Assignment_3_PF

May 6, 2025

0.1 Particle filter

#Process model

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ G(\mathbf{q})^{-1}(\mathbf{u}_\omega - \mathbf{b}_g) \\ \mathbf{g} + R(\mathbf{q})(\mathbf{u}_a - \mathbf{b}_a) \\ \mathbf{n}_{bg} \\ \mathbf{n}_{ba} \end{bmatrix}$$

0.1.1 Particle Filter Algorithm

Predict Step

1. **Particle Initialization**
   Particles are uniformly distributed within an approximate feasible volume of the environment map, representing the initial belief over position and orientation.
2. **Sampling IMU Noise**
   Gyroscope and accelerometer readings are perturbed with zero-mean Gaussian noise to model sensor uncertainty. These noisy inputs are used to compute the state derivative using the process model.
3. **State Propagation**
   Each particle's state is propagated forward in time by integrating the state derivative (e.g., velocity and orientation changes). This results in a new predicted set of particles representing the system's state after motion.

Update Step

1. **Observation Model**
   An observation model is applied to each predicted particle to simulate the expected measurement from that state (e.g., expected sensor reading or landmark observation).
2. **Measurement Likelihood**
   The actual sensor measurements are compared with the predicted observations to compute the likelihood (or error). An importance weight is assigned to each particle based on this likelihood.

3. **Weight Normalization**
   The importance weights are normalized across all particles to form a valid probability distribution.

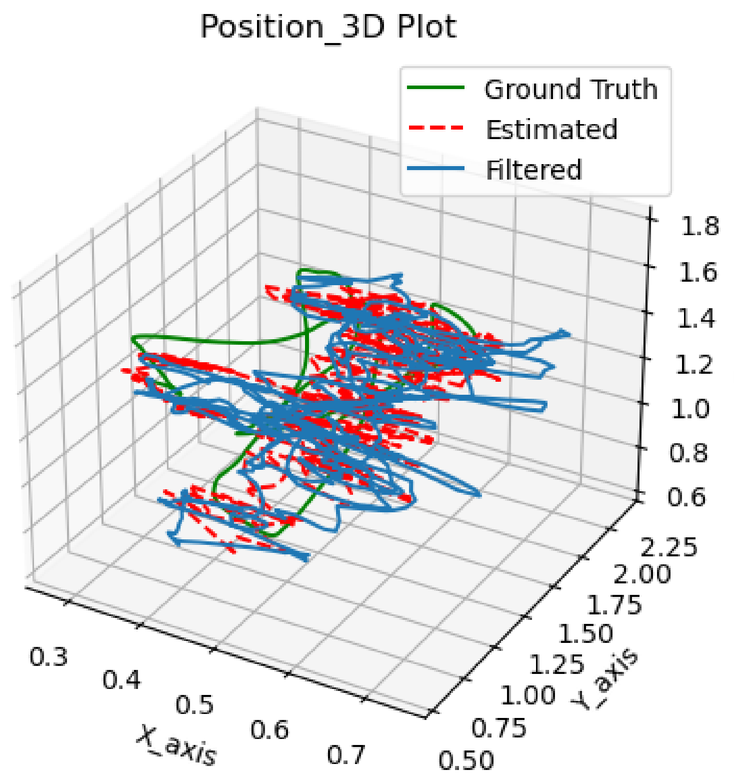
4. **State Estimation**
   The current position and orientation estimates are obtained as a weighted average over the particle set.

5. **Resampling**
   To avoid particle degeneracy, particles are resampled according to their normalized weights using **low-variance resampling**, focusing computational resources on high-likelihood regions of the state space.

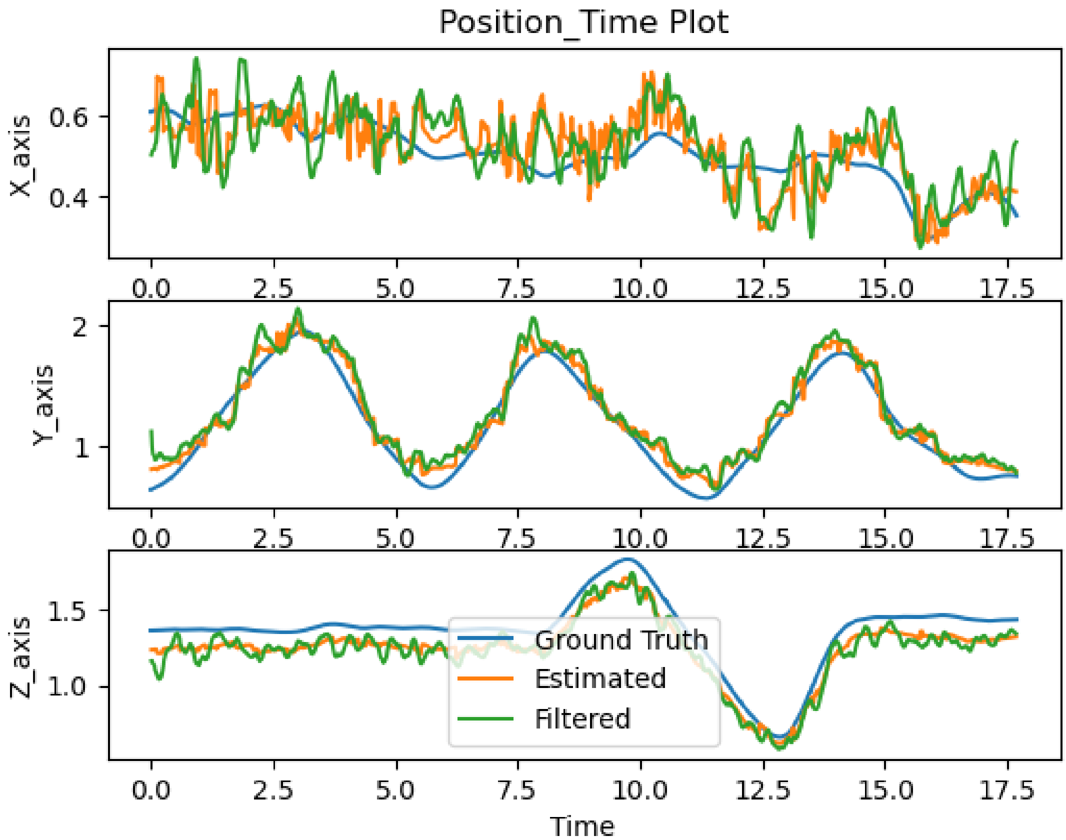
[13]:
```
#Compute Covariance

import compute_covariance as covariances
R = covariances.average_covariance()
```

0.1.2 Plots

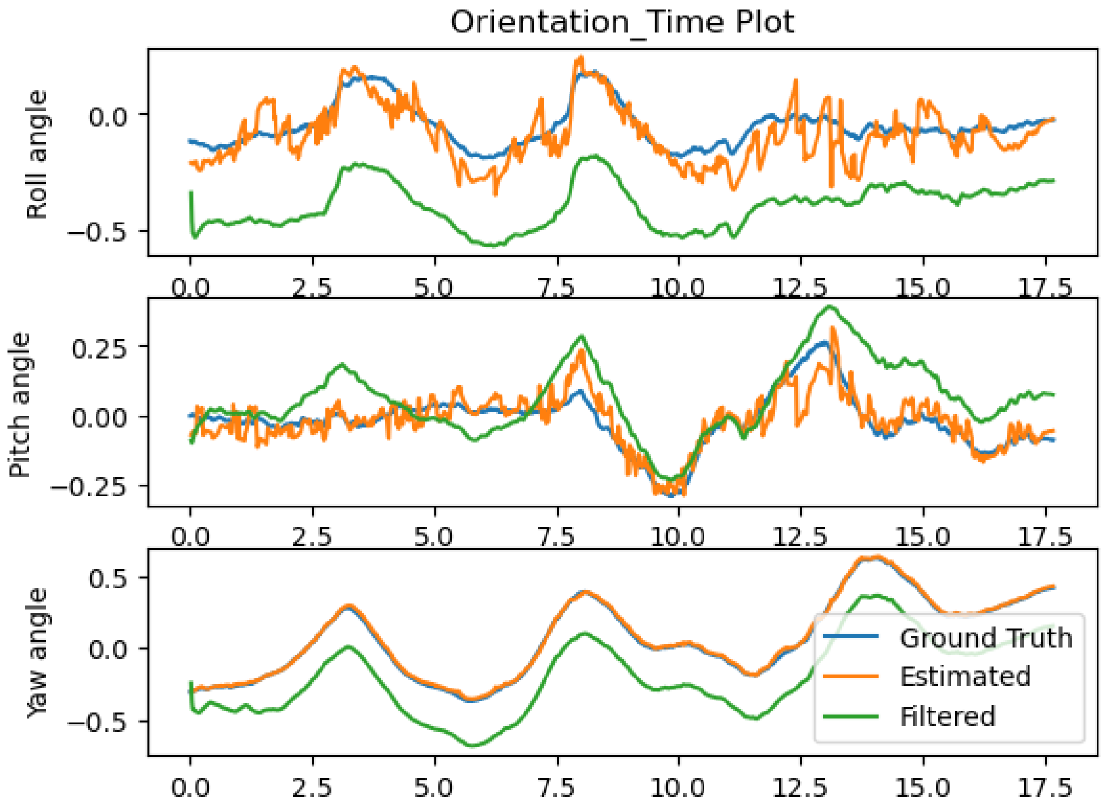
[19]:
```
#Simulate the results and plot

from simulation_PF import simulate
import plots


estimated, est_t, ground_truth, ground_truth_t, filt_data, partcl_hist, exe_t =␣
 ↪simulate(r"data\data\studentdata1.mat", 1000, "weighted_avg", R, Qa=90, Qg=0.
 ↪1)
plots.plot(estimated, est_t, ground_truth, ground_truth_t, filt_data)
```



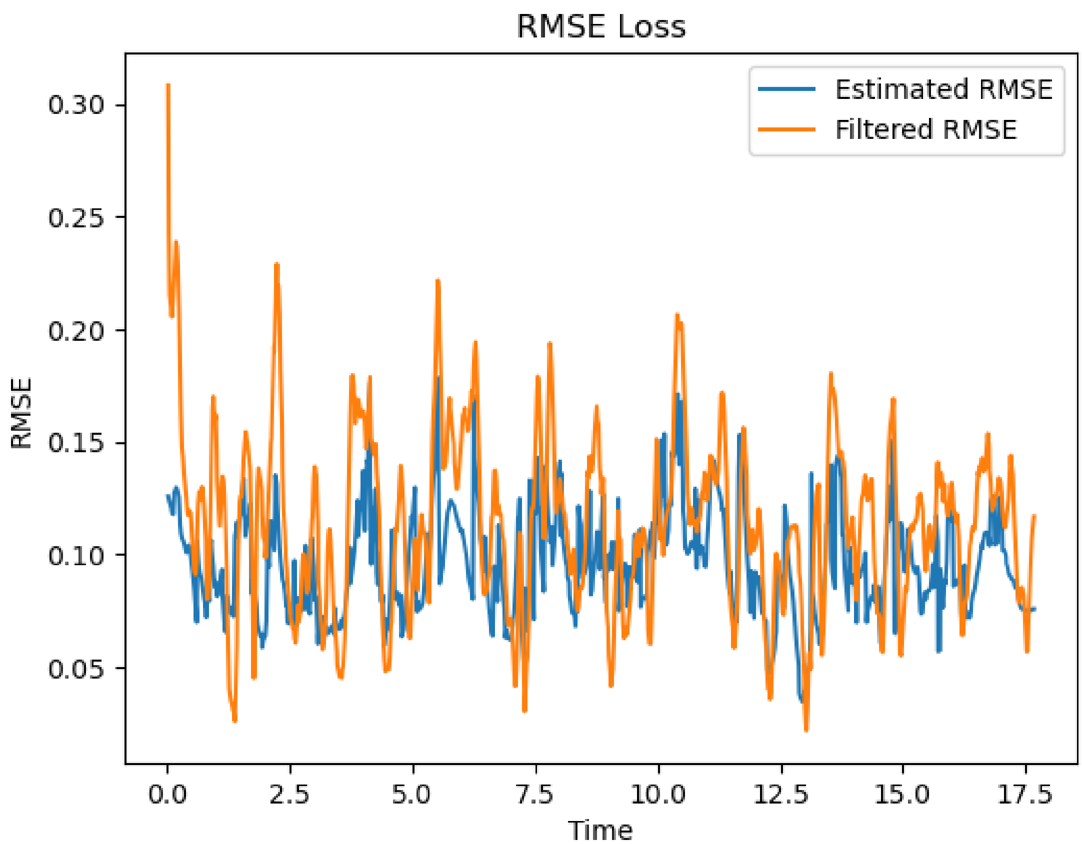
Position_3D Plot
Ground Truth
Estimated
Filtered
1.8
1.6
1.4
1.2
1.0
0.8
0.6
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
Y_axis
0.3
0.4
0.5
0.6
0.7
X_axis



Position_Time Plot
X_axis
0.6
0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Y_axis
2
1
Z_axis
1.5
1.0
Ground Truth
Estimated
Filtered
Time

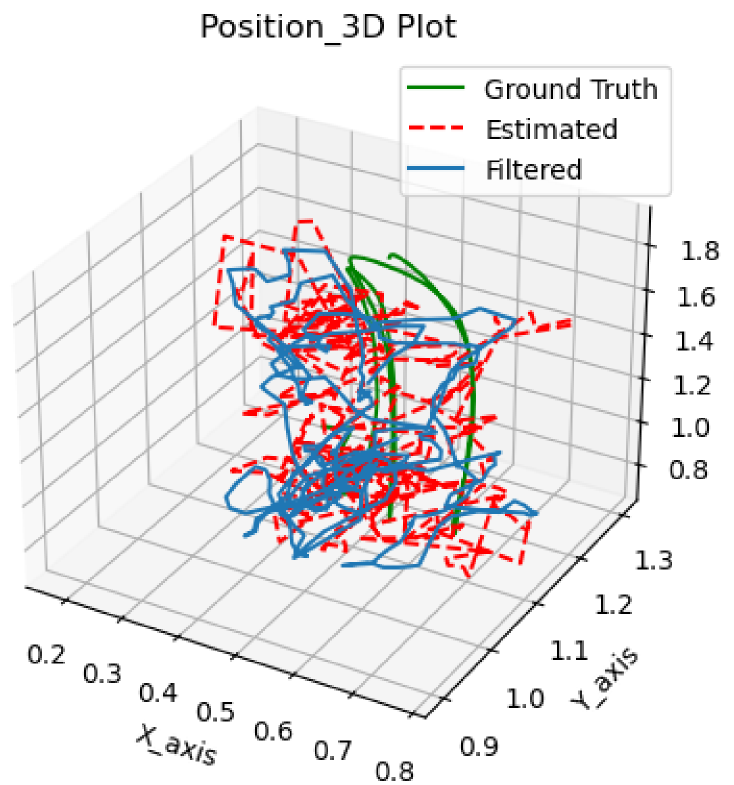


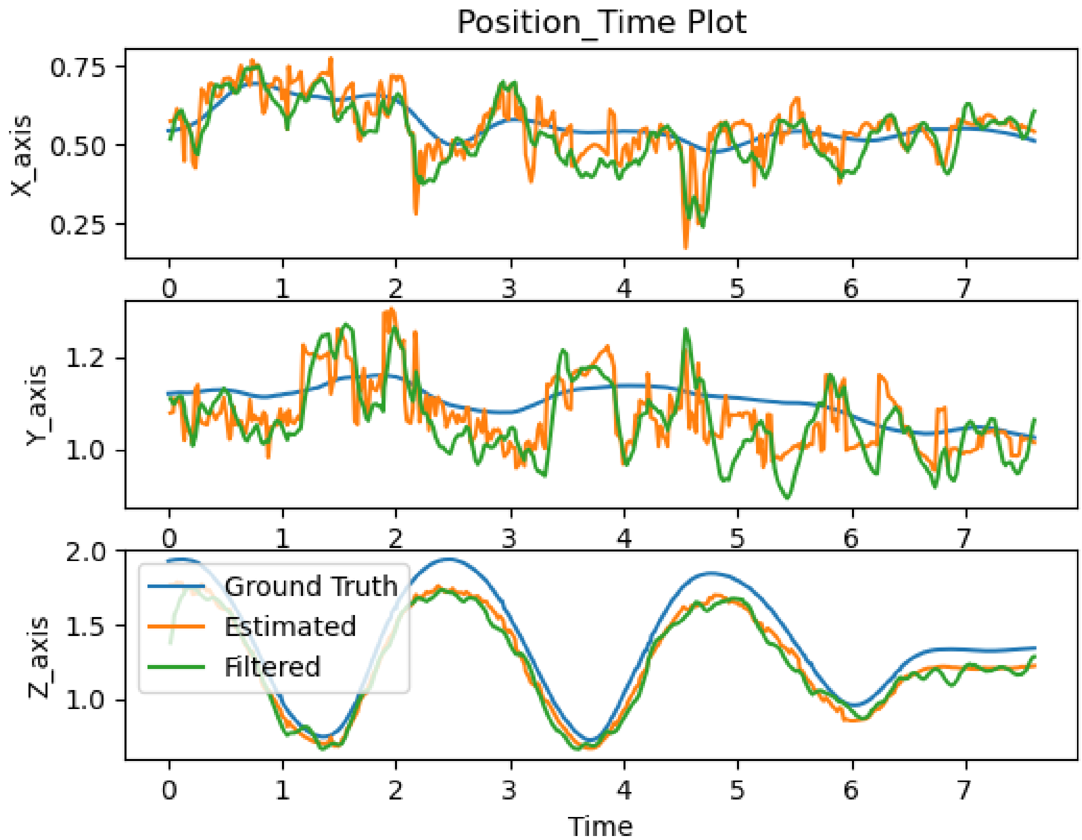
Orientation_Time Plot
Roll angle
Pitch angle
Yaw angle
Ground Truth
Estimated
Filtered
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0.25
0.00
−0.25
0.5
−0.5



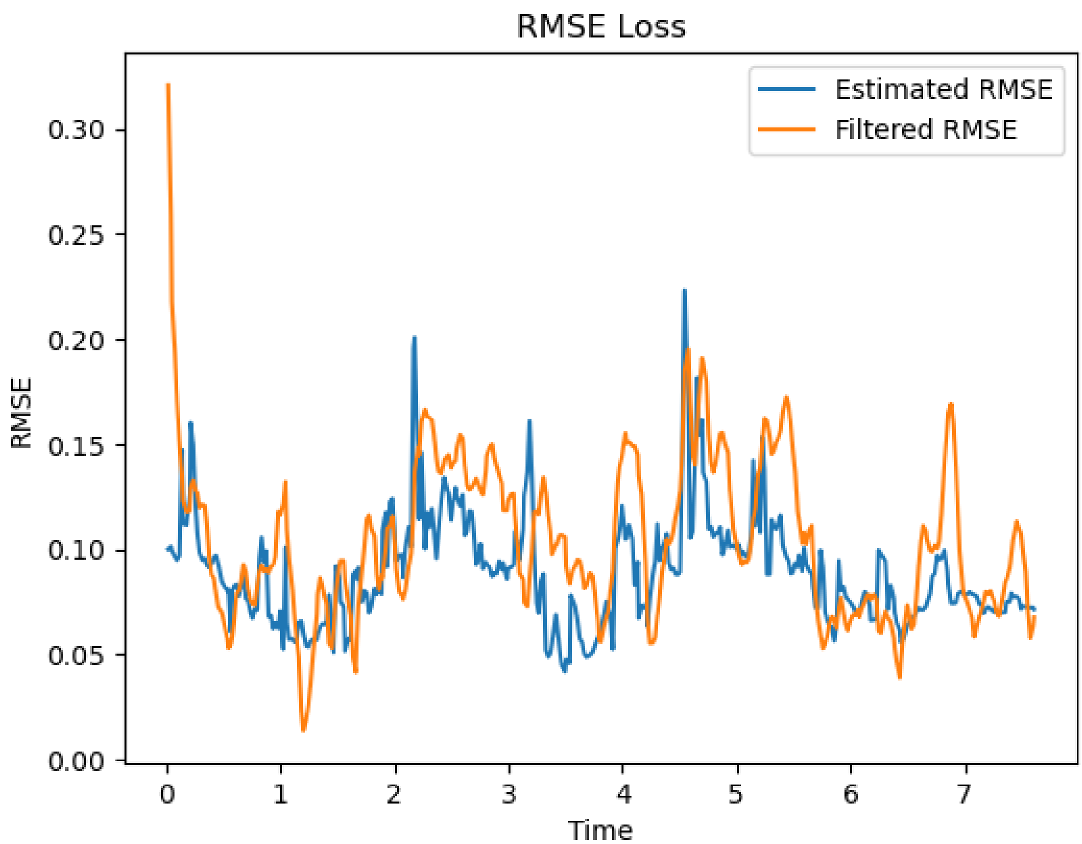
[20]:

```
#Simulate the results and plot

from simulation_PF import simulate
import plots

estimated, est_t, ground_truth, ground_truth_t, filt_data, partcl_hist, exe_t =
  ↪simulate(r"data\data\studentdata2.mat", 1000, "weighted_avg", R, Qa=100,
  ↪Qg=0.1)
plots.plot(estimated, est_t, ground_truth, ground_truth_t, filt_data)
```

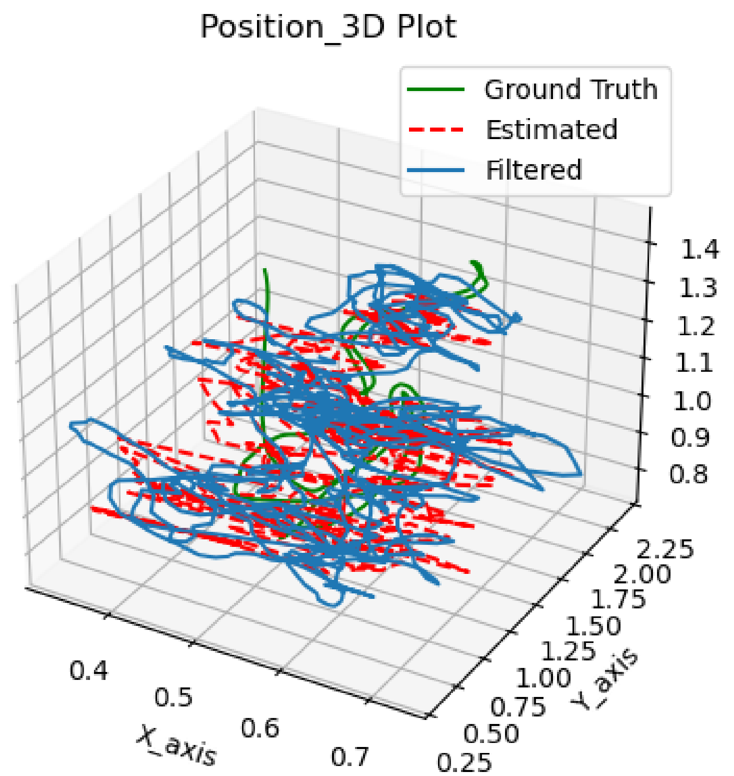

Position_3D Plot
Ground Truth
Estimated
Filtered
0.2
0.3
0.4
0.5
0.6
0.7
0.8
X_axis
0.9
1.0
1.1
1.2
1.3
Y_axis
0.8
1.0
1.2
1.4
1.6
1.8

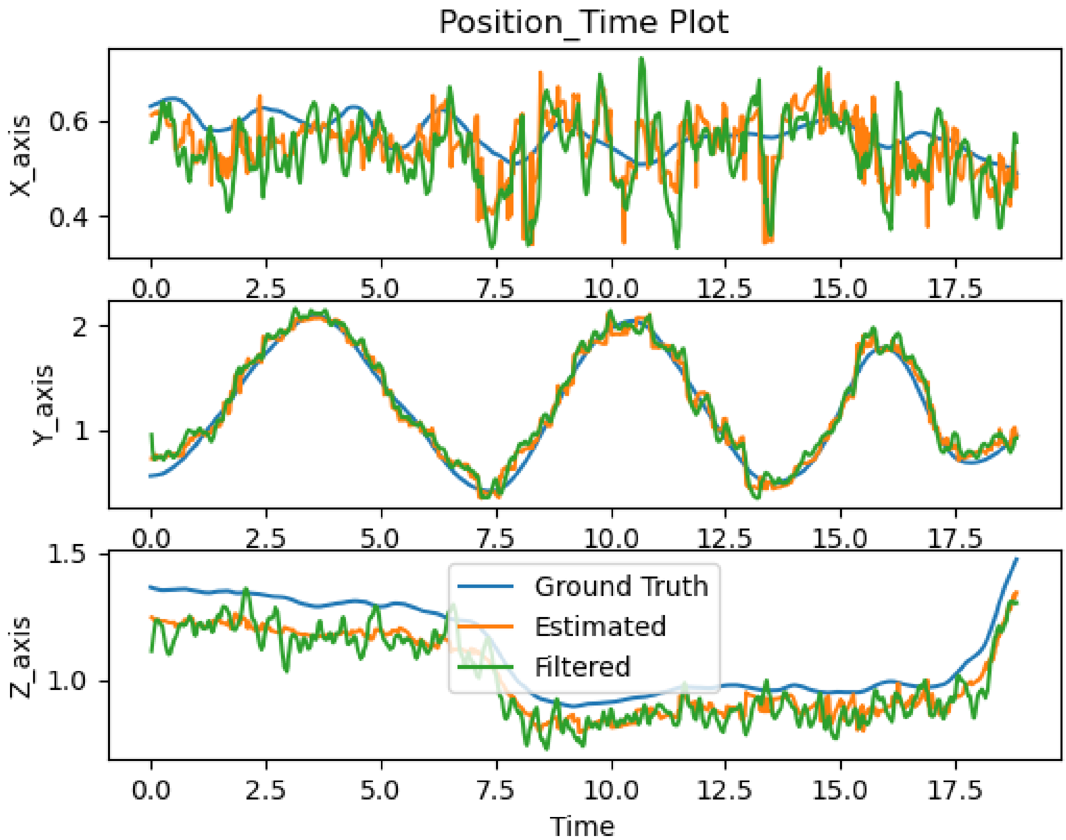

Position_Time Plot
X_axis
0.75
0.50
0.25
Y_axis
1.2
1.0
Z_axis
2.0
1.5
1.0
Ground Truth
Estimated
Filtered
0
1
2
3
4
5
6
7
Time

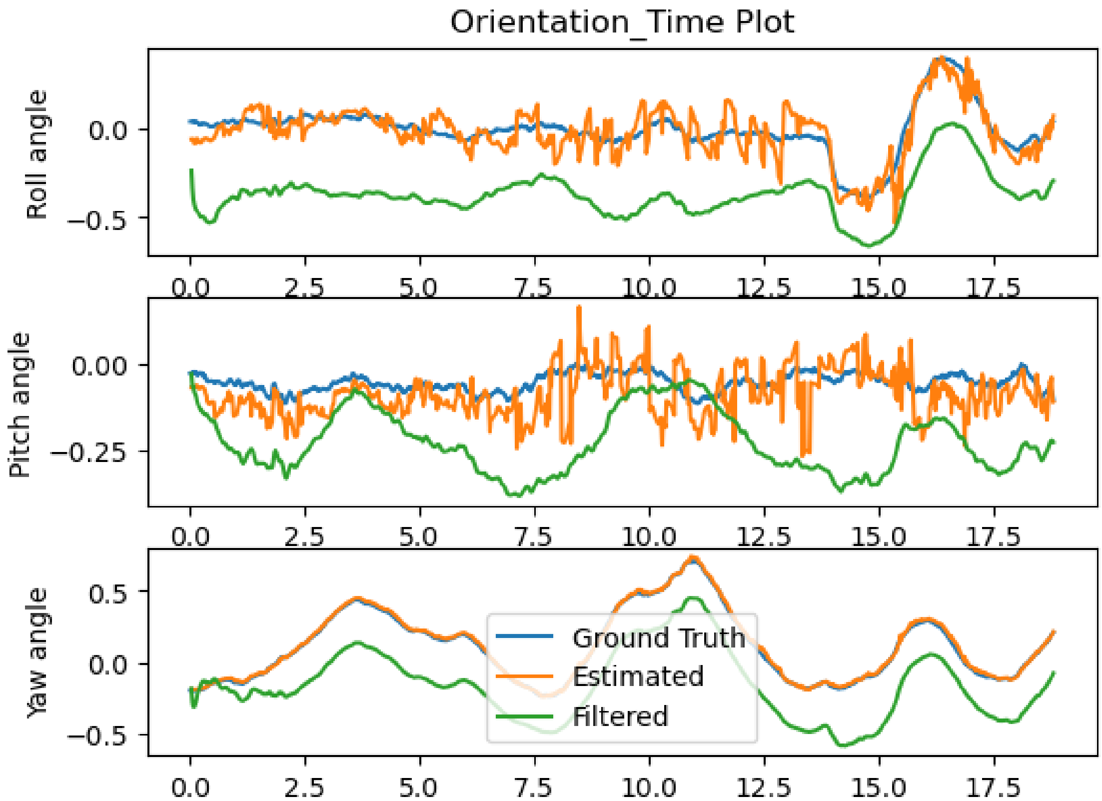

Orientation_Time Plot
Roll angle
0.25
0.00
−0.25
0
1
2
3
4
5
6
7
Pitch angle
0.0
−0.5
0
1
2
3
4
5
6
7
Yaw angle
0.5
0.0
−0.5
Ground Truth
Estimated
Filtered
0
1
2
3
4
5
6
7

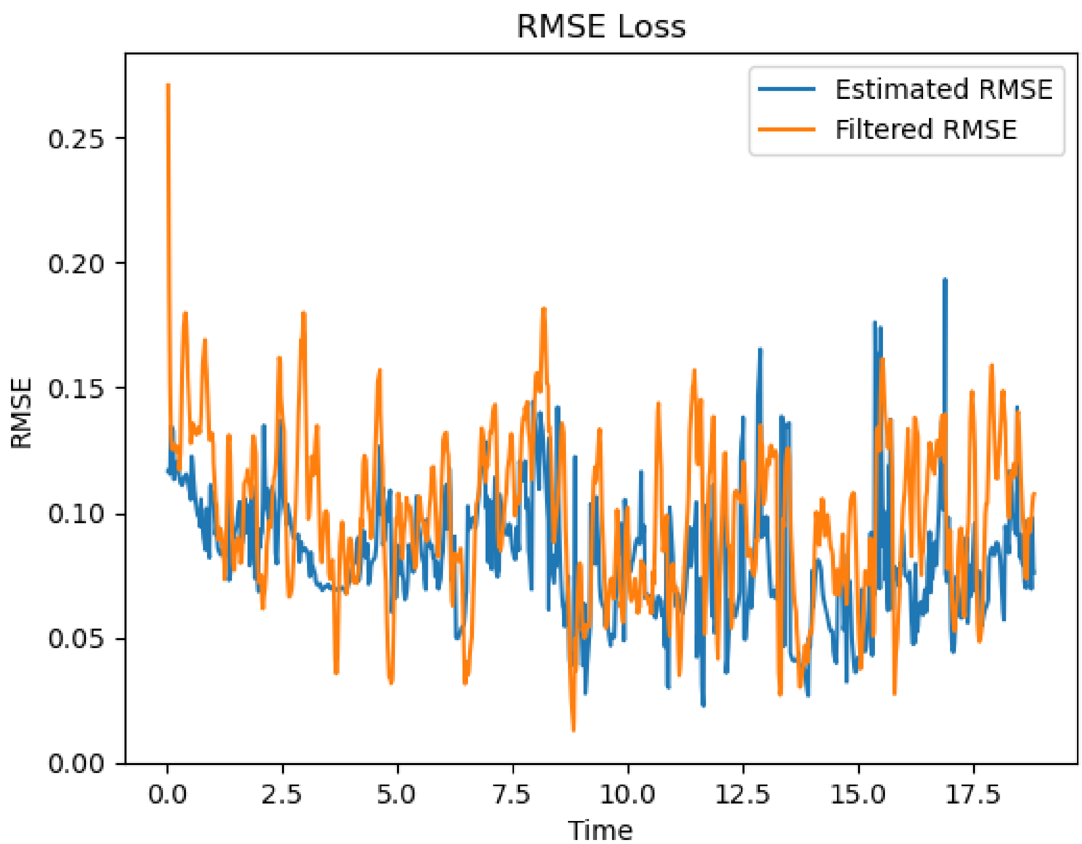


[21]:
```
#Simulate the results and plot

from simulation_PF import simulate
import plots

estimated, est_t, ground_truth, ground_truth_t, filt_data, partcl_hist, exe_t =␣
    ↪simulate(r"data\data\studentdata3.mat", 1000, "weighted_avg", R, Qa=110,␣
    ↪Qg=0.1)
plots.plot(estimated, est_t, ground_truth, ground_truth_t, filt_data)
```


Position_3D Plot
Ground Truth
Estimated
Filtered
1.4
1.3
1.2
1.1
1.0
0.9
0.8
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
Y_axis
0.4
0.5
0.6
0.7
X_axis

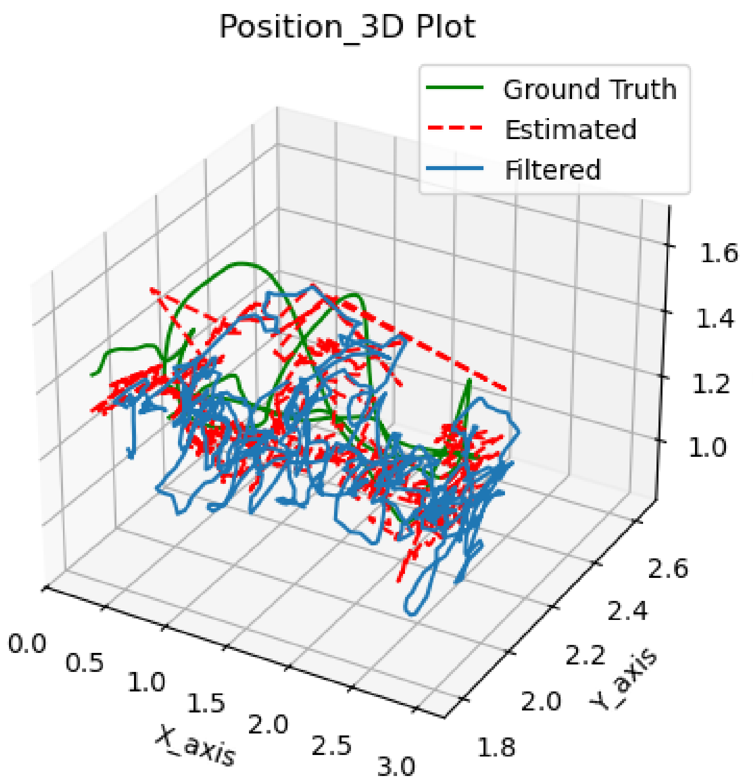

Position_Time Plot
X_axis
0.6
0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Y_axis
2
1
Z_axis
1.5
1.0
Ground Truth
Estimated
Filtered
Time

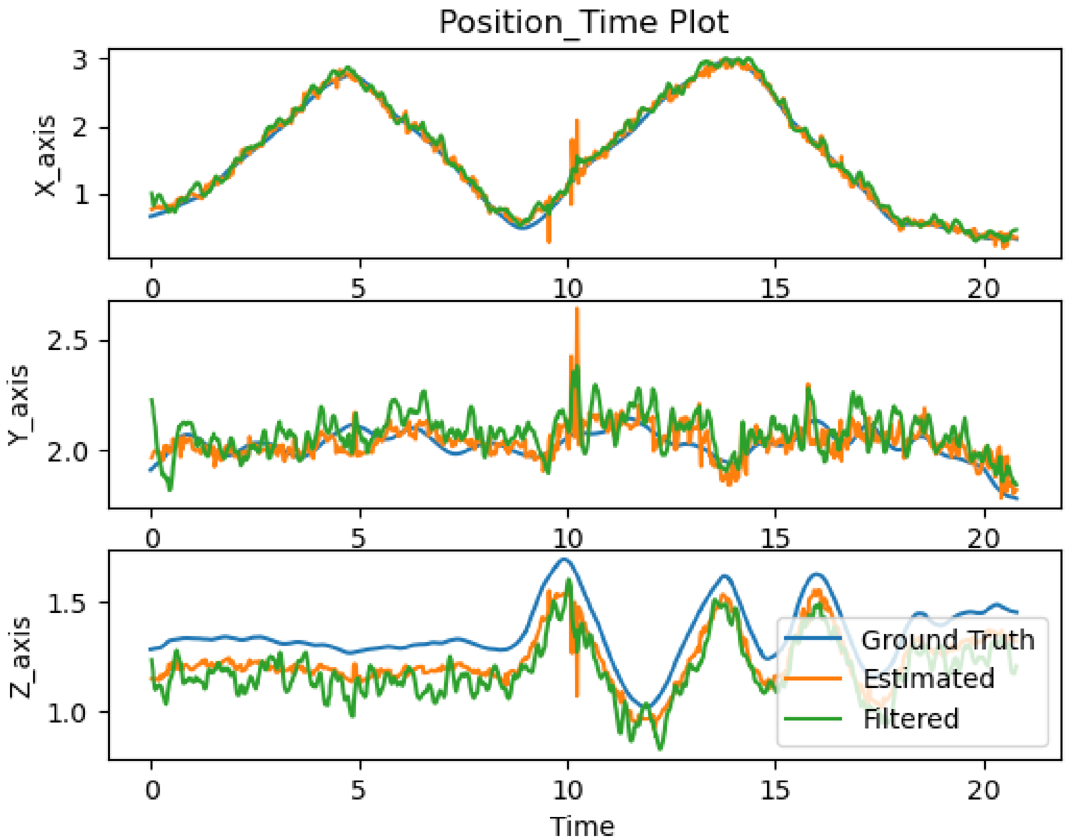

Orientation_Time Plot
Roll angle
0.0
−0.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Pitch angle
0.00
−0.25
Yaw angle
0.5
0.0
−0.5
Ground Truth
Estimated
Filtered

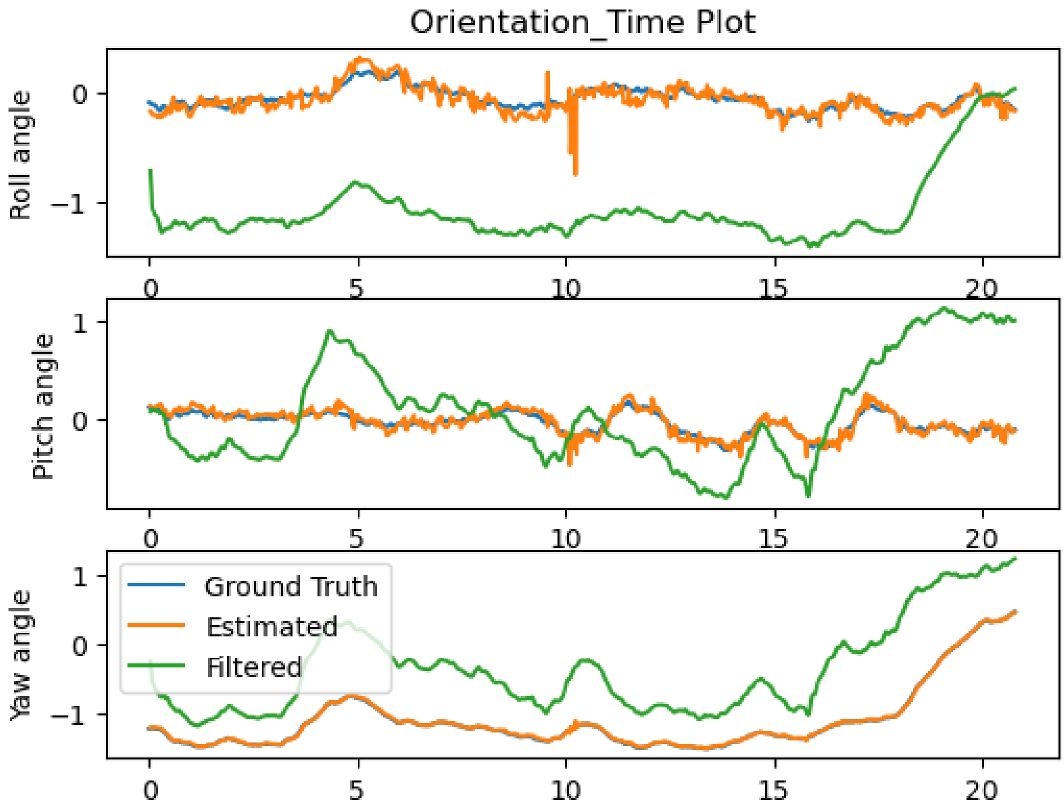


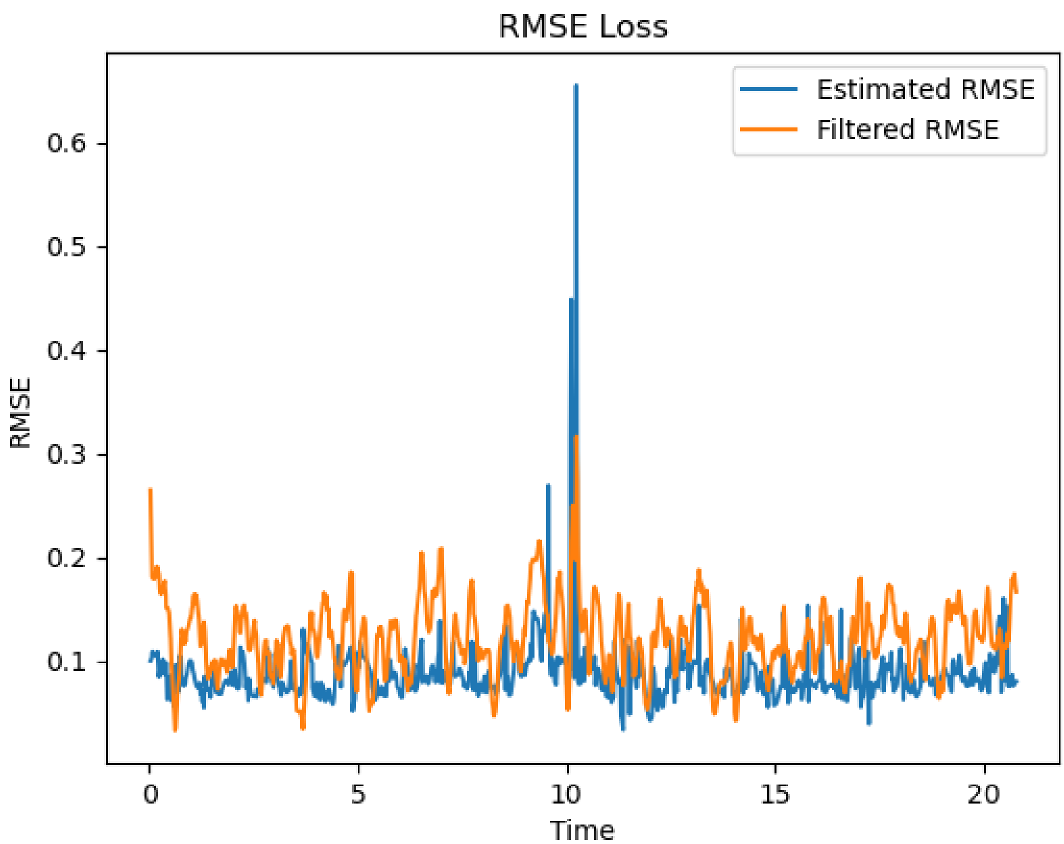
[22]:

```
#Simulate the results and plot

from simulation_PF import simulate
import plots

estimated, est_t, ground_truth, ground_truth_t, filt_data, partcl_hist, exe_t =
    simulate(r"data\data\studentdata4.mat", 1000, "weighted_avg", R, Qa=120,
    Qg=0.1)
plots.plot(estimated, est_t, ground_truth, ground_truth_t, filt_data)
```

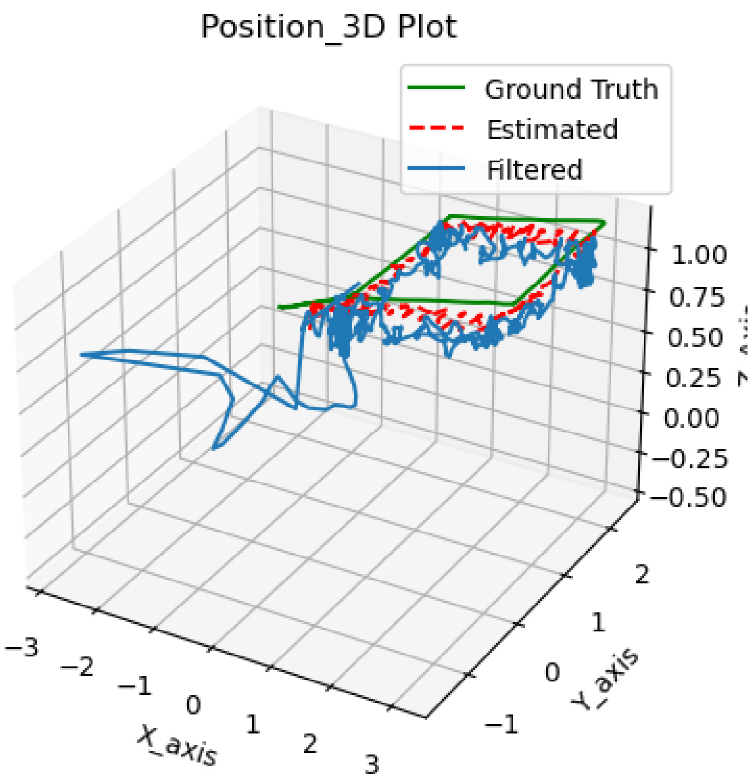

Position_3D Plot
Ground Truth
Estimated
Filtered
1.6
1.4
1.2
1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
X_axis
1.8
2.0
2.2
2.4
2.6
Y_axis

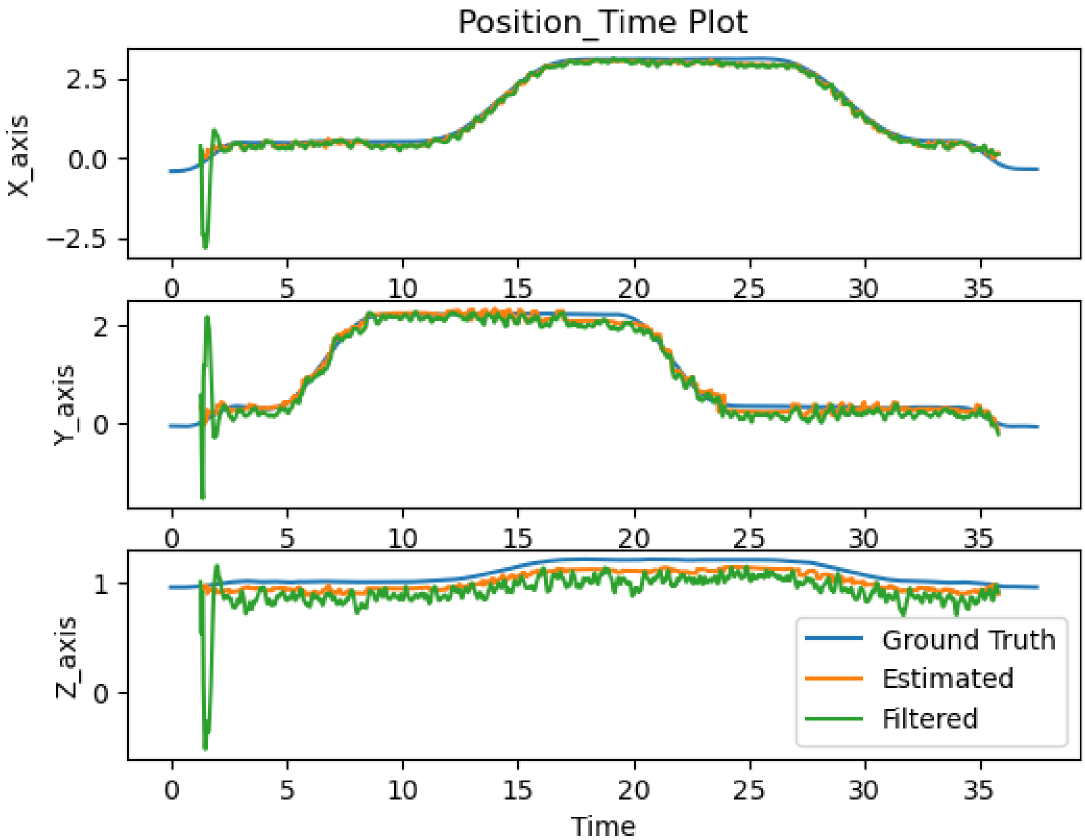

Position_Time Plot
X_axis
Y_axis
Z_axis
Time
Ground Truth
Estimated
Filtered

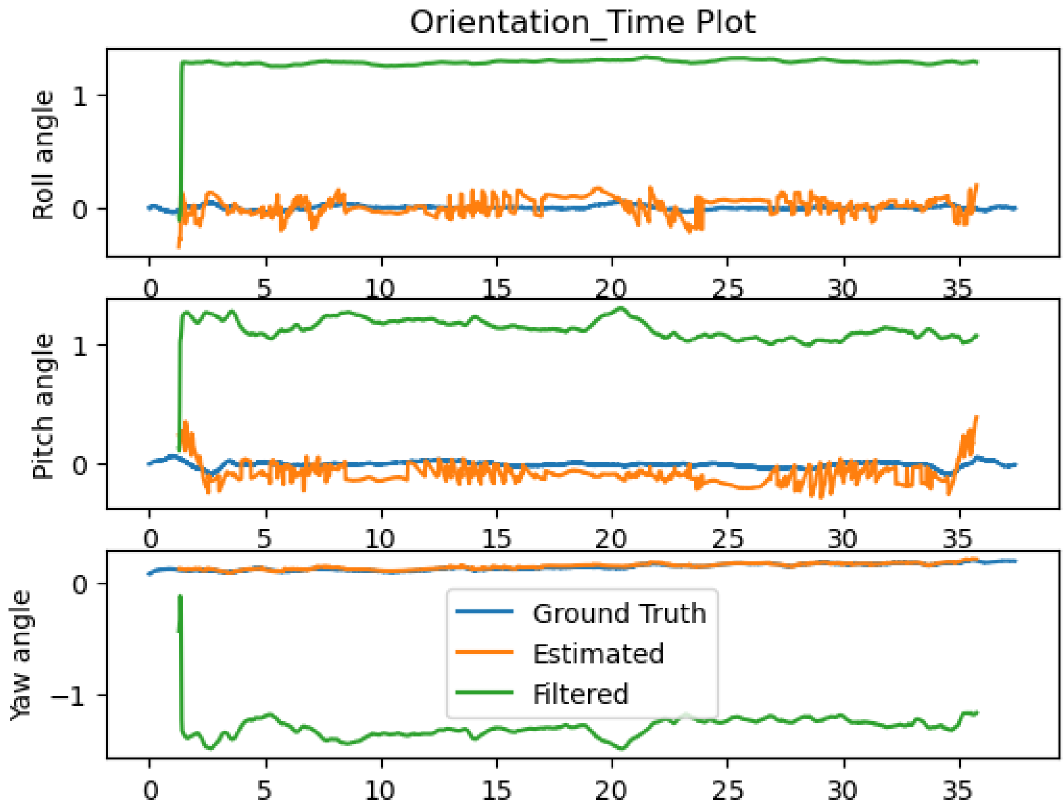

Orientation_Time Plot
Roll angle
0
−1
0
5
10
15
20
Pitch angle
1
0
0
5
10
15
20
Yaw angle
1
0
−1
Ground Truth
Estimated
Filtered
0
5
10
15
20



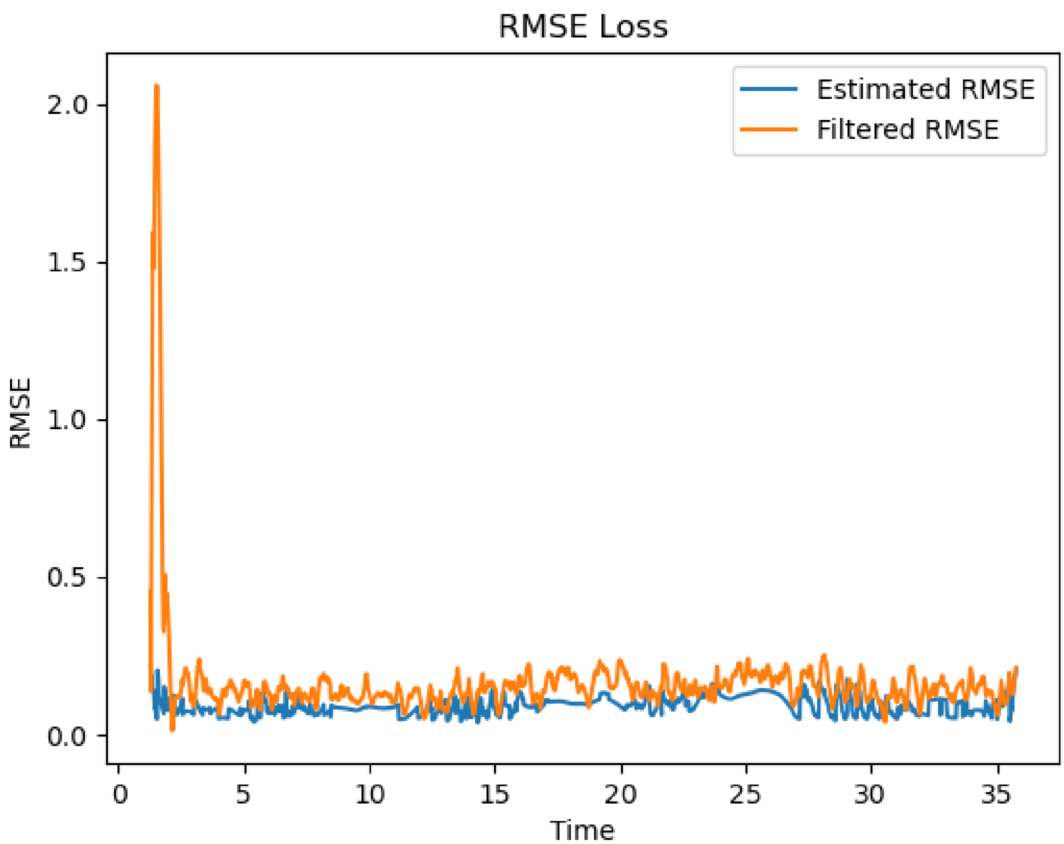
[23]:
```
#Simulate the results and plot

from simulation_PF import simulate
import plots

estimated, est_t, ground_truth, ground_truth_t, filt_data, partcl_hist, exe_t =␣
↪simulate(r"data\data\studentdata5.mat", 1000, "weighted_avg", R, Qa=100,␣
↪Qg=0.01)
plots.plot(estimated, est_t, ground_truth, ground_truth_t, filt_data)
```


Position_3D Plot
Ground Truth
Estimated
Filtered
X_axis
Y_axis

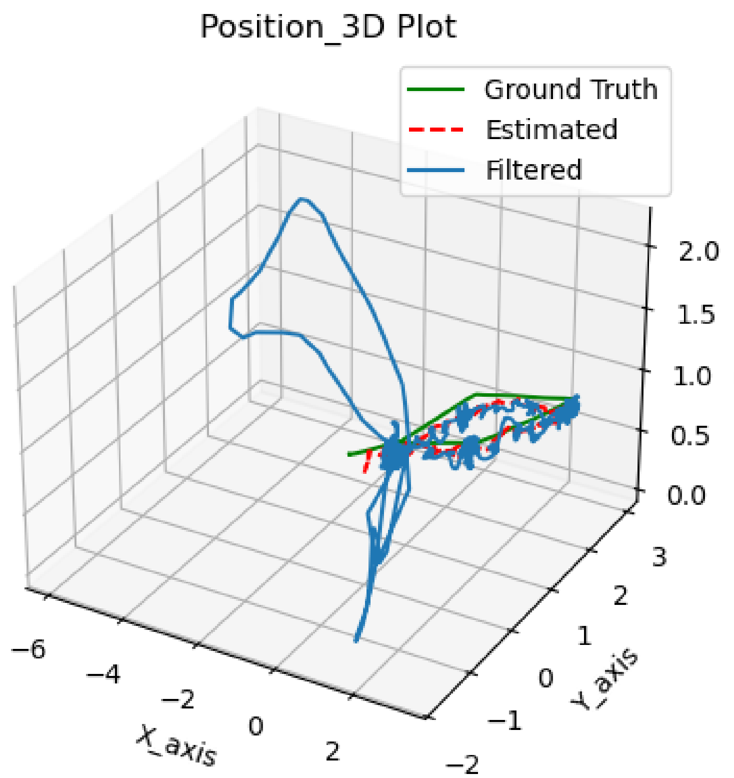

Position_Time Plot
X_axis
2.5
0.0
−2.5
0
5
10
15
20
25
30
35
Y_axis
2
0
Z_axis
1
0
Ground Truth
Estimated
Filtered
Time

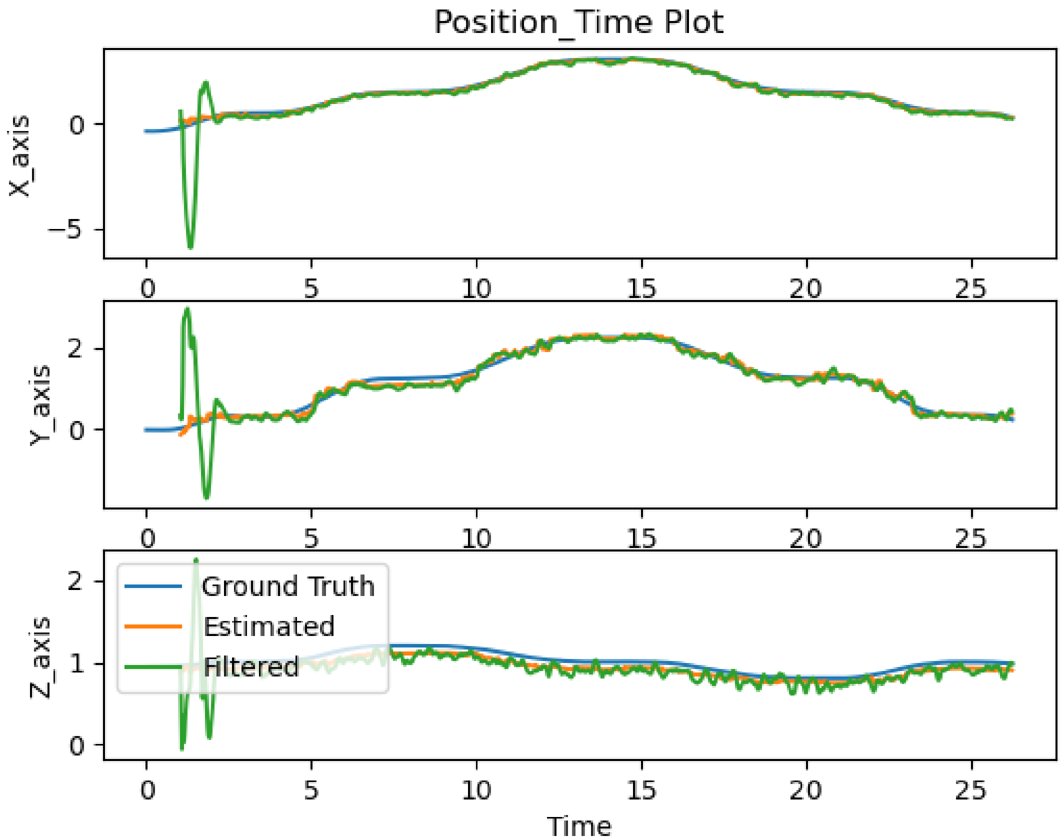


Orientation_Time Plot
Roll angle
Pitch angle
Yaw angle
Ground Truth
Estimated
Filtered

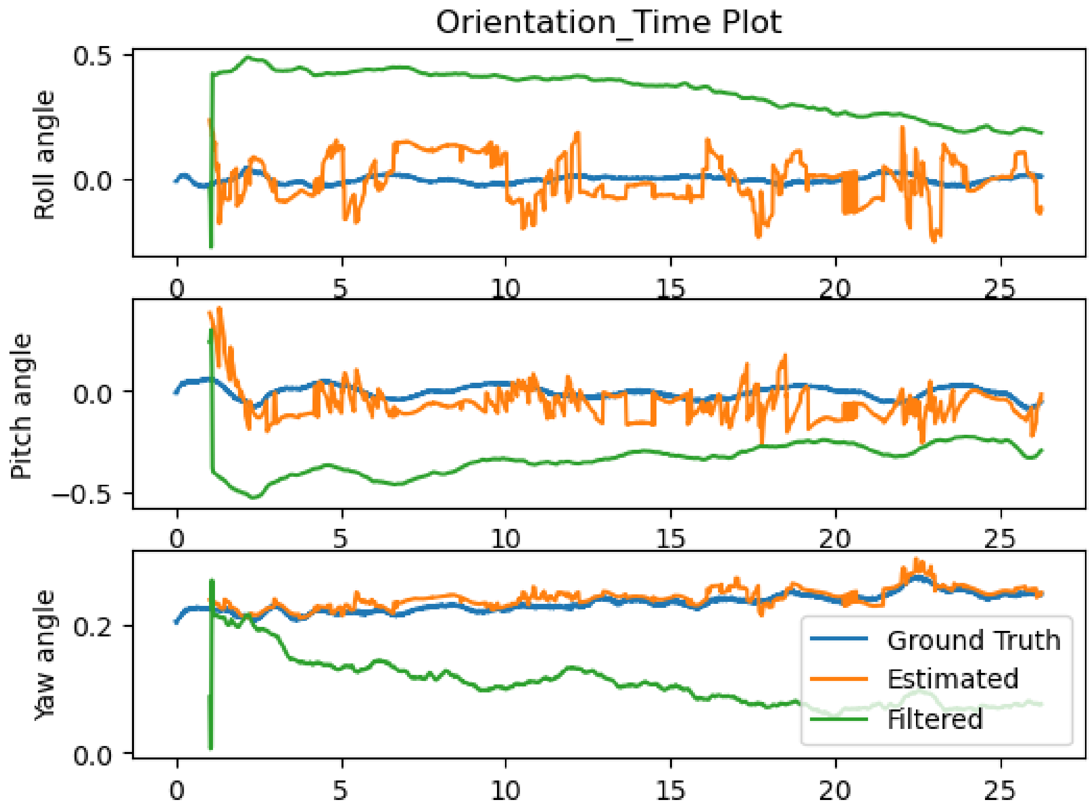


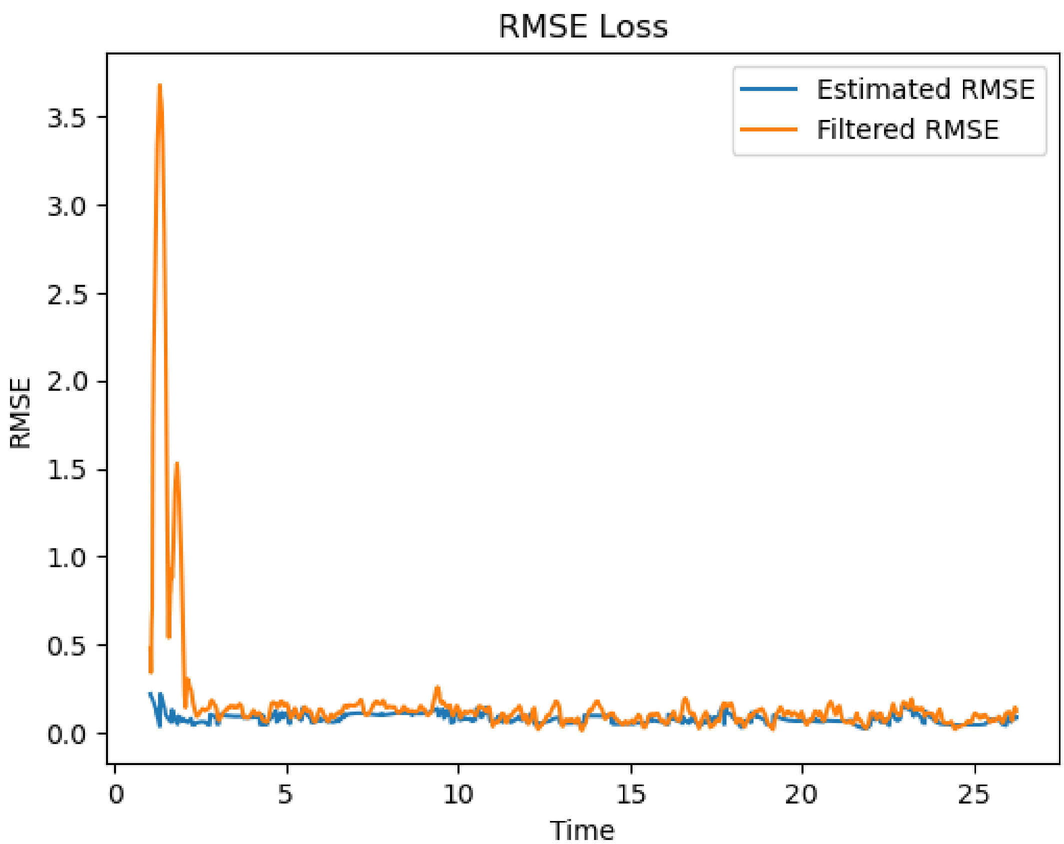
[24]:
```
#Simulate the results and plot

from simulation_PF import simulate
import plots

estimated, est_t, ground_truth, ground_truth_t, filt_data, partcl_hist, exe_t =␣
 ↪simulate(r"data\data\studentdata6.mat", 1000, "weighted_avg", R, Qa=110,␣
 ↪Qg=0.1)
plots.plot(estimated, est_t, ground_truth, ground_truth_t, filt_data)
```


Position_3D Plot
Ground Truth
Estimated
Filtered
2.0
1.5
1.0
0.5
0.0
−6
−4
−2
0
2
X_axis
−2
−1
0
1
2
3
Y_axis



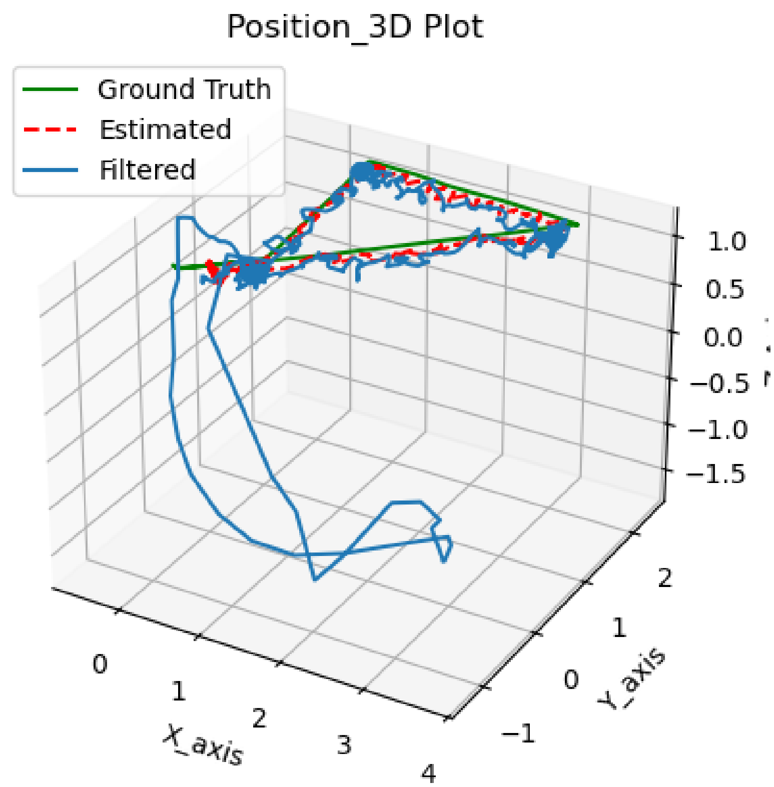
Position_Time Plot
X_axis
Y_axis
Z_axis
Time
Ground Truth
Estimated
Filtered

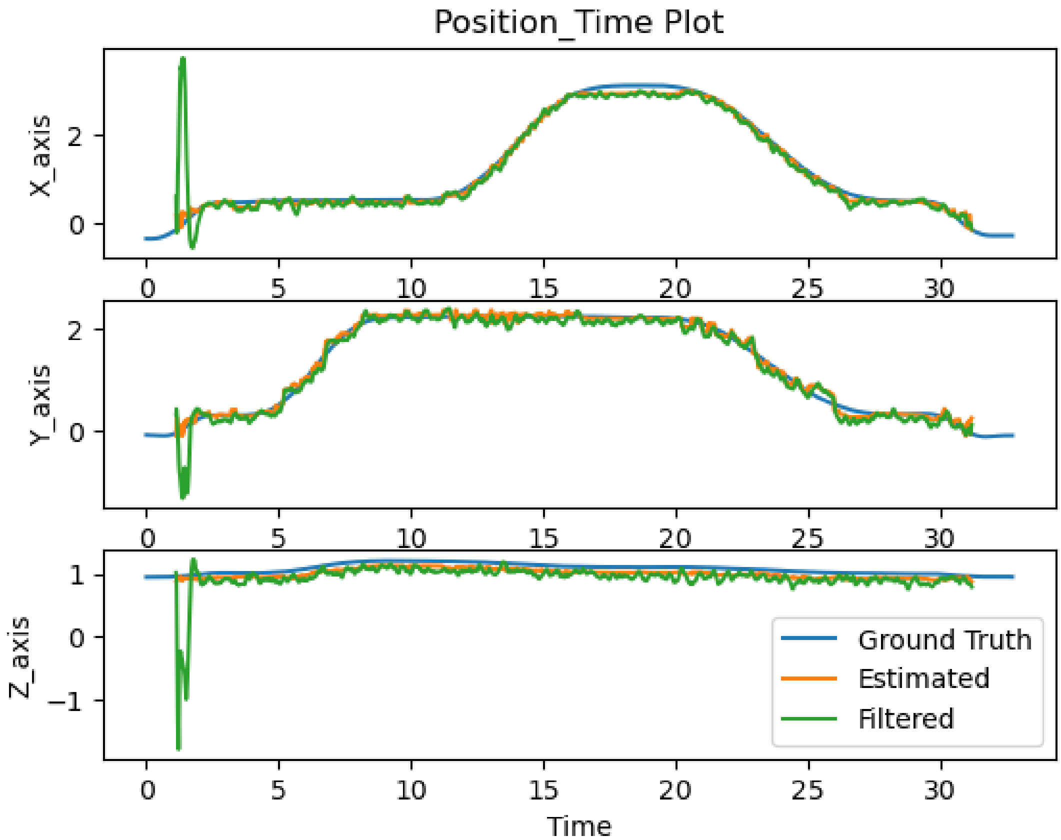

Orientation_Time Plot
Roll angle
Pitch angle
Yaw angle
Ground Truth
Estimated
Filtered
0.5
0.0
−0.5
0.2
0
5
10
15
20
25



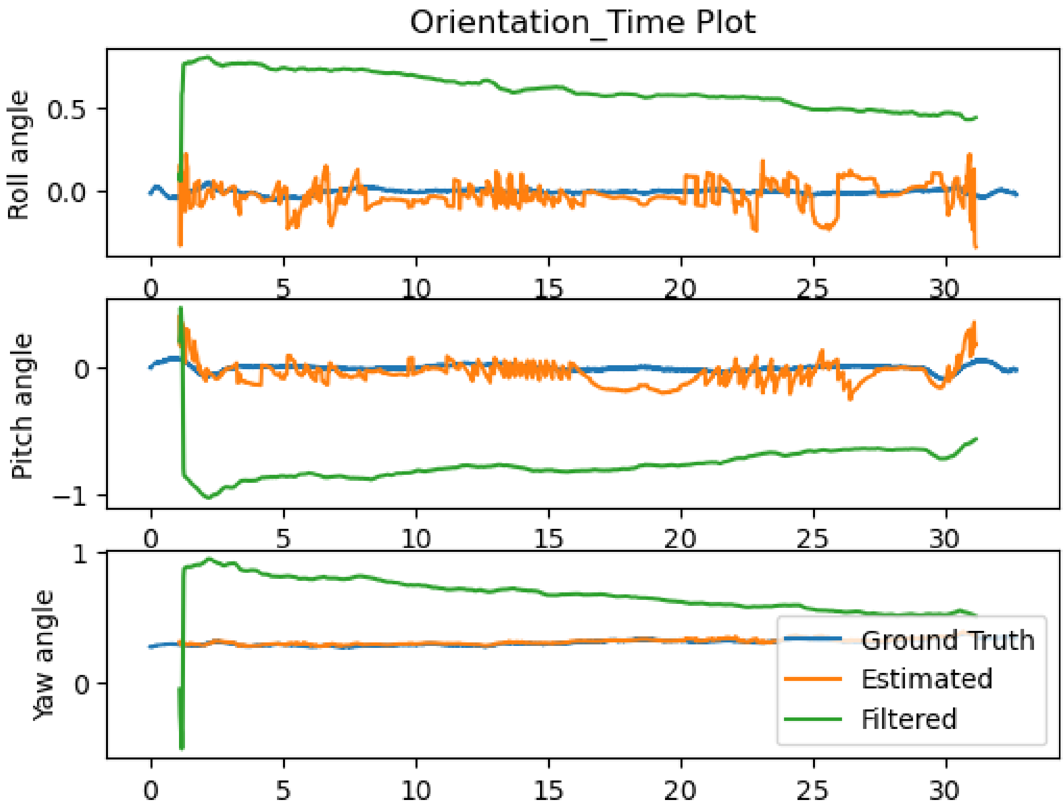
[25]:

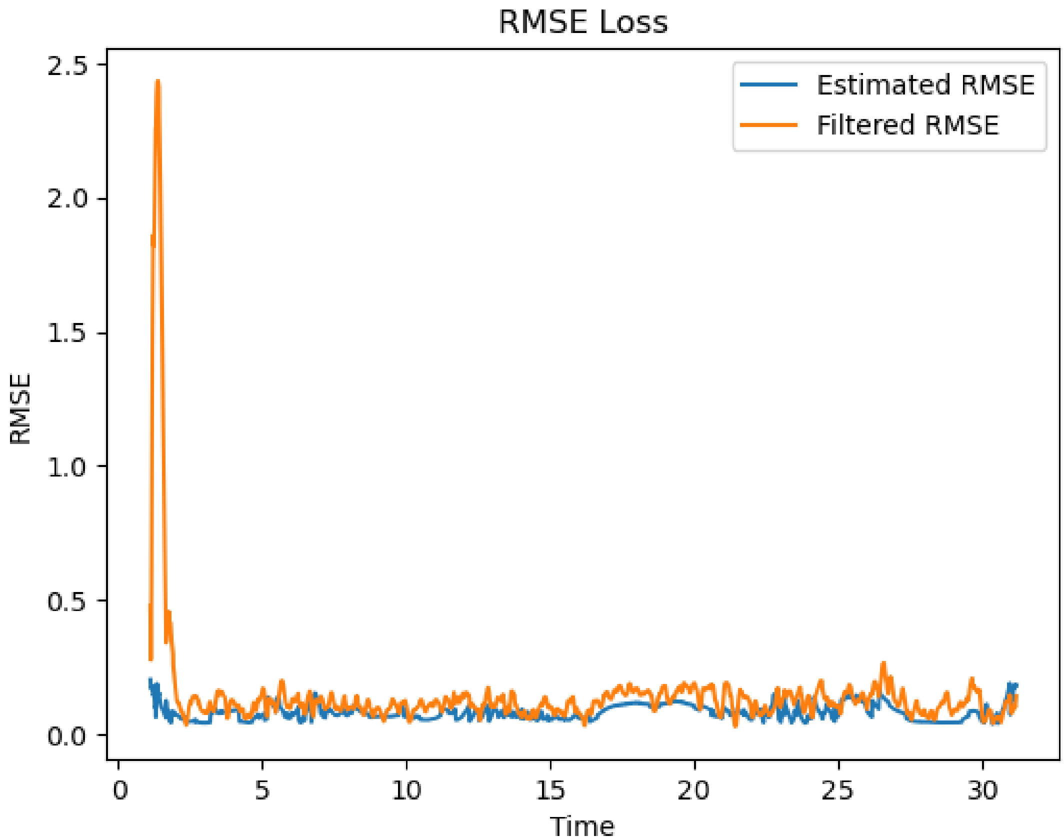
```
#Simulate the results and plot

from simulation_PF import simulate
import plots

estimated, est_t, ground_truth, ground_truth_t, filt_data, partcl_hist, exe_t =␣
 ↪simulate(r"data\data\studentdata7.mat", 1000, "weighted_avg", R, Qa=100,␣
 ↪Qg=0.1)
plots.plot(estimated, est_t, ground_truth, ground_truth_t, filt_data)
```


Position_3D Plot
Ground Truth
Estimated
Filtered
1.0
0.5
0.0
−0.5
−1.0
−1.5
2
1
0
−1
Y_axis
0
1
2
3
4
X_axis



Position_Time Plot
X_axis
Y_axis
Z_axis
Time
Ground Truth
Estimated
Filtered
0
5
10
15
20
25
30
2
1
−1


Orientation_Time Plot
Roll angle
0.5
0.0
Pitch angle
0
−1
Yaw angle
1
0
0
5
10
15
20
25
30
Ground Truth
Estimated
Filtered


RMSE Loss
Estimated RMSE
Filtered RMSE
RMSE
Time
0.0
0.5
1.0
1.5
2.0
2.5
0
5
10
15
20
25
30