Laporan Analisis: Praktik Robotika dan Teori Filtering

1. Pendahuluan

Estimasi posisi robot dalam lingkungan yang terkontaminasi oleh noise adalah salah satu tantangan utama dalam robotika. Praktik ini bertujuan untuk mempelajari penerapan berbagai metode filtering, termasuk Filter Kalman, Filter Partikel, dan gabungan data sensor, untuk meningkatkan akurasi lokalization robot.

2. Tujuan Praktikum

Praktikum 1: Filter Kalman

Mempelajari cara menggunakan Filter Kalman untuk estimasi posisi robot dalam lingkungan dengan noise sensor.

Praktikum 2: Filter Partikel

Menggunakan Filter Partikel untuk estimasi posisi robot dalam lingkungan dengan ketidakpastian tinggi.

Praktikum 3: Kombinasi Data Sensor IMU dan Lidar

Menggabungkan data sensor IMU dan Lidar untuk meningkatkan akurasi lokalization.

Praktikum 4: Extended Kalman Filter (EKF)

Mempelajari penggunaan EKF untuk navigasi robot dalam sistem non-linear.

Praktikum 5: Filter Partikel untuk Navigasi

Menggunakan Filter Partikel untuk estimasi posisi dalam lingkungan kompleks.

3. Analisis Implementasi

3.1 Filter Kalman

Teori Dasar

Filter Kalman adalah metode filtering berbasis model linier yang memadukan data sensor dengan model sistem untuk memperkirakan keadaan sebenarnya. Dua langkah utama dalam Filter Kalman:

1. **Prediksi**: Memperkirakan keadaan robot berdasarkan model.
2. **Koreksi**: Memperbarui perkiraan menggunakan data pengukuran.

Implementasi Kode

```
import numpy as np

import matplotlib.pyplot as plt
```

```
# Definisi model sistem
A = np.array([[1, 1], [0, 1]])
B = np.array([[0.5], [1]])
H = np.array([[1, 0]])
Q = np.array([[1, 0], [0, 1]])
R = np.array([[10]])

# Inisialisasi
x = np.array([[0], [0]])
P = np.array([[1000, 0], [0, 1000]])
T = 50
measurements = np.linspace(0, T, T) + np.random.normal(0, 1, size=T)

# Filter Kalman
estimated_position = []
for z in measurements:
    x = np.dot(A, x) + np.dot(B, 1)
    P = np.dot(np.dot(A, P), A.T) + Q
    K = np.dot(np.dot(P, H.T), np.linalg.inv(np.dot(np.dot(H, P), H.T) + R))
    x = x + np.dot(K, z - np.dot(H, x))
    P = np.dot(np.eye(2) - np.dot(K, H), P)
    estimated_position.append(x[0, 0])

# Plot hasil
plt.plot(measurements, label="Measured Position")
plt.plot(estimated_position, label="Estimated Position")
plt.legend()
```

```
plt.show()
```

3.2 Filter Partikel

Teori Dasar

Filter Partikel adalah metode berbasis probabilitas yang memperkirakan posisi dengan menyebarkan partikel secara acak di ruang keadaan. Langkah utama meliputi:

1. **Prediksi**: Memperbarui posisi partikel berdasarkan model gerak.
2. **Update**: Menghitung bobot partikel berdasarkan pengukuran.
3. **Resampling**: Memilih kembali partikel berdasarkan bobot.

Implementasi Kode

```
import numpy as np
import matplotlib.pyplot as plt

# Inisialisasi partikel
N = 1000
particles = np.random.uniform(low=-5, high=5, size=N)
weights = np.ones(N) / N
measurements = 2 + np.random.normal(0, 0.5, size=N)

for measurement in measurements:
    weights = np.exp(-0.5 * (particles - measurement)**2)
    weights /= np.sum(weights)
    particles = particles[np.random.choice(range(N), size=N, p=weights)]

estimated_position = np.mean(particles)

# Visualisasi
plt.hist(particles, bins=50, alpha=0.6, color='g')
plt.axvline(x=estimated_position, color='r', linestyle='dashed')
```

```
plt.show()
```

3.3 Kombinasi Sensor IMU dan Lidar

Implementasi Kode

```
imu_data = np.linspace(0, 10, 100)

lidar_data = np.random.normal(5, 1, size=100)

alpha, beta = 0.7, 0.3

estimated_position = alpha * imu_data + beta * lidar_data

plt.plot(imu_data, label="IMU Data")

plt.plot(lidar_data, label="Lidar Data")

plt.plot(estimated_position, label="Fusion")

plt.legend()

plt.show()
```

4. Hasil dan Diskusi

- **Filter Kalman**: Mampu memberikan estimasi yang halus dan mendekati posisi sebenarnya.
- **Filter Partikel**: Cocok untuk lingkungan dengan noise tinggi, meskipun membutuhkan banyak komputasi.
- **Kombinasi Sensor**: Memberikan hasil estimasi lebih akurat dibandingkan data individu sensor.

5. Kesimpulan

Metode filtering memberikan alat yang efektif untuk memperkirakan posisi robot dalam lingkungan dengan ketidakpastian. Penggunaan metode yang sesuai bergantung pada kompleksitas lingkungan dan kemampuan komputasi.