Problem Report
Kalman Filtering angles

Frank Fourlas 06/04/2022

Workflow

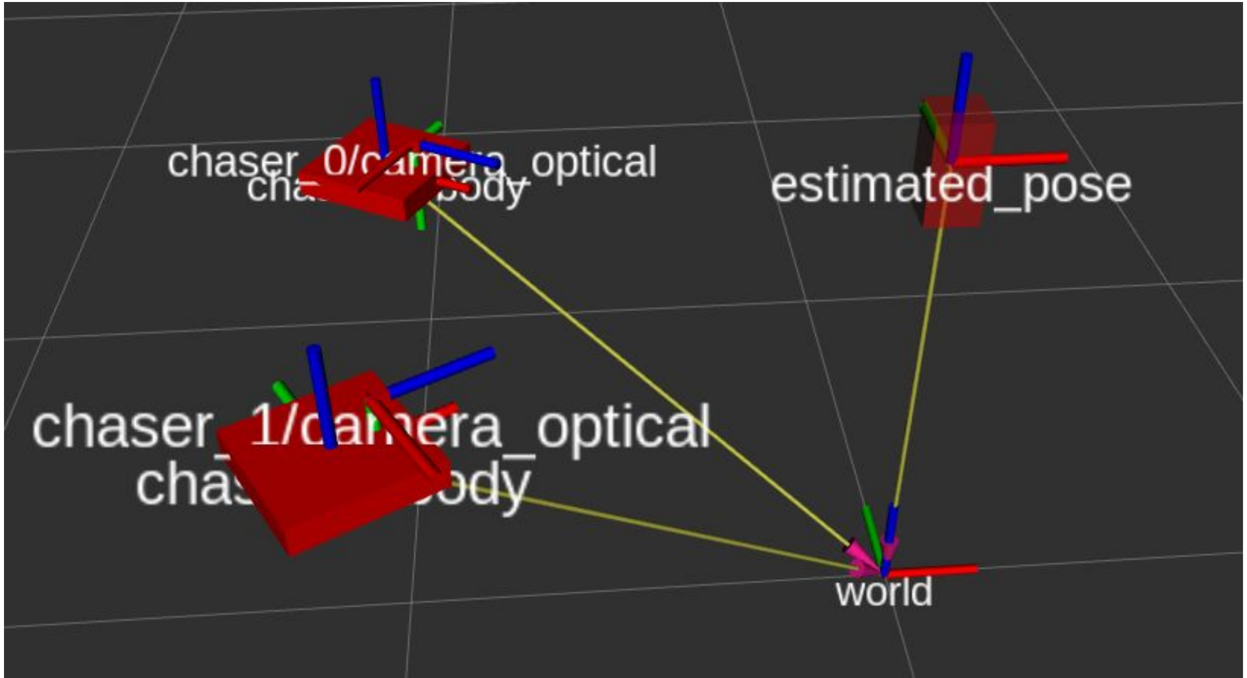
- Gazebo
  - Publish odometry for chaser
  - Publish images from chaser cameras
- Robot State Publisher
- Each Chaser
  - Entity Spawner
  - Undistorter
  - Odometry Translator (Odometry msg -> PoseStampted in /tf)
  - ArUco Board pose estimation
    - Convert Rodrigues matrix -> quaternion
- RViz
- Estimation Combiner
  - Convert quaternion -> Euler
  - Average (atan2(sum(sinθi), sum(cosθi)))
  - Euler -> quaternion

Constraints (for now)

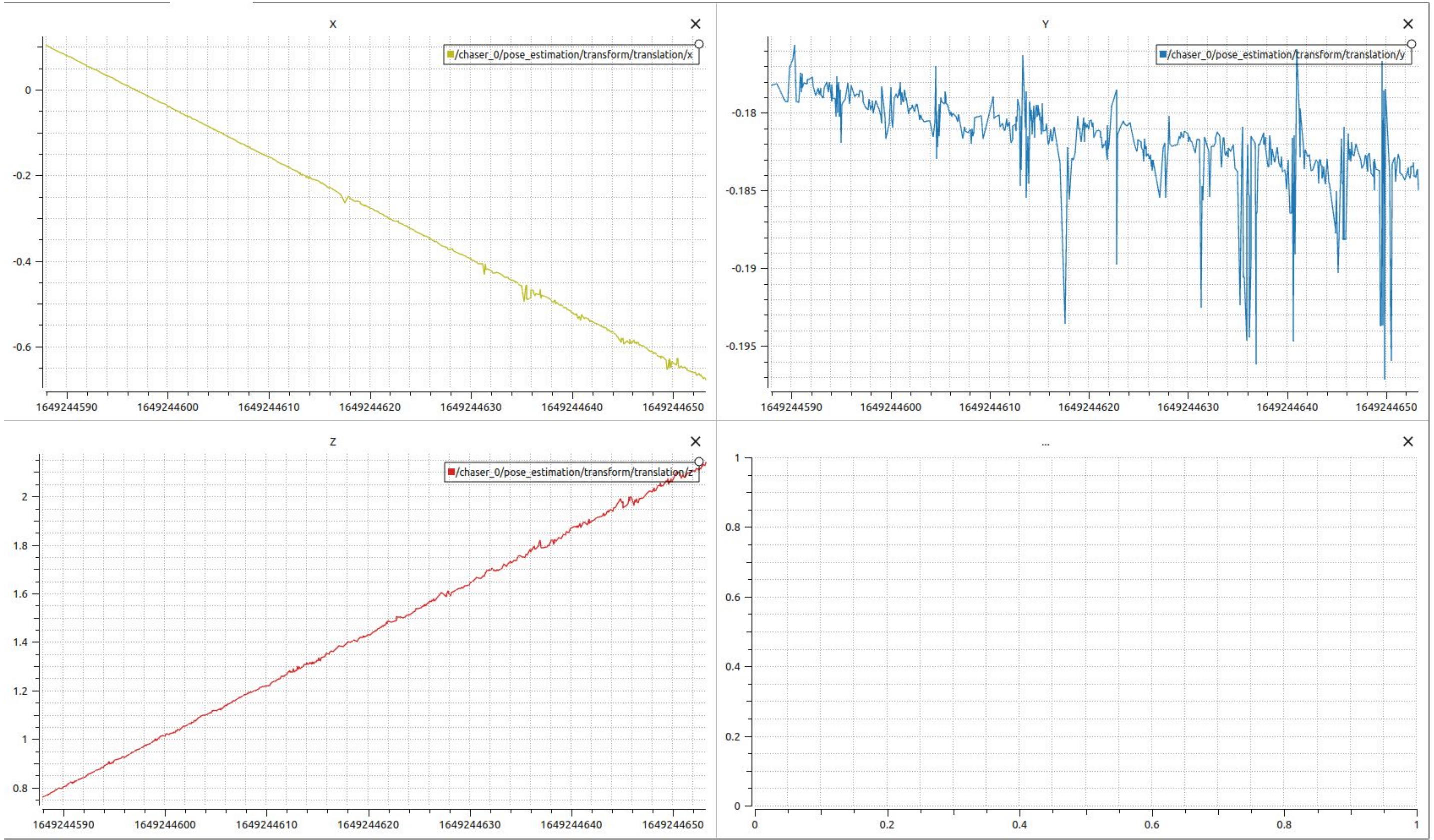
- No odometry for target
- No live rpy data (calculated in plotjuggler on rosbag)

Simulation

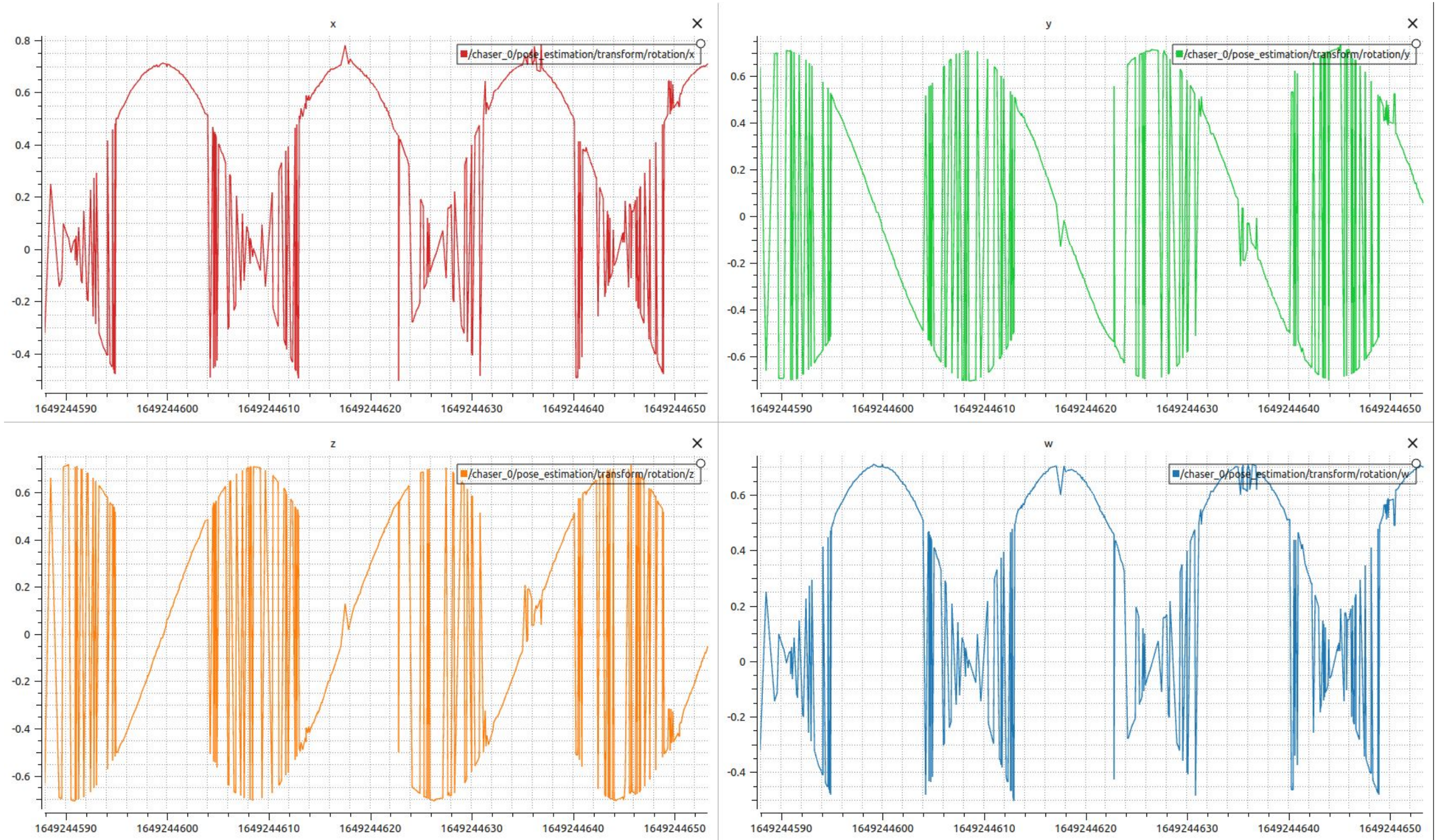
chaser_0 is looking at the target from an angle as the target moves in the x-axis (relative to world) rotating around its z-axis with constant velocities



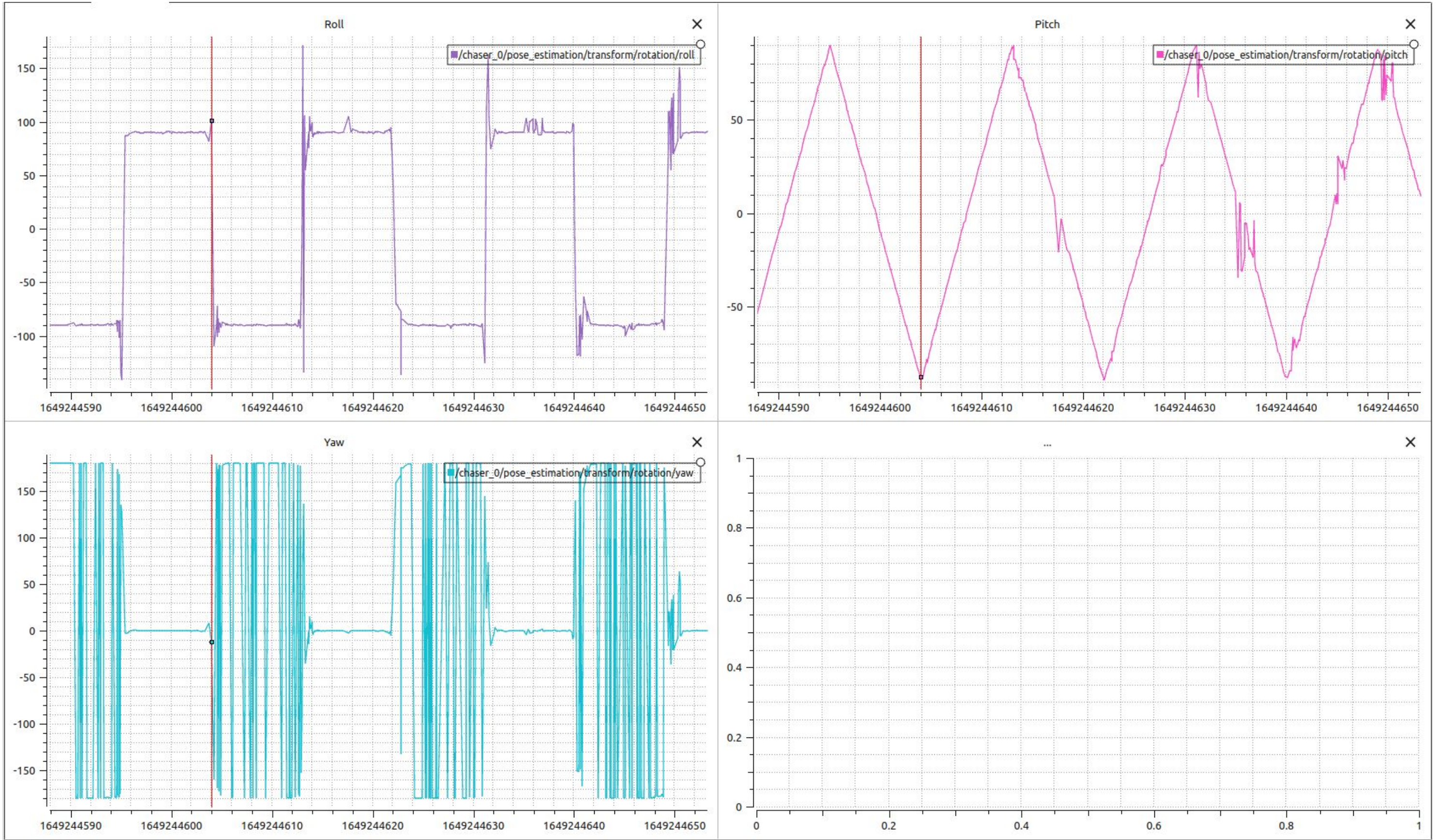
Unfiltered Estimation - Translation



Unfiltered Estimation - Rotation (Quaternion)



Unfiltered Estimation - Rotation (Euler)



Notes

- Angles present singularities/discontinuities
  - Discontinuity in one axis matches singularity in the others
  - Has to do with the way angles cycle to 0° after 359°
  - Has to do with the way angles transform from rodrigues to euler? (same pose might have several representations??)
- Discontinuities cannot happen when filtering with kalman and a constant acceleration or (worst) velocity model because they represent an impulse acceleration change.

The KF node

- Using the filterpy library
- Listen to /pose_estimation
- Convert to euler
- Predict state
- Update state with measurements
- Convert to quaternion
- Publish to /filtered_estimation

Constant Acceleration Model

- State Vector: $\mathbf{x} = (x, y, z, \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z}, \psi, \theta, \phi, \dot{\psi}, \dot{\theta}, \dot{\phi}, \ddot{\psi}, \ddot{\theta}, \ddot{\phi})^T$
- Process Model: $\mathbf{x}_k = \begin{pmatrix} A_T & \mathbf{0} \\ \mathbf{0} & A_T \end{pmatrix} \mathbf{x}_{k-1} + \mathbf{w}_{k-1}$

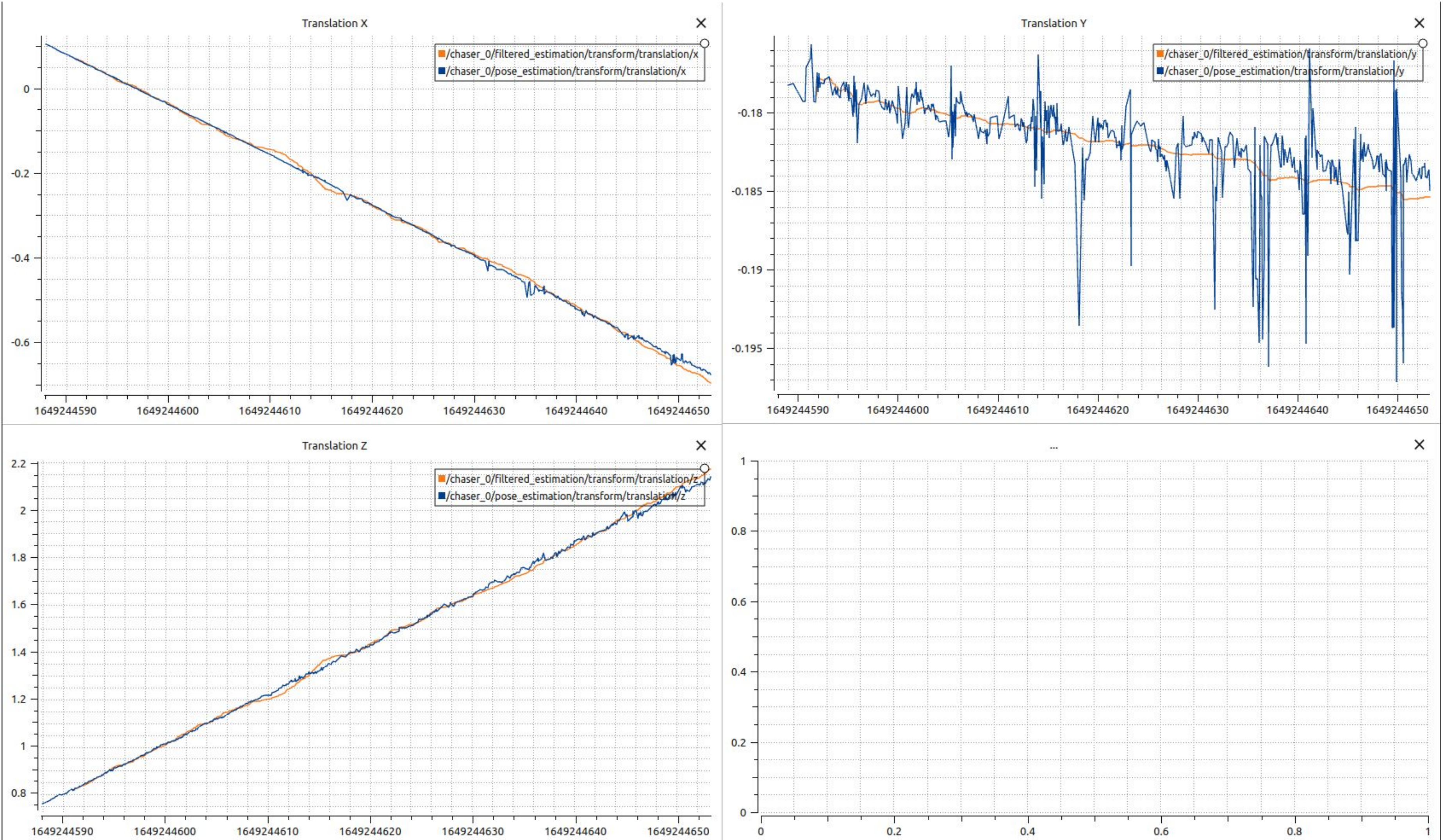
$$A_T = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{1}{2}(\Delta t)^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{1}{2}(\Delta t)^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- Measurement Model: $\begin{pmatrix} x_k \\ y_k \\ z_k \\ \psi_k \\ \theta_k \\ \phi_k \end{pmatrix} = \begin{pmatrix} I & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I & \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{x}_k + \mathbf{v}_k$

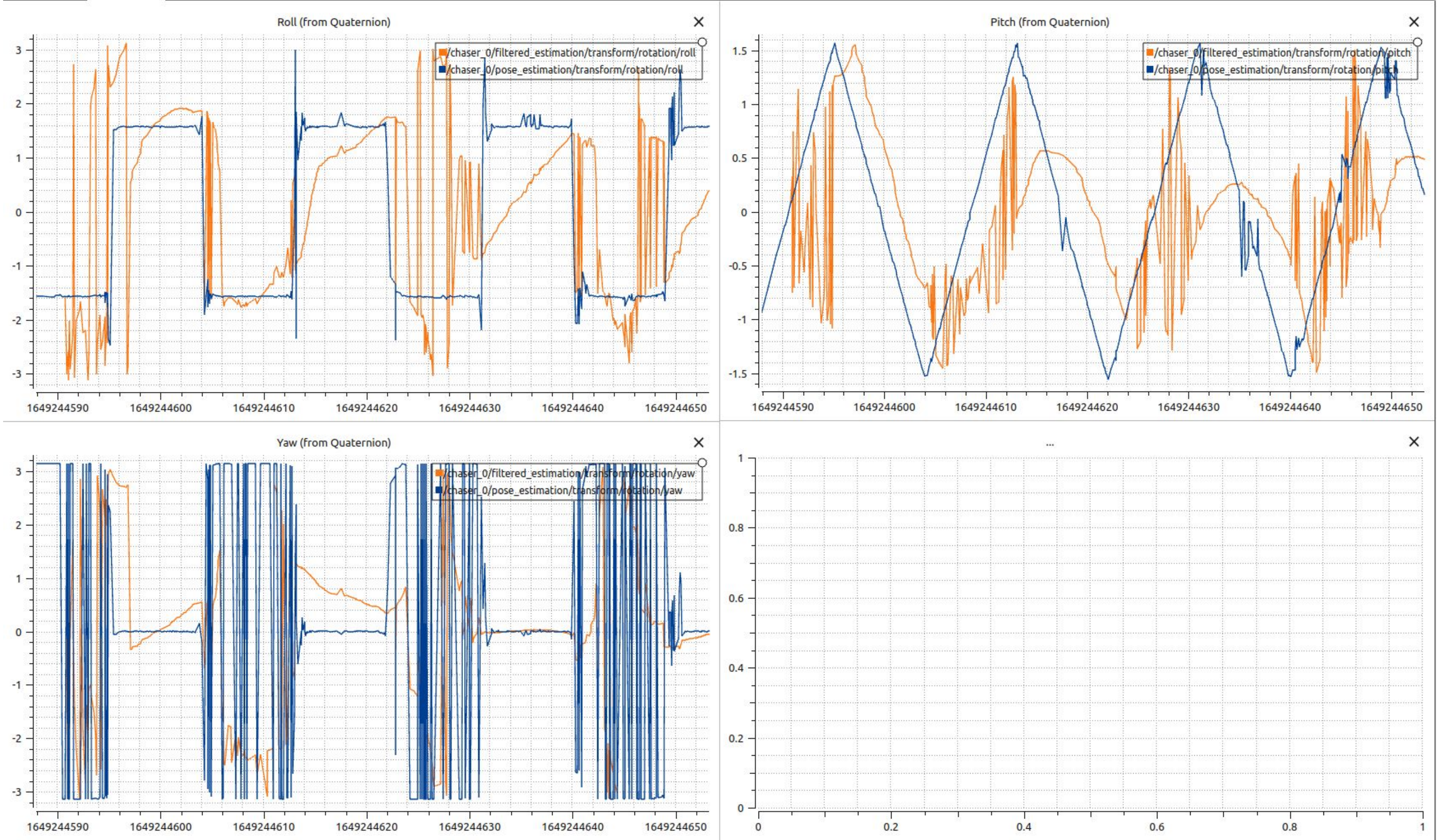
Results kalman_test_1

In the same previous scenario

- Covariance Matrix P = 1e4
- Process Noise Q = 1e-6
- Measurement Noise R = 1e-2



Translation X
/chaser_0/filtered_estimation/transform/translation/x
/chaser_0/pose_estimation/transform/translation/x
0
-0.2
-0.4
-0.6
1649244590
1649244600
1649244610
1649244620
1649244630
1649244640
1649244650
Translation Y
/chaser_0/filtered_estimation/transform/translation/y
/chaser_0/pose_estimation/transform/translation/y
-0.18
-0.185
-0.19
-0.195
1649244590
1649244600
1649244610
1649244620
1649244630
1649244640
1649244650
Translation Z
/chaser_0/filtered_estimation/transform/translation/z
/chaser_0/pose_estimation/transform/translation/z
2.2
2
1.8
1.6
1.4
1.2
1
0.8
1649244590
1649244600
1649244610
1649244620
1649244630
1649244640
1649244650
...
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1


Roll (from Quaternion)
/chaser_0/filtered_estimation/transform/rotation/roll
/chaser_0/pose_estimation/transform/rotation/roll
Pitch (from Quaternion)
/chaser_0/filtered_estimation/transform/rotation/pitch
/chaser_0/pose_estimation/transform/rotation/pitch
Yaw (from Quaternion)
/chaser_0/filtered_estimation/transform/rotation/yaw
/chaser_0/pose_estimation/transform/rotation/yaw
...

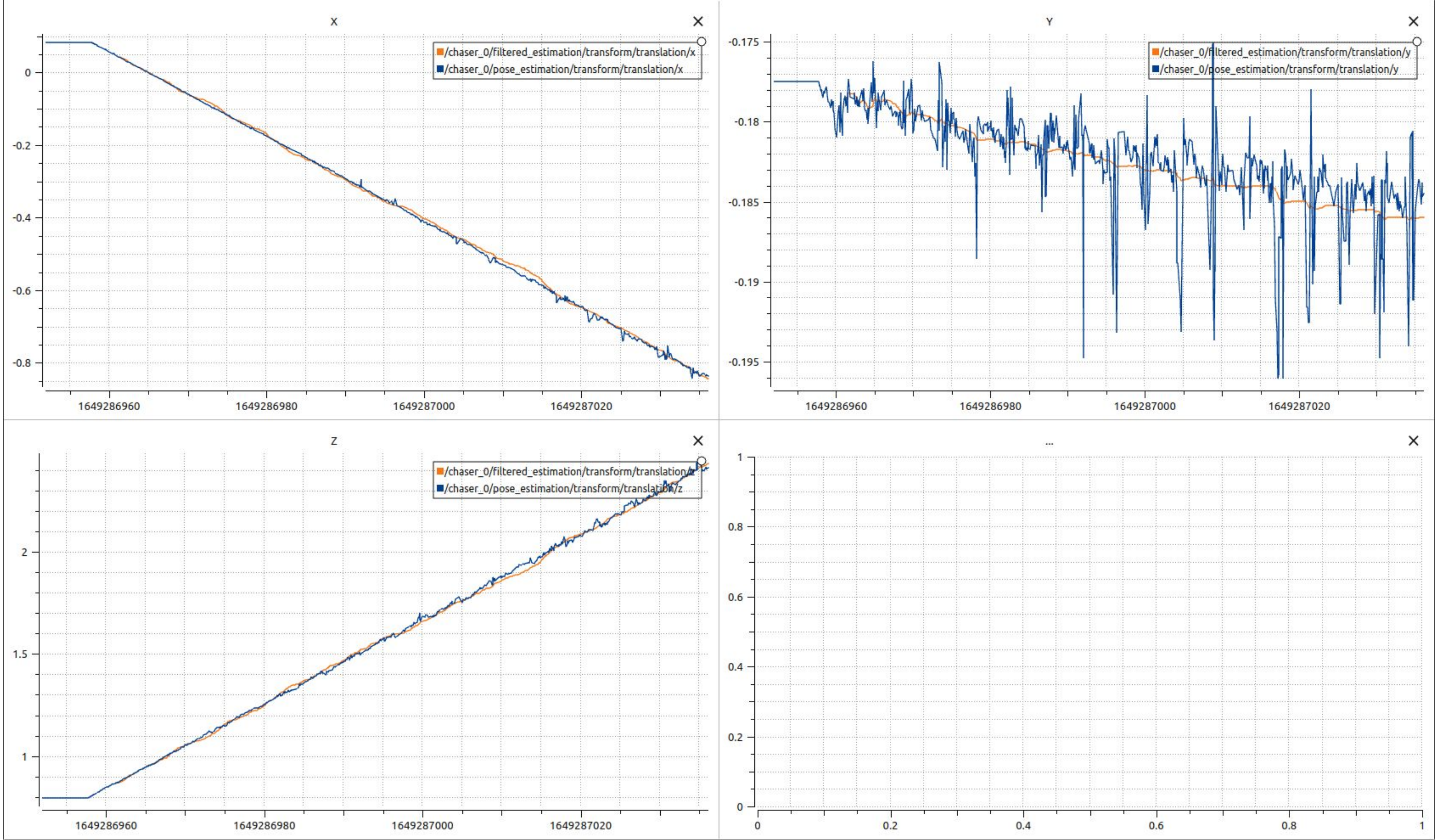
Notes

- The gains trust the model heavily and the measurements less. This makes the system VERY slow when there is a change in acceleration such as the discontinuities.
- The singularities are not something that can be processed correctly by a KF.

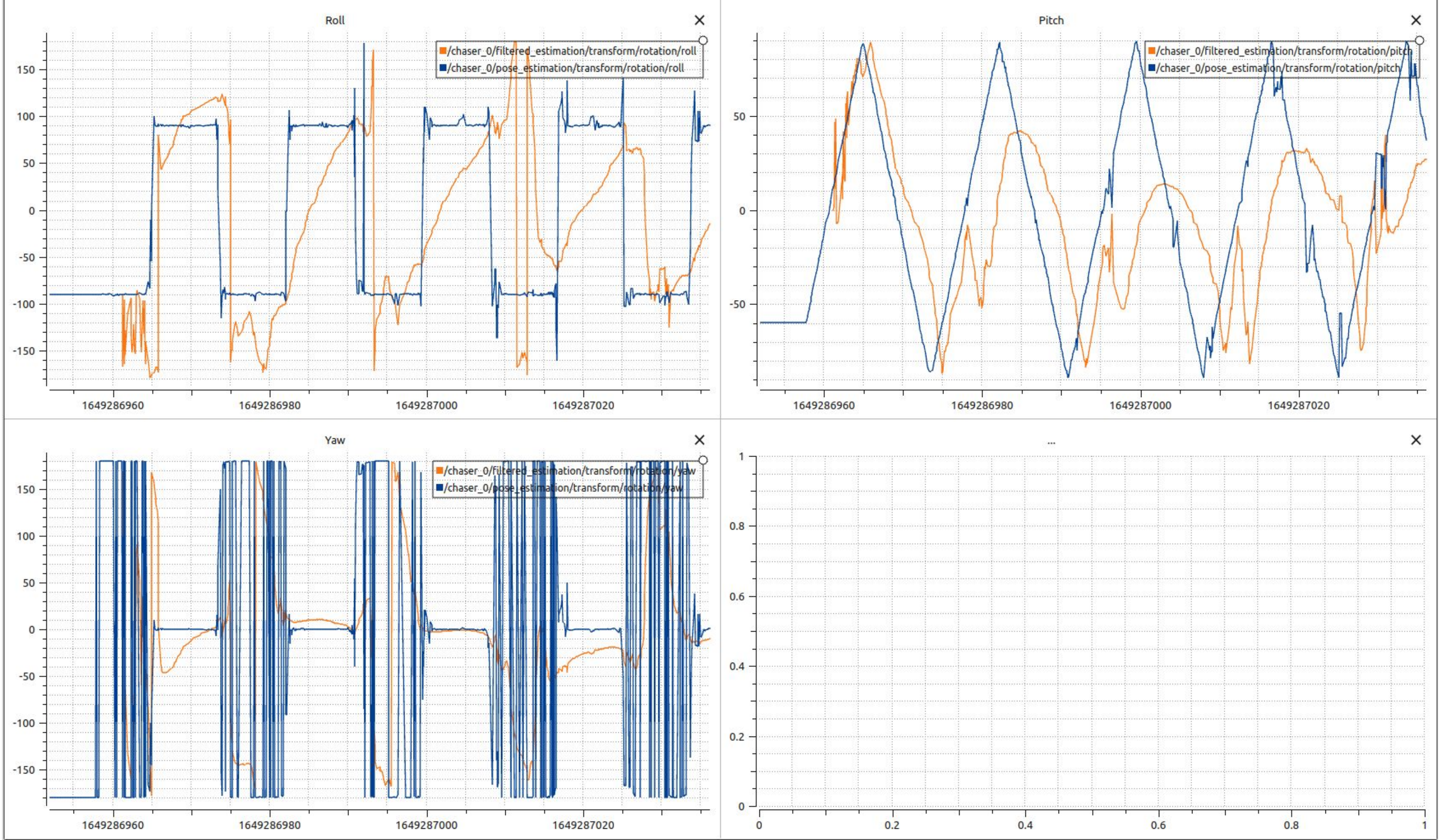
Results kalman_test_2

In the same previous scenario

- Covariance Matrix P = 1e4
- Process Noise Q = 1e-3
- Measurement Noise R = 1e-4



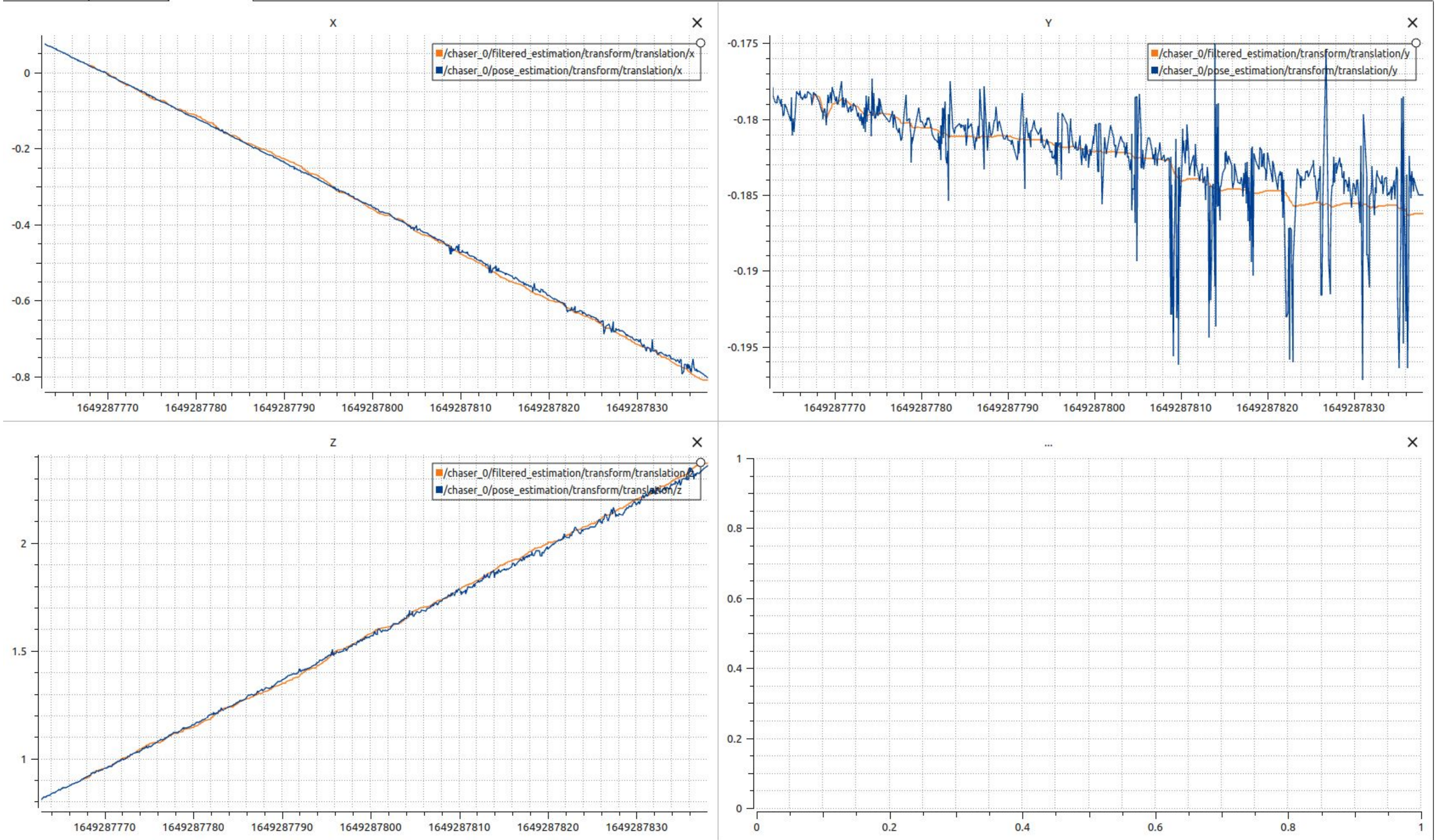
X
/chaser_0/filtered_estimation/transform/translation/x
/chaser_0/pose_estimation/transform/translation/x
Y
/chaser_0/filtered_estimation/transform/translation/y
/chaser_0/pose_estimation/transform/translation/y
Z
/chaser_0/filtered_estimation/transform/translation/z
/chaser_0/pose_estimation/transform/translation/z
1649286960
1649286980
1649287000
1649287020

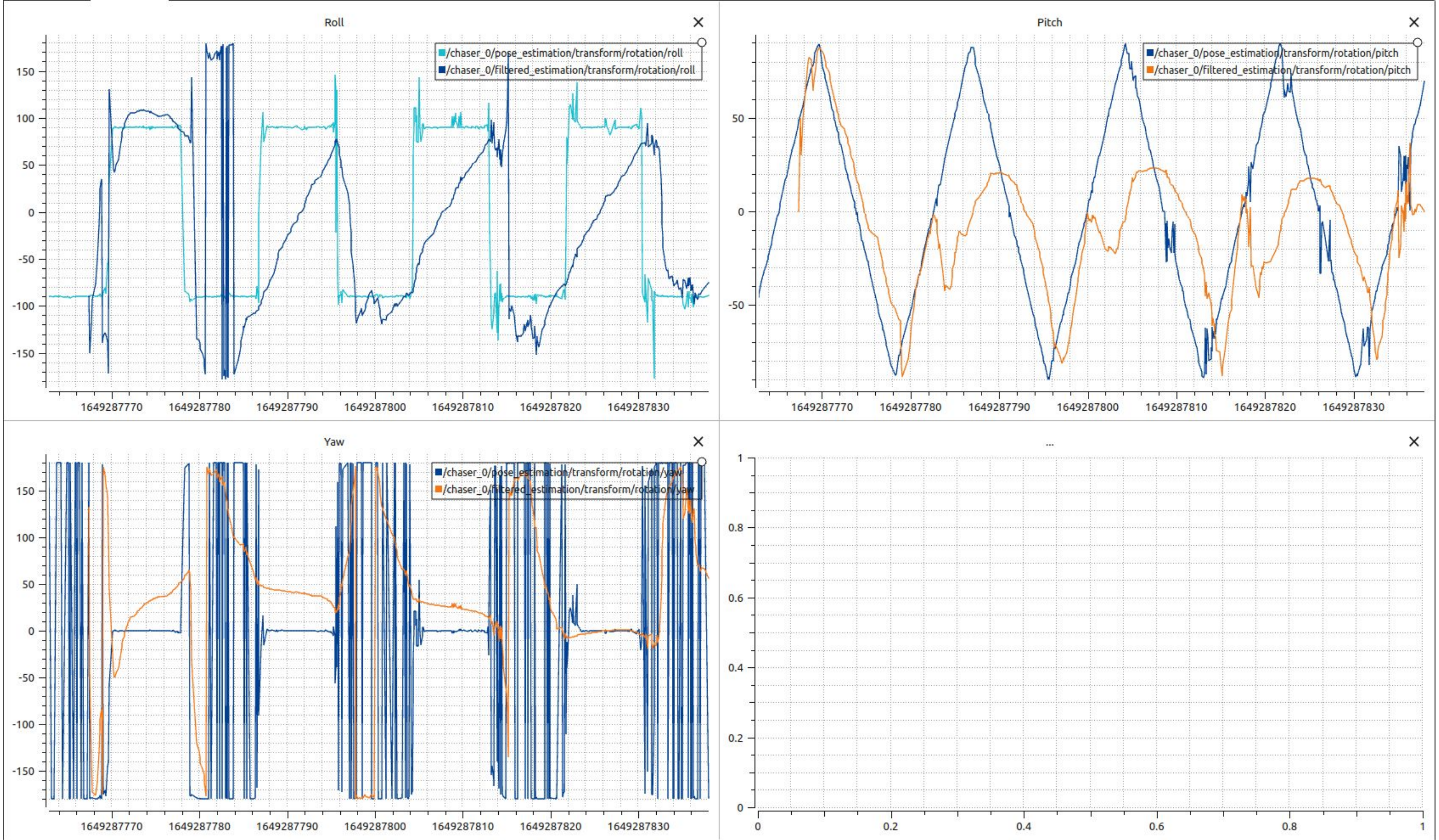

Roll
/chaser_0/filtered_estimation/transform/rotation/roll
/chaser_0/pose_estimation/transform/rotation/roll
150
100
50
0
-50
-100
-150
1649286960
1649286980
1649287000
1649287020
Pitch
/chaser_0/filtered_estimation/transform/rotation/pitch
/chaser_0/pose_estimation/transform/rotation/pitch
50
0
-50
1649286960
1649286980
1649287000
1649287020
Yaw
/chaser_0/filtered_estimation/transform/rotation/yaw
/chaser_0/pose_estimation/transform/rotation/yaw
150
100
50
0
-50
-100
-150
1649286960
1649286980
1649287000
1649287020
...
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1

Results kalman_test_3

In the same previous scenario

- Covariance Matrix P = 0
- Process Noise Q = 1e-3
- Measurement Noise R = 1e-4


X
/chaser_0/filtered_estimation/transform/translation/x
/chaser_0/pose_estimation/transform/translation/x
0
-0.2
-0.4
-0.6
-0.8
1649287770
1649287780
1649287790
1649287800
1649287810
1649287820
1649287830
Y
/chaser_0/filtered_estimation/transform/translation/y
/chaser_0/pose_estimation/transform/translation/y
-0.175
-0.18
-0.185
-0.19
-0.195
1649287770
1649287780
1649287790
1649287800
1649287810
1649287820
1649287830
Z
/chaser_0/filtered_estimation/transform/translation/z
/chaser_0/pose_estimation/transform/translation/z
2
1.5
1
1649287770
1649287780
1649287790
1649287800
1649287810
1649287820
1649287830
...
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1



Roll
/chaser_0/pose_estimation/transform/rotation/roll
/chaser_0/filtered_estimation/transform/rotation/roll
150
100
50
0
-50
-100
-150
1649287770
1649287780
1649287790
1649287800
1649287810
1649287820
1649287830
Pitch
/chaser_0/pose_estimation/transform/rotation/pitch
/chaser_0/filtered_estimation/transform/rotation/pitch
50
0
-50
1649287770
1649287780
1649287790
1649287800
1649287810
1649287820
1649287830
Yaw
/chaser_0/pose_estimation/transform/rotation/yaw
/chaser_0/filtered_estimation/transform/rotation/yaw
150
100
50
0
-50
-100
-150
1649287770
1649287780
1649287790
1649287800
1649287810
1649287820
1649287830
...
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1

Notes

Improved performance (still REALLY bad) because:

- Zero covariance (No correlation between the process and measurement noise)
  - Measurement noise is the camera noise which does not affect the processing directly
  - Things that could affect processing like light flares are not gaussian and distortion is removed by undistortion
  - Processing noise is faults in the aruco marker detection, corner detection, PnP solving, etc.
- More trust to the measurements
- Less trust to the model
  - Might not be that good for the translation
  - Non-Identity Q, R matrices??

Index

- **Chaser**: the controlled spacecraft mounted with a camera trying to dock the **target**.
- **Target**: the uncontrolled/uncooperative spacecraft with imprinted markers.
- **pose_estimation**: the raw output of the ArUco node which processes images and outputs an estimation for the **chasers** pose.
- **filtered_estimation**: the output of the kalman filter which filters the **pose_estimation** to remove process and measurement noise.
- Kalman gains:
  - **P**: Covariance matrix
  - **R**: Measurement noise
  - **Q**: Process noise