Kaivalya Mishra CS460, NetID: km1233

# Assignment 3 Report

---

**Problem 1. Bayes Filter**

Probabilty of the floor being clean after vacuum-cleaning:

$P(x_{t+1} = clean|z = clean, u_t = vacuum\_clean)$

Applying Bayes Filter:

$$
\begin{aligned}
&P(x_{t+1} = clean|z = clean, u_t = vacuum\_clean) \\
&\quad = \eta P(z = clean|x_{t+1} = clean) \sum_{x_t} P(x_{t+1} = clean|x_t, u_t = vacuum\_clean)P(x_t) \\
&\quad = \eta P(z = clean|x_{t+1} = clean) \\
&\qquad \cdot (P(x_{t+1} = clean|x_t = clean, u_t = vacuum\_clean)P(x_t = clean) \\
&\qquad + P(x_{t+1} = clean|x_t = dirty, u_t = vacuum\_clean)P(x_t = dirty)) \\
&\quad = \eta 0.9(1 \cdot c + 0.7 \cdot (1 - c)) = \eta(0.63 + 0.27c)
\end{aligned}
$$

Now, the value of $\eta$ needs to be determined. Since $P(x_{t+1} = clean|z = clean, u_t = vacuum\_clean) + P(x_{t+1} = dirty|z = clean, u_t = vacuum\_clean) = 1$, $P(x_{t+1} = dirty|z = clean, u_t = vacuum\_clean)$ needs to be determined:

$$
\begin{aligned}
&P(x_{t+1} = dirty|z = clean, u_t = vacuum\_clean) \\
&\quad = \eta P(z = clean|x_{t+1} = dirty) \sum_{x_t} P(x_{t+1} = dirty|x_t, u_t = vacuum\_clean)P(x_t) \\
&\quad = \eta P(z = clean|x_{t+1} = dirty) \\
&\qquad \cdot (P(x_{t+1} = dirty|x_t = clean, u_t = vacuum\_clean)P(x_t = clean) \\
&\qquad + P(x_{t+1} = dirty|x_t = dirty, u_t = vacuum\_clean)P(x_t = dirty)) \\
&\quad = \eta 0.3(0 \cdot c + 0.3 \cdot (1 - c)) = \eta(0.09 - 0.09c)
\end{aligned}
$$

The value of $\eta$ can now be determined:

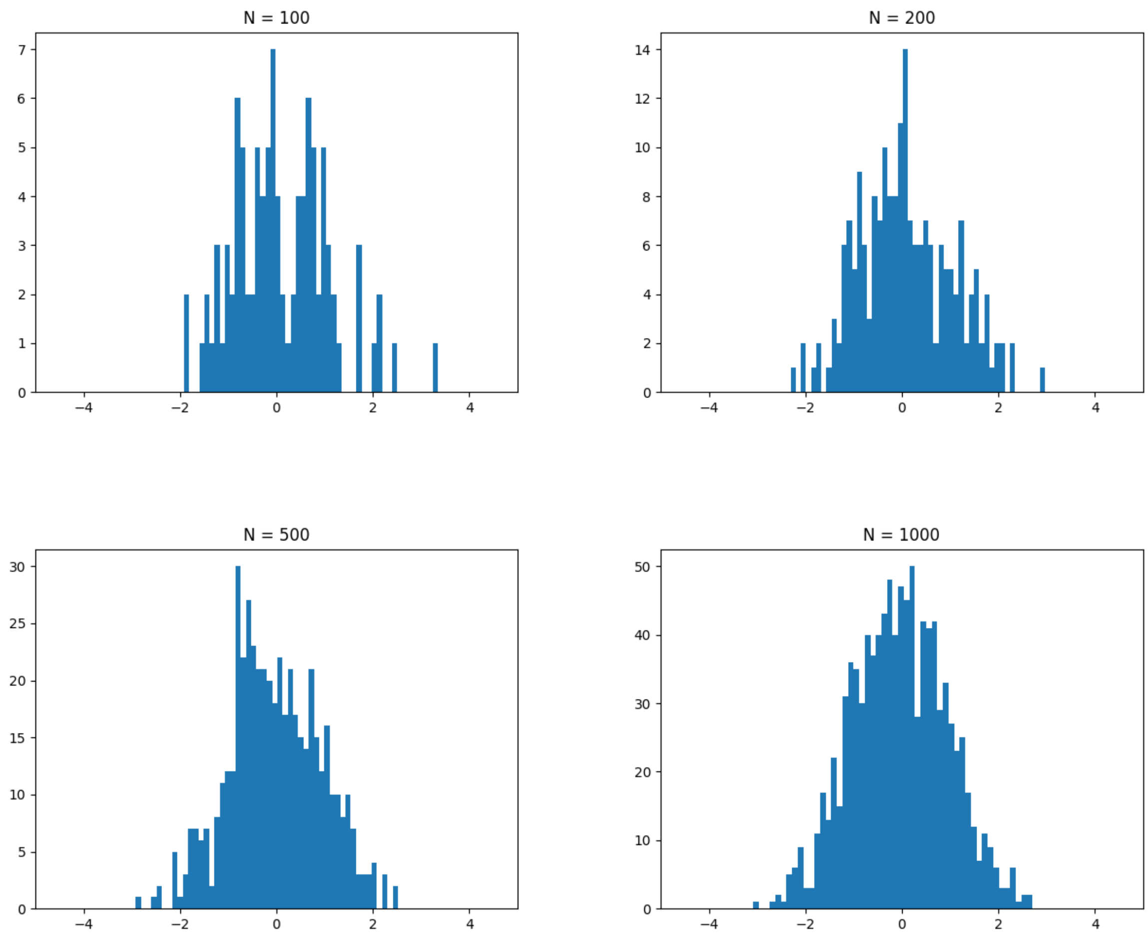
$\eta(0.63 + 0.27c) + \eta(0.09 - 0.09c) = 1$

$\eta = \frac{1}{0.72+0.18c}$

Finally, $P(x_{t+1} = clean|z = clean, u_t = vacuum\_clean) = \frac{0.63+0.27c}{0.72+0.18c}$, which represents the probabilty of the floor tile being clean after vacuum-cleaning.

**Problem 2. Sampling from a Distribution**

```
def f():
    u = random.random()
    if u == 0.5: return 0
    if u > 0.5:
        return min(sqrt((-pi*log(-4*u*u + 4*u))/2), 5)
    else:
        return max(-sqrt((-pi*log(-4*u*u + 4*u))/2), -5)

samples = [f() for i in range(n)]
```

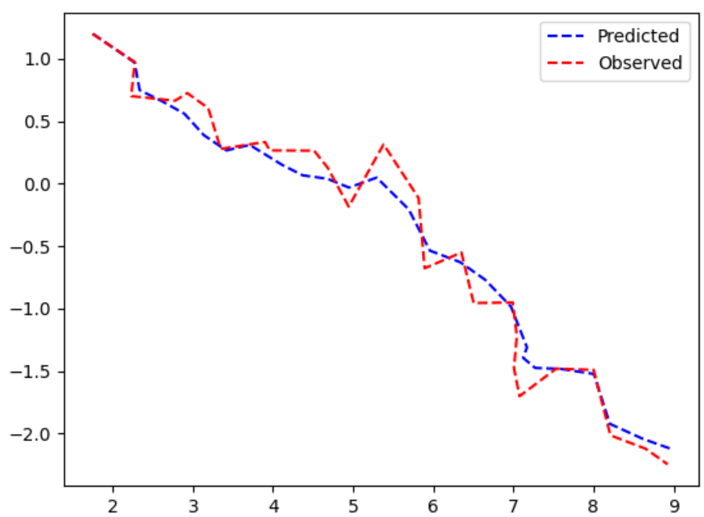


**Problem 3. Kalman Filter**

```python
def kalman(mu0, sig0, u, z):
    R = np.array([[0.01, 0.005], [0.005, 0.02]])
    Q = np.array([[0.0001, 0.00002], [0.00002, 0.0001]])

    # Predict
    predict_mu = mu0 + u
    predict_sig = sig0 + Q

    # Update
    K = predict_sig @ np.linalg.inv(predict_sig + R)
    mu1 = predict_mu + K @ (z - predict_mu)
    sig1 = (scaler*np.identity(2) - K) @ predict_sig

    return mu1, sig1
```

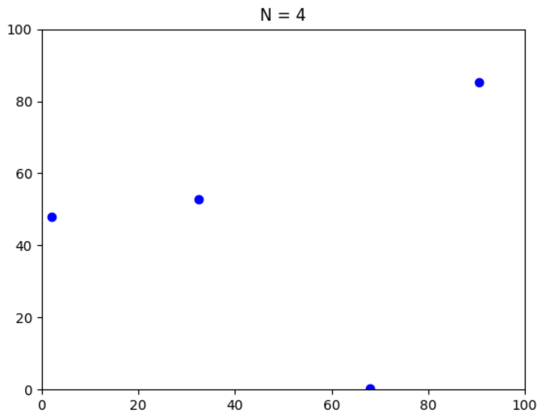
In the above algorithm, mu0 is the initial state vector of the form $\binom{x_0}{y_0}$ and sig0 is the state covariance matrix $\Sigma_0$. Both u and z correspond to the vectors $u_{k-1}$ and $z_k$ as indicated by the instructions.

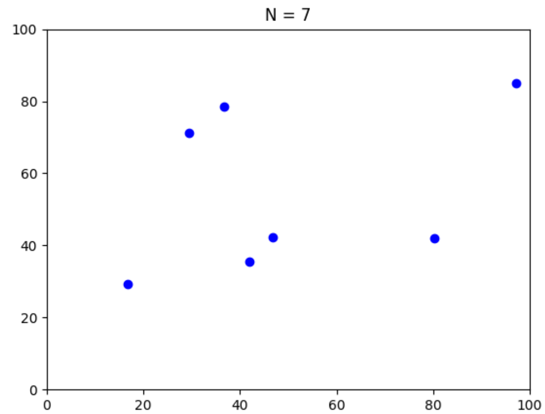
Processed Plots of Given Data ($\hat{x_0} = 1.75, \hat{y_0} = 1.2, \lambda = 1$):

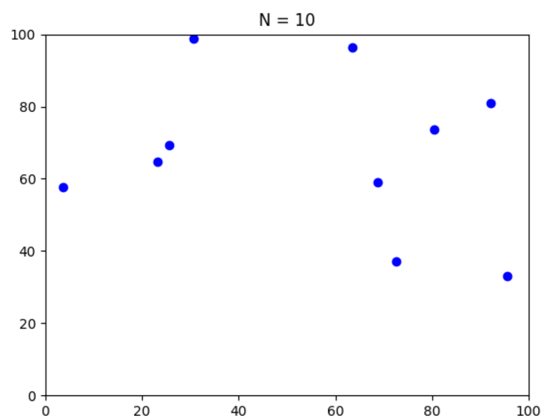


**Problem 4. Non-Linear Landmark-based Localization**

**A. Landmark Maps and Visualization**

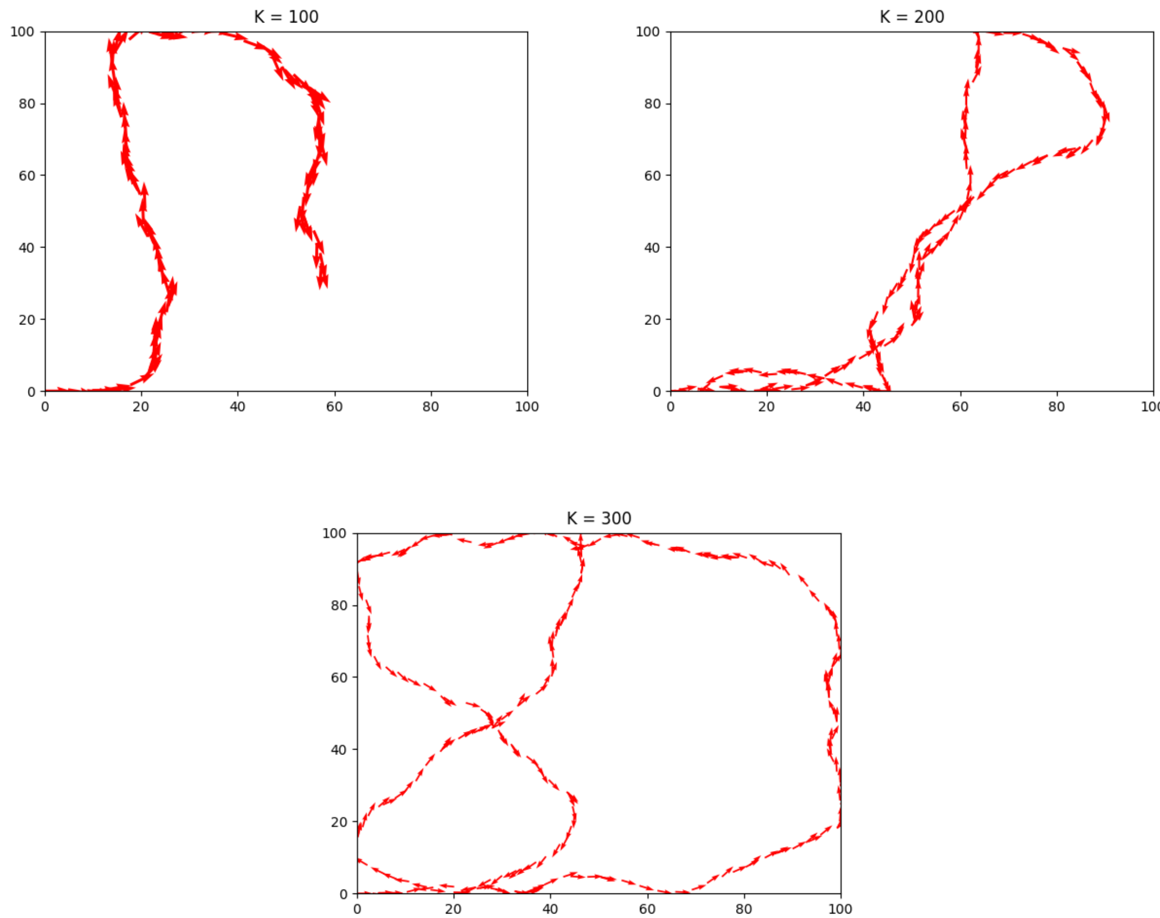



The landmarks were generated using random samples of $x$ and $y$ values within $[0, 100]$. A set of such landmarks can be generated by using the `visualizer.py` file and calling the `generate_landmark` function, which accepts an input `n` of the desired number of landmarks and a string input `id` for the generated file's ID. The function will print out the number of landmarks requested and the corresponding coordinates of the generated landmarks in the proper format. The function `visualize_landmarks` can be called to show a list of landmarks (an array of $(x, y)$ coordinates) and `parse_landmarks` can be called to read landmarks from an input file.

## B. "Ground Truth" Trajectories

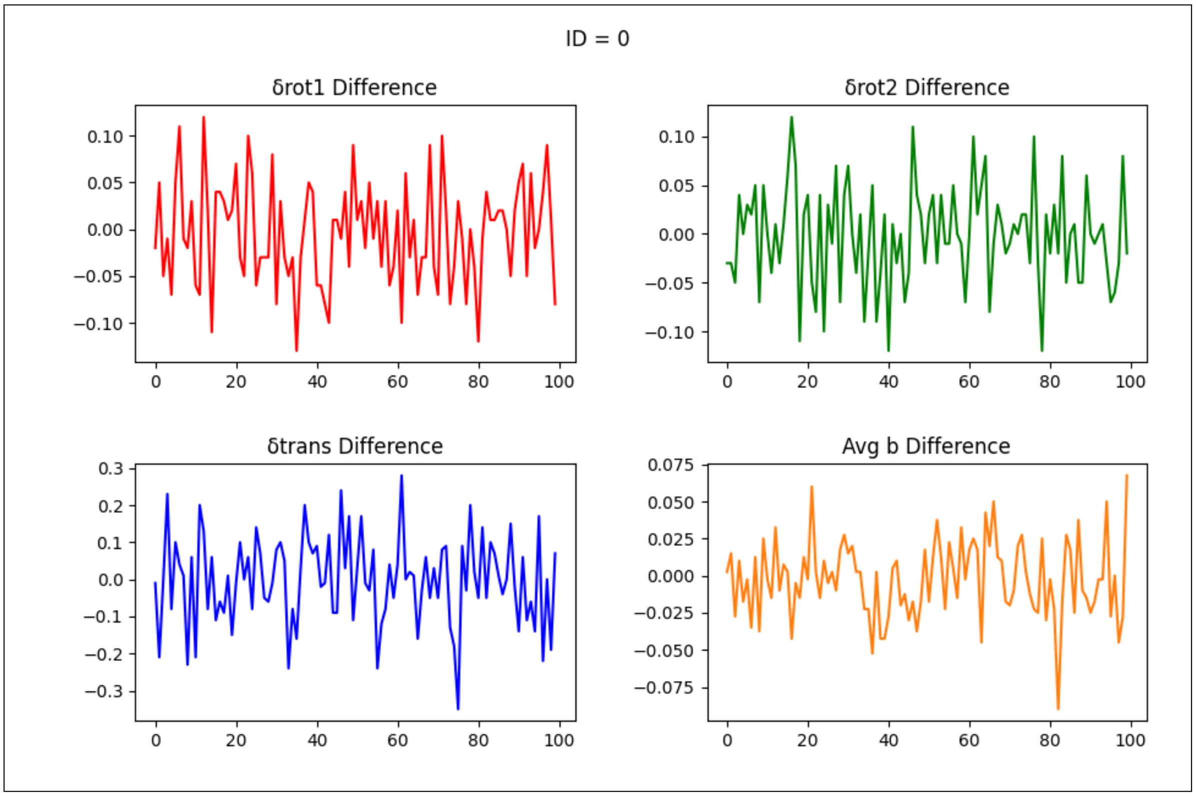


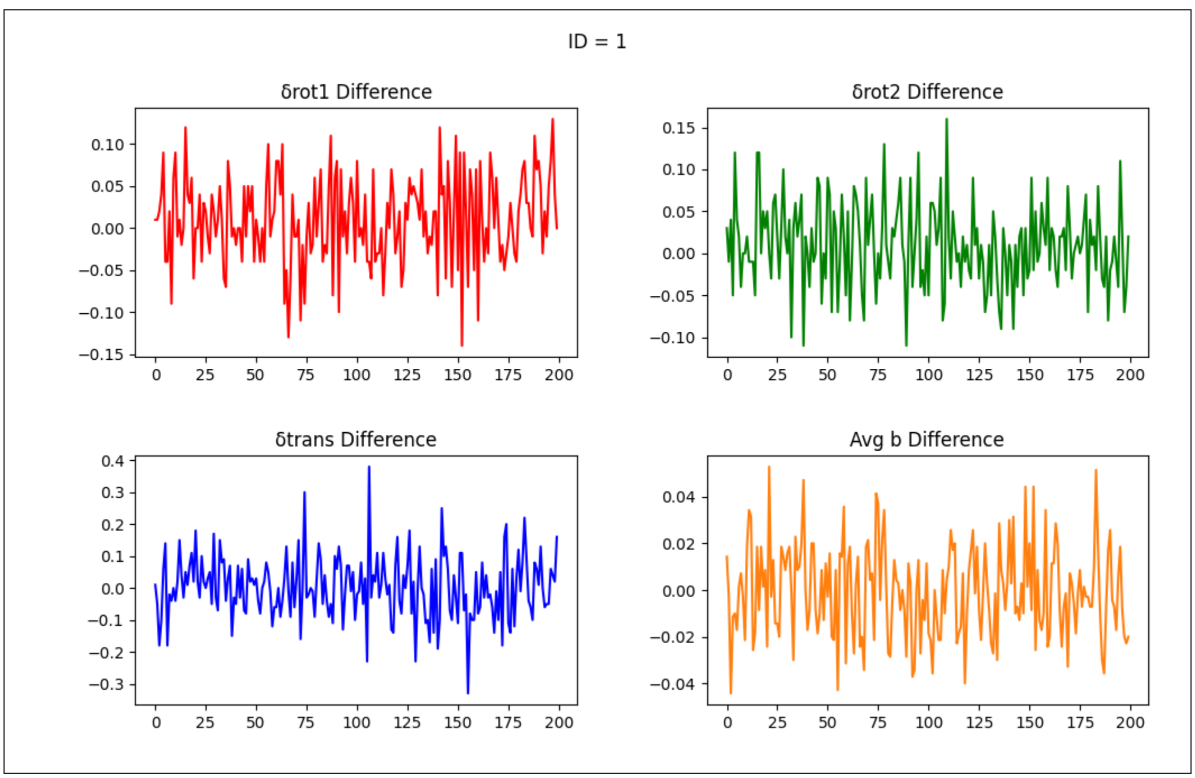

The "ground truth" trajectories are generated using the following random sample state transitions:

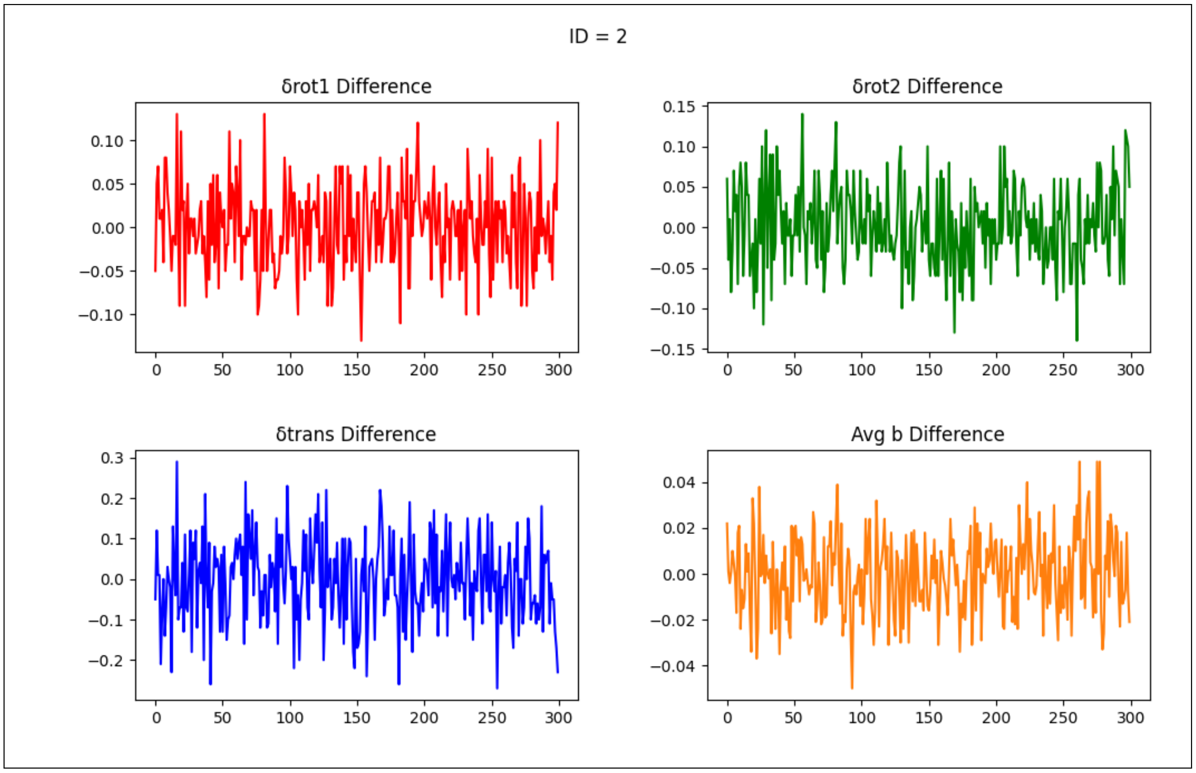
$$\delta_{rot1} = random(-\frac{\pi}{12}, \frac{\pi}{12})$$
$$\delta_{trans} = random(0, 5)$$
$$\delta_{rot1} = random(-\frac{\pi}{12}, \frac{\pi}{12})$$

Similar to the previous part, there is a `generate_ground_truths` function with an input `k` to generate `k` "ground truth" trajectories in addition to the original state and a string input `id` for the generated file's ID. Analogous to the previous section, there are also the `visualize_ground_truths` and `parse_ground_truths` functions.

## C. Input Data for Localization Challenges






In order to generate the measurement files, `generate_measurements` must be called. The function has 3 inputs: `id` (must be a string), `landmark_file` (must be an open file), and `ground_truth_file` (must be an open file).