Nama : Muhammad Husain Kaasyiful Ghitha

NIM : 1103220047

Kelas TK-45-G09

Robotika dan Sistem Cerdas

Analisis Tugas Robotika Week 13

Pada tugas ini, dilaksanakan beberapa peragaan teori Parametric Filtering menggunakan bahasa Python dengan library numpy dan matplotlib. Praktik-praktik yang dilakukan adalah sebagai berikut:

- Extended Kalman Filter (EKF): Robot Navigasi dengan GPS dan IMU
- Unscented Kalman Filter (UKF) untuk Estimasi Navigasi Robot Menggunakan Data GPS dan IMU
- Tracking Objek Bergerak dengan Kalman Filter
- Tracking Drone dengan Gerakan Parabola

Selain memrogram kode Python untuk simulasi tersebut, terdapat juga simulasi-simulasi yang berfokus pada implementasinya pada robot menggunakan simulator Webots. Simulasi yang dilakukan pada Webots tersebut antara lain:

- Estimasi Posisi Robot Menggunakan Teknik Pembelajaran Mesin
- Lokalisasi Robot Beroda Empat dengan Filter Kalman

Berikut adalah pembahasan mengenai kode Python dan simulasi-simulasi yang dilakukan.

1. Membuat kode Python menggunakan numpy dan matplotlib untuk simulasi mengenai Parametric Filtering.

   Pada kode simulasi ini, penjelasan fungsi-fungsi yang dimuat di program adalalah sebagai berikut:

   a. Extended Kalman Filter (EKF): Robot Navigasi dengan GPS dan IMU.

      Kode yang diberikan mengimplementasikan Extended Kalman Filter (EKF) untuk memperkirakan posisi objek yang bergerak menggunakan data GPS dan IMU (Inertial Measurement Unit) yang disimulasikan. Kode ini ditulis dalam Python dan

menggunakan NumPy untuk operasi numerik. Fungsi jacobian_motion menghitung matriks Jacobian untuk model gerak, yang digunakan dalam langkah prediksi EKF. Fungsi measurement_model memodelkan pengamatan GPS dengan mengekstrak posisi x dan y dari vektor keadaan. Fungsi jacobian_measurement mengembalikan matriks Jacobian untuk model pengukuran.

Simulasi dimulai dengan mengatur seed acak untuk reproduktibilitas dan membuat daftar untuk menyimpan posisi sebenarnya, data GPS, dan kontrol IMU. Dalam setiap iterasi, kontrol IMU (kecepatan linier dan sudut) diatur dan ditambahkan ke daftar imu_controls. Posisi sebenarnya diperbarui menggunakan fungsi motion_model dan ditambahkan ke daftar true_positions. Data GPS dihasilkan dengan menambahkan noise ke posisi sebenarnya dan ditambahkan ke daftar gps_data. EKF melakukan langkah prediksi dan pembaruan untuk setiap pengukuran GPS, memperkirakan posisi dan menyimpannya dalam daftar estimated_positions..

Pada gambaran output program, garis biru (estimasi EKF) mendekati garis hijau (jalur sebenarnya). Ini menunjukkan bahwa EKF bekerja dengan baik dalam memperkirakan posisi robot meskipun ada noise pada data GPS. Dengan membandingkan titik-titik merah (data GPS) dan garis biru (estimasi EKF), kita dapat melihat bagaimana EKF mampu menghaluskan noise dari data GPS dan memberikan estimasi yang lebih stabil dan akurat. Jalur estimasi EKF juga konsisten dan tidak terlalu berfluktuasi menunjukkan bahwa filter ini stabil dan dapat diandalkan dalam kondisi noise yang diberikan.

Secara keseluruhan, output visualisasi ini menunjukkan bahwa EKF mampu memperbaiki estimasi posisi robot dengan baik, meskipun terdapat noise pada data GPS, dan memberikan hasil yang mendekati jalur sebenarnya.

b. Unscented Kalman Filter (UKF) untuk Estimasi Navigasi Robot Menggunakan Data GPS dan IMU.

Kode yang diberikan mengimplementasikan Unscented Kalman Filter (UKF) untuk memperkirakan posisi objek yang bergerak menggunakan data GPS dan IMU (Inertial Measurement Unit) yang disimulasikan. Fungsi ‘fx’ mendefinisikan model transisi keadaan yang menggambarkan bagaimana keadaan sistem berkembang seiring waktu berdasarkan keadaan saat ini, input kontrol (kecepatan dan kecepatan sudut), dan interval waktu ‘dt’. Fungsi ini menghitung keadaan baru dengan

memperbarui posisi x dan y berdasarkan kecepatan dan sudut orientasi ‘theta’, serta memperbarui ‘theta’ berdasarkan kecepatan sudut. Fungsi ‘hx’ mendefinisikan model pengukuran yang mengekstrak posisi x dan y dari vektor keadaan.

Simulasi data dimulai dengan mengatur seed acak untuk reproduktibilitas dan menginisialisasi daftar untuk menyimpan keadaan sebenarnya, data GPS, dan input kontrol. Keadaan sebenarnya diinisialisasi dengan posisi awal ‘[0, 0, 0]’. Dalam setiap iterasi, input kontrol (kecepatan linier dan sudut) diatur dan ditambahkan ke daftar ‘controls’. Keadaan sebenarnya diperbarui menggunakan fungsi ‘fx’ dan ditambahkan ke daftar ‘true_states’. Data GPS dihasilkan dengan menambahkan noise ke posisi sebenarnya dan ditambahkan ke daftar ‘gps_data’. Implementasi UKF dimulai dengan menginisialisasi daftar kosong untuk posisi yang diperkirakan. Untuk setiap input kontrol, UKF melakukan langkah prediksi menggunakan metode ‘predict’ dan langkah pembaruan menggunakan metode ‘update’ dengan pengukuran GPS yang sesuai. Posisi yang diperkirakan kemudian divisualisasikan menggunakan Matplotlib, menunjukkan jalur sebenarnya, data GPS yang berisik, dan jalur yang diperkirakan oleh UKF..

Pada output dari kode ini, Jalur sebenarnya ditampilkan sebagai garis hijau, data GPS yang berisik ditampilkan sebagai titik merah, dan estimasi UKF ditampilkan sebagai garis ungu. Dari plot tersebut, dapat dilihat bahwa data GPS memiliki banyak noise, yang menyebabkan penyebaran titik-titik merah. Namun, estimasi UKF berhasil mengikuti jalur sebenarnya dengan lebih baik, menunjukkan bahwa UKF efektif dalam mengurangi noise dan memberikan estimasi posisi yang lebih akurat

Hasil ini menunjukkan bahwa meskipun data GPS memiliki banyak noise, estimasi UKF berhasil mengikuti jalur sebenarnya dari robot dengan lebih akurat, membuktikan efektivitas UKF dalam mengurangi noise dan meningkatkan akurasi estimasi posisi.

c. Tracking Objek Bergerak dengan Kalman Filter.

Kode yang diberikan mengimplementasikan Kalman Filter untuk memperkirakan posisi objek yang bergerak dalam ruang dua dimensi menggunakan data sensor yang disimulasikan. Fungsi ‘motion_model’ mendefinisikan model transisi keadaan yang menggambarkan bagaimana keadaan sistem berkembang seiring waktu. Vektor keadaan mencakup posisi x, kecepatan x, posisi y, dan

kecepatan y. Matriks transisi keadaan ‘F’ digunakan untuk memperbarui posisi berdasarkan kecepatan dan interval waktu ‘dt’.

Fungsi ‘measurement_model’ memodelkan proses pengamatan dengan mengekstrak posisi x dan y dari vektor keadaan. Fungsi ini digunakan untuk mensimulasikan pengukuran sensor. Fungsi ‘jacobian_measurement’ mengembalikan matriks Jacobian untuk model pengukuran, yang digunakan dalam langkah pembaruan Kalman Filter. Kode ini menginisialisasi berbagai variabel, termasuk interval waktu, vektor keadaan awal, matriks kovarians, matriks kovarians noise proses, dan matriks kovarians noise pengukuran. Kode ini kemudian mensimulasikan keadaan sebenarnya dan pengukuran sensor selama 100 langkah waktu. Keadaan sebenarnya diperbarui menggunakan model gerak sinusoidal, dan pengukuran sensor dihasilkan dengan menambahkan noise ke posisi sebenarnya.

Kalman Filter kemudian diterapkan untuk memperkirakan keadaan berdasarkan pengukuran yang berisik. Dalam setiap iterasi, filter melakukan langkah prediksi untuk memperkirakan keadaan dan kovarians berikutnya, diikuti dengan langkah pembaruan untuk mengoreksi perkiraan menggunakan pengukuran sensor. Keadaan yang diperkirakan disimpan dalam daftar. Akhirnya, keadaan sebenarnya, pengukuran, dan keadaan yang diperkirakan dipetakan menggunakan Matplotlib.

Output dari kode ini menunjukkan hasil estimasi posisi dan kecepatan objek yang bergerak di bidang 2D menggunakan Kalman Filter. Grafik yang dihasilkan memperlihatkan perbandingan antara posisi sebenarnya (true states), sensor yang terkontaminasi noise (Sensor Data), dan estimasi posisi oleh Kalman Filter (KF Estimates).

Hasilnya menunjukkan bahwa Kalman Filter mampu memperbaiki estimasi posisi objek dengan cukup baik meskipun terdapat noise pada sensor, sehingga estimasi posisi lebih mendekati posisi sebenarnya dibandingkan dengan hasil penginderaan mentah.

d. Implementasi Simulasi Extended Kalman Filter untuk Navigasi.

Kode yang diberikan mengimplementasikan Kalman Filter untuk memperkirakan posisi drone yang bergerak dalam lintasan parabola, dipengaruhi oleh gravitasi. Fungsi motion_model mendefinisikan model transisi keadaan yang menggambarkan bagaimana keadaan sistem berkembang seiring waktu. Vektor keadaan mencakup posisi x, kecepatan x, posisi y, dan kecepatan y. Matriks transisi

keadaan F digunakan untuk memperbarui posisi berdasarkan kecepatan dan interval waktu dt.

Kode ini menginisialisasi berbagai variabel, termasuk interval waktu, vektor keadaan awal, matriks kovarians, matriks kovarians noise proses, dan matriks kovarians noise pengukuran. Kode ini kemudian mensimulasikan keadaan sebenarnya dan pengukuran sensor selama 100 langkah waktu. Keadaan sebenarnya diperbarui menggunakan model gerak, dengan kecepatan y yang berkurang karena gravitasi. Pengukuran sensor dihasilkan dengan menambahkan noise ke posisi sebenarnya. Kalman Filter kemudian diterapkan untuk memperkirakan keadaan berdasarkan pengukuran yang berisik. Akhirnya, keadaan sebenarnya, pengukuran, dan keadaan yang diperkirakan dipetakan menggunakan Matplotlib, menunjukkan jalur sebenarnya dari drone, data sensor yang berisik, dan jalur yang diperkirakan oleh Kalman Filter.

Grafik yang dihasilkan memperlihatkan perbandingan antara jalur sebenarnya dari drone (true path), hasil sensor yang terkontaminasi noise (noisy measurements), dan estimasi jalur oleh Kalman Filter (estimated path). Hasilnya menunjukkan bahwa Kalman Filter mampu memperbaiki estimasi posisi drone dengan cukup baik meskipun terdapat noise pada pengukuran, sehingga estimasi posisi lebih mendekati jalur sebenarnya dibandingkan dengan hasil penginderaan.

2. Simulasi Webots Implementasi Parametric Filtering untuk Lokalisasi Robot.

   Pada simulasi Webots ini, penjelasan controller-controller yang dimuat di program adalalah sebagai berikut:

   a. Robot Positioning Estimation using ML Techniques

      Meskipun kode pada bagian utama dari controller robot itu sendiri terlihat ringkas, namun kode tersebut memanggil beberapa program ‘manager’ yang menangani operasi robot secara umum. Modul-modul tersebut, terutama ‘robot_manager’, berisi pengelolaan pergerakan robot, pengumpulan data, dan estimasi posisinya menggunakan berbagai teknik seperti odometri dan filter partikel. Kelas utama, RobotManager, menginisialisasi komponen robot, termasuk motor, sensor, dan antarmuka komunikasi. Kelas ini juga menyiapkan struktur data untuk menyimpan posisi nyata, odometri, dan prediksi robot.

Kelas RobotManager mencakup beberapa metode untuk mengontrol dan memantau robot. Metode init_actuators mengaktifkan sensor robot dan mengatur motor ke mode posisi tak terbatas, memungkinkan rotasi terus menerus. Metode init_robot_pos mengatur posisi dan orientasi awal robot. Metode step memajukan simulasi satu langkah waktu. Metode get_bearing_degrees menghitung orientasi robot dalam derajat menggunakan sensor kompas.

Pengumpulan data dan plotting ditangani oleh metode save_supervisor_coordinates, save_odometry_coordinates, dan save_sensor_distances, yang menyimpan posisi robot dan pembacaan sensor. Metode plot memvisualisasikan posisi nyata, odometri, dan prediksi robot menggunakan matplotlib. Metode execute adalah loop utama yang menjalankan simulasi robot, menginisialisasi aktuator dan posisi robot, kemudian memasuki loop di mana ia melakukan berbagai tugas seperti menerima pesan, melacak odometri, mengumpulkan data sensor, dan menghitung kecepatan baru.

Kelas DataCollector bertanggung jawab untuk mengumpulkan dan menyimpan data robot, termasuk posisi dan pembacaan sensor. Kelas MovementController menyediakan metode untuk menghitung kecepatan robot berdasarkan data sensor dan mengontrol pergerakan robot. Kelas WindowCommunicator menangani komunikasi antara robot dan antarmuka pengguna. Kelas ParticlesFilter mengimplementasikan algoritma filter partikel untuk memperkirakan posisi robot berdasarkan tindakan kontrol dan pembacaan sensor. Secara keseluruhan, kode ini mengintegrasikan berbagai komponen untuk mensimulasikan dan mengontrol robot, mengumpulkan data, dan memperkirakan posisinya menggunakan teknik filtering.

Ketika simulasi dilakukan, dapat dilihat bahwa robot berjalan dengan pola yang acak, disesuaikan dengan estimasi posisi yang dilakukan oleh robot pada saat itu. Pada Robot Window yang disediakan, terdapat sebuah plot yang menggambarkan jejak robot tersebut dari segi odometri maupun perkiraan yang muncul dari gambaran partikel. Meskipun pada percobaan hasil odometri menyimpang dari posisi asli, namun jejak odometri masih menunjukkan bahwa jalan prediksi posisi masih dapat mengikuti pergerakan dari robot.

b. Four-Wheeled Robot Localization with Kalman Filter

Kode controller robot pada Webots yang ada di repository memiliki fungsi dalam navigasi melalui beberapa waypoint. Pertama, robot diinisialisasi dengan perangkat-perangkatnya, termasuk motor, rem, sensor posisi roda, akselerometer, dan IMU (Inertial Measurement Unit). Motor belakang dan depan diatur dalam mode infinity untuk memungkinkan pengaturan kecepatan tanpa batasan posisi, sementara sensor posisi roda, akselerometer, dan IMU diaktifkan dengan waktu langkah (timestep) tertentu. Parameter dasar robot, seperti radius roda, jarak antar roda, dan kecepatan maksimum, juga didefinisikan untuk membantu perhitungan navigasi.

Fungsi utama dalam kode ini adalah moveToWaypoint, yang bertanggung jawab untuk mengarahkan robot menuju titik-titik tujuan (waypoints). Fungsi ini menggunakan pendekatan berbasis status (state-based approach), di mana robot menjalankan serangkaian langkah: pengereman, pengaturan arah roda (steering), dan pergerakan lurus. Status ditentukan berdasarkan orientasi robot relatif terhadap tujuan menggunakan informasi dari IMU untuk sudut yaw dan algoritma trigonometri untuk menghitung arah menuju tujuan (theta). Jika robot mendekati tujuan dalam batas toleransi (range), status akan beralih untuk berhenti dan mempersiapkan pergerakan ke waypoint berikutnya.

Proses navigasi ini diperhalus dengan kalkulasi jarak tempuh roda dari sensor posisi (odometer) dan akselerasi yang diubah menjadi kecepatan oleh akselerometer. Data ini digunakan untuk memperkirakan posisi robot saat ini, yang penting dalam menentukan jarak dan arah ke tujuan. Logika kontrol kecepatan motor dan pengaturan rem diterapkan untuk memastikan robot bergerak secara halus melalui transisi setiap status.

Terakhir, controller ini mengimplementasikan simulasi secara real-time dengan menjalankan lingkaran utama yang memperbarui waktu, memproses input sensor, memperbarui status navigasi, dan mengirimkan perintah ke perangkat robot seperti motor dan rem. Dengan struktur ini, robot mampu bergerak secara otonom melalui beberapa waypoint dalam lingkungan simulasi Webots, menunjukkan integrasi antara sensor, aktuator, dan algoritma kontrol.

Ketika simulasi dilakukan, robot bergerak dengan perkiraan yang telah dilakukan. Robot mengevaluasi status posisinya dengan hasil yang dikeluarkan ke console, sehingga dapat dianalisis secara langsung. Kemudian robot akan bergerak

dan berputar untuk mendekati titik tujuan, sampai robot berada di tujuan akhir dan diam sambil terus mengeluarkan output ‘reached’.