling partikel berdasarkan bobot yang diperbarui. Partikel dengan bobot lebih tinggi memiliki peluang lebih besar untuk dipilih.

Estimasi Posisi: Posisi estimasi dihitung dengan mengambil rata-rata posisi partikel setelah resampling.
Visualisasi: Hasil estimasi posisi yang diperoleh dari filter partikel digambarkan untuk melihat seberapa baik metode ini dapat memperkirakan posisi robot dalam waktu yang berjalan.

B. SIMULASI

Simulasi ini menggunakan Webots, platform robotika simulasi, dengan **Python controller**. Kode ini mengimplementasikan **Kalman Filter** untuk estimasi posisi robot dalam sebuah lingkungan berbasis sensor dan encoder.

Penjelasan Kode Program :

A. Inisialisasi dan Pengaturan Waktu Simulasi

TIME_STEP mengatur waktu langkah simulasi (dalam milidetik). Setiap langkah simulasi di Webots akan memproses kode sebanyak satu kali per interval waktu yang ditentukan oleh TIME_STEP. Di sini, 32 ms berarti simulasi akan dijalankan setiap 32 milidetik.

B. **Fungsi Kalman Filter**

Input Kalman Filter:

z: Pengukuran sensor yang didapat dari sensor jarak.
u: Input kontrol atau estimasi pergerakan robot berdasarkan pembacaan dari encoder roda.
x: Estimasi posisi robot sebelumnya (posisi awalnya adalah 0.0).
P: Ketidakpastian dari estimasi posisi sebelumnya.
Langkah-langkah dalam Kalman Filter:
Prediksi Langkah:

- **Posisi Prediksi**: x_pred = x + u, yaitu estimasi posisi robot berdasarkan pergerakan yang dihitung dengan data encoder roda.
- **Ketidakpastian Prediksi**: P_pred = P + 0.1, menambah ketidakpastian pada prediksi, yang disebabkan oleh noise proses (misalnya kesalahan perhitungan atau faktor eksternal).

Koreksi Langkah:

- **Gain Kalman**: K = P_pred / (P_pred + 1), ini adalah faktor yang mengatur seberapa banyak pengukuran (z) akan mempengaruhi pembaruan estimasi posisi. Nilai ini akan lebih besar jika ketidakpastian prediksi (P_pred) lebih kecil daripada noise pengukuran (1).
- **Update Posisi**: x = x_pred + K * (z - x_pred), di sini estimasi posisi diperbarui dengan mengoreksi prediksi berdasarkan pengukuran sensor (z), yang dikalikan dengan **Gain Kalman**.
- **Update Ketidakpastian**: P = (1 - K) * P_pred, di mana ketidakpastian diperbarui berdasarkan gain yang dihitung.

Fungsi kalman_filter() ini mengembalikan dua nilai: posisi terestimasi terbaru (x) dan ketidakpastian estimasi (P).

C. Inisialisasi Robot, Sensor, dan Motor

Inisialisasi Robot: robot = Robot() menginisialisasi objek Robot yang digunakan untuk mengakses perangkat di dalam simulasi Webots.

Motor Roda:

left_motor dan right_motor adalah objek motor yang menggerakkan roda kiri dan kanan robot. Motor ini diatur ke mode kecepatan dengan setPosition(float('inf')), yang berarti robot bergerak dengan kecepatan konstan (bukan pada posisi tertentu).

Sensor Encoder:

left_encoder dan right_encoder digunakan untuk menghitung jarak yang ditempuh oleh masing-masing roda dengan mengakses sensor roda.

enable(TIME_STEP) mengaktifkan sensor setiap langkah simulasi.

Sensor Jarak:

distance_sensor adalah sensor yang mengukur jarak ke objek terdekat di depan robot. Sensor ini diaktifkan dengan enable(TIME_STEP) agar bisa memberikan pembacaan setiap langkah simulasi.

D. **Variabel Kalman Filter**

x: Posisi awal robot (diasumsikan 0).

P: Ketidakpastian awal estimasi posisi (nilai yang lebih besar berarti estimasi lebih tidak pasti).

E. Loop Utama Simulasi

Loop Utama: robot.step(TIME_STEP) memastikan simulasi berjalan langkah demi langkah. Setiap iterasi di dalam loop ini mengambil data dari sensor dan encoder, menghitung estimasi posisi dengan Kalman Filter, dan mencetak hasil estimasi posisi.

Langkah-langkah dalam Loop:

Ambil Nilai Encoder:

- left_distance = left_encoder.getValue() dan right_distance = right_encoder.getValue() membaca jarak yang telah ditempuh oleh roda kiri dan kanan berdasarkan sensor encoder.

Estimasi Pergerakan Robot (u):

- u = (left_distance + right_distance) / 2.0 menghitung rata-rata jarak yang ditempuh oleh kedua roda, yang digunakan untuk estimasi pergerakan robot (kecepatan atau jarak total yang ditempuh).

Ambil Pengukuran Sensor Jarak (z):

- z = distance_sensor.getValue() mendapatkan pengukuran jarak dari sensor jarak, yang memberikan informasi mengenai posisi robot relatif terhadap objek di depannya.

Terapkan Kalman Filter:

- Fungsi kalman_filter(z, u, x, P) dijalankan untuk memperbarui estimasi posisi robot (x) dan ketidakpastian (P) dengan pengukuran sensor dan input kontrol (pergerakan robot).

Cetak Estimasi Posisi:

- print(f"Estimasi Posisi Robot: {x}") mencetak posisi robot yang terestimasi berdasarkan hasil Kalman Filter.