Problem Report
Kalman Filtering angles

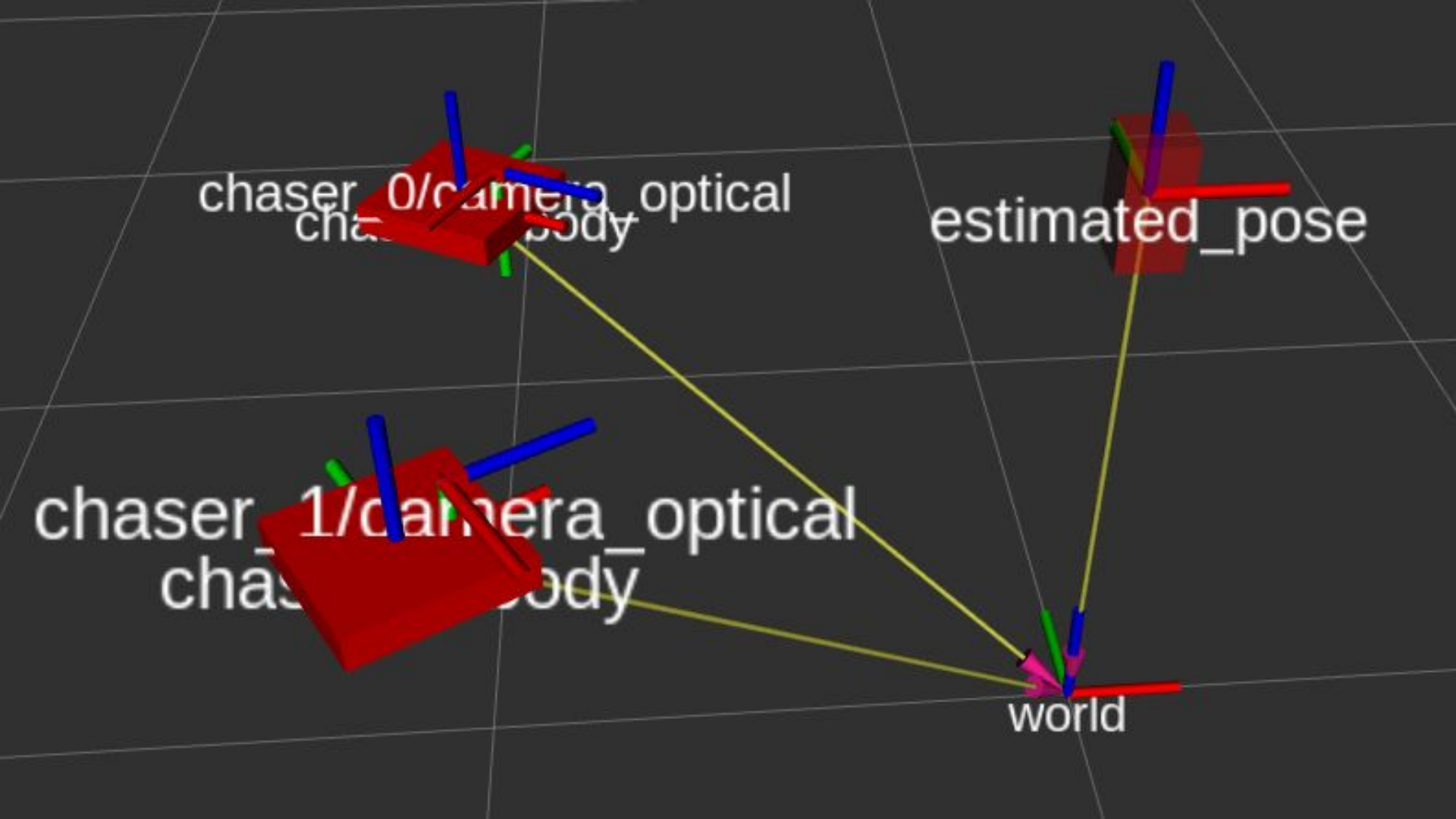
Frank Fourlas 09/04/2022

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

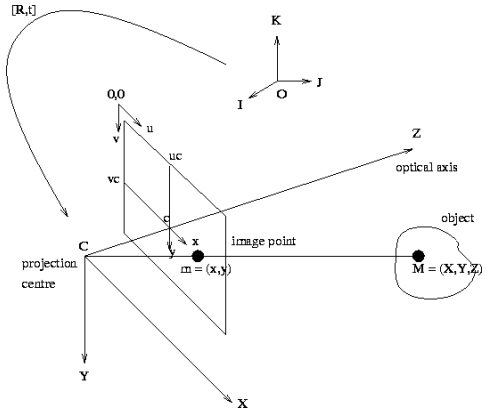
Constraints (for now)

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


chaser_0/camera_optical
estimated_pose
chaser_1/camera_optical
world

Reference Systems

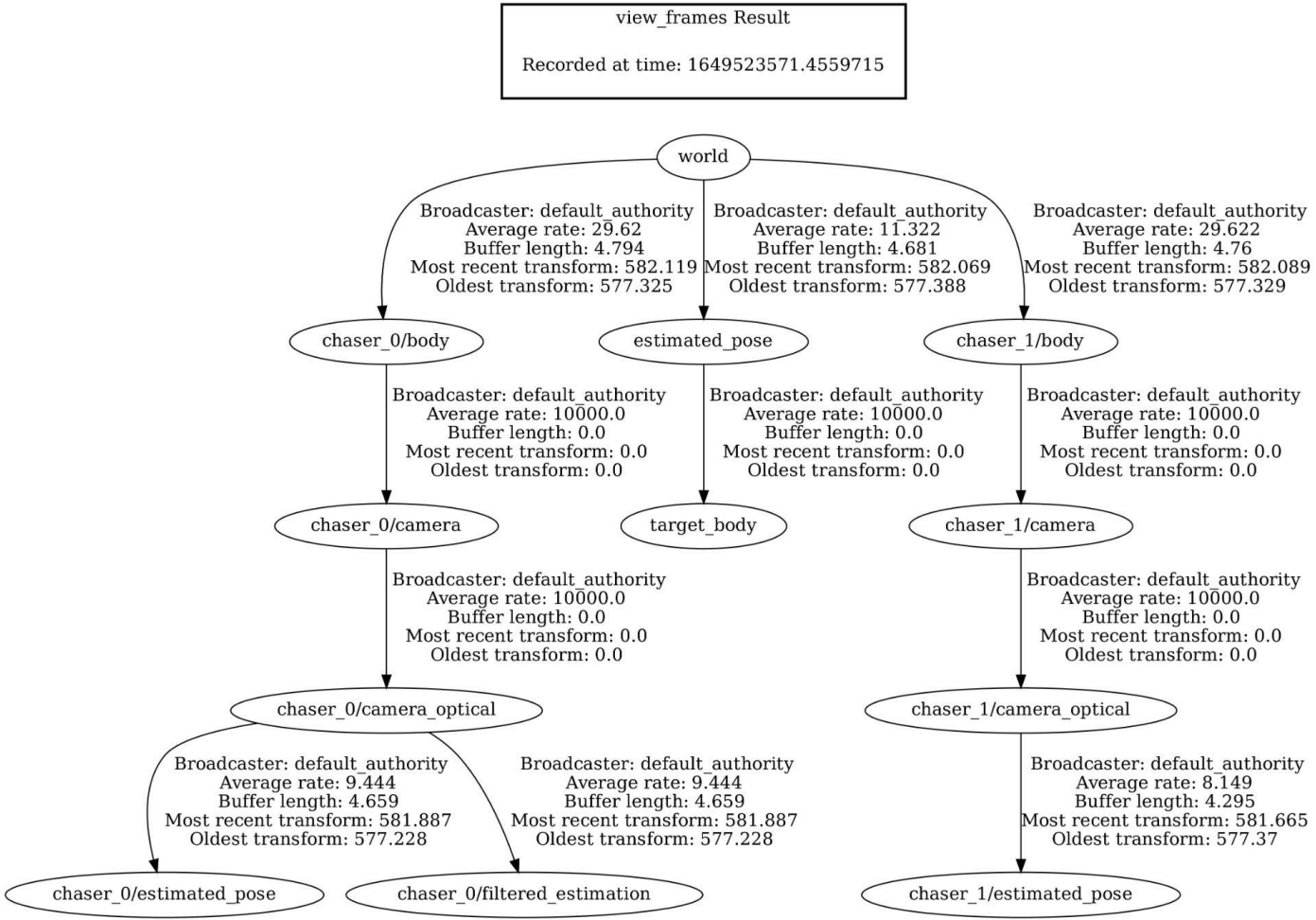
- **world**
- **target_body**: is fixed in the center of mass of the target initially in the same orientation as the **world**
- **chaser_n/body**: is fixed in the center of mass of each chaser initially in the same orientation as the **world**
- **chaser_n/camera_optical**: is fixed in each chasers camera sensor center in [this orientation]



Simulation setup (some changes since last time)

- All plots are in **meters** and **radians** and in relation to the **world** reference system unless stated otherwise
- This time, the estimator node publishes pose data (chaser_n/estimated_pose) to the filter node, as well as in the **/tf** topic, relative to the **camera_optical** frame.
- A utility node uses lookup_transform to find the estimated and filtered pose in relation to the **world** frame from the tf tree and publish it for use in the graphs

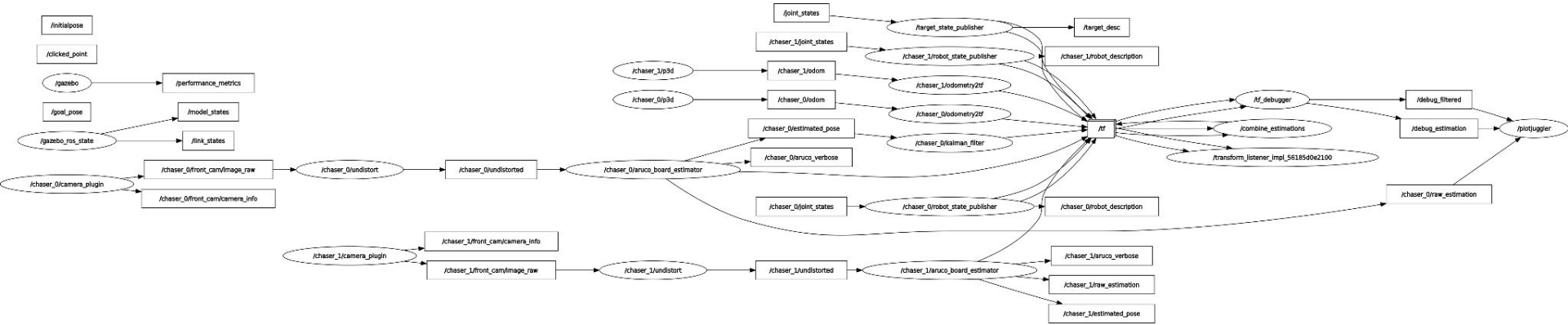
tf tree


view_frames Result
Recorded at time: 1649523571.4559715
world
Broadcaster: default_authority
Average rate: 29.62
Buffer length: 4.794
Most recent transform: 582.119
Oldest transform: 577.325
Broadcaster: default_authority
Average rate: 11.322
Buffer length: 4.681
Most recent transform: 582.069
Oldest transform: 577.388
Broadcaster: default_authority
Average rate: 29.622
Buffer length: 4.76
Most recent transform: 582.089
Oldest transform: 577.329
chaser_0/body
estimated_pose
chaser_1/body
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
chaser_0/camera
target_body
chaser_1/camera
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: default_authority
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
chaser_0/camera_optical
chaser_1/camera_optical
Broadcaster: default_authority
Average rate: 9.444
Buffer length: 4.659
Most recent transform: 581.887
Oldest transform: 577.228
Broadcaster: default_authority
Average rate: 9.444
Buffer length: 4.659
Most recent transform: 581.887
Oldest transform: 577.228
Broadcaster: default_authority
Average rate: 8.149
Buffer length: 4.295
Most recent transform: 581.665
Oldest transform: 577.37
chaser_0/estimated_pose
chaser_0/filtered_estimation
chaser_1/estimated_pose

The KF node

- Using the filterpy library
- Listen to /pose_estimation topic and get the estimated pose relative to the **camera_optical** frame
- Convert to euler
- Predict state
- Update state with measurements
- Convert to quaternion
- Publish to /tf as **chaser_n/filtered_estimation**

RQT Graph


/initialpose
/clicked_point
/gazebo
/performance_metrics
/goal_pose
/model_states
/gazebo_ros_state
/link_states
/chaser_0/camera_plugin
/chaser_0/front_cam/image_raw
/chaser_0/front_cam/camera_info
/chaser_0/undistort
/chaser_0/undistorted
/chaser_0/aruco_board_estimator
/chaser_0/estimated_pose
/chaser_0/aruco_verbose
/joint_states
/target_state_publisher
/target_desc
/chaser_1/joint_states
/chaser_1/robot_state_publisher
/chaser_1/robot_description
/chaser_1/p3d
/chaser_1/odom
/chaser_1/odometry2tf
/chaser_0/p3d
/chaser_0/odom
/chaser_0/odometry2tf
/chaser_0/kalman_filter
/tf
/tf_debugger
/debug_filtered
/combine_estimations
/debug_estimation
/plotjuggler
/transform_listener_impl_56185d0e2100
/chaser_0/joint_states
/chaser_0/robot_state_publisher
/chaser_0/robot_description
/chaser_0/raw_estimation
/chaser_1/camera_plugin
/chaser_1/front_cam/camera_info
/chaser_1/front_cam/image_raw
/chaser_1/undistort
/chaser_1/undistorted
/chaser_1/aruco_board_estimator
/chaser_1/aruco_verbose
/chaser_1/raw_estimation
/chaser_1/estimated_pose

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

Simulation kalman_test_4

chaser_0 is looking at the target from an angle. The target is moving with constant velocities in the +X **world** axis while rotating around **its own** Z axis. Initial pose of target is xyz = [-0.3, 0, 1], rpy = [0, 0, 0] in the world frame

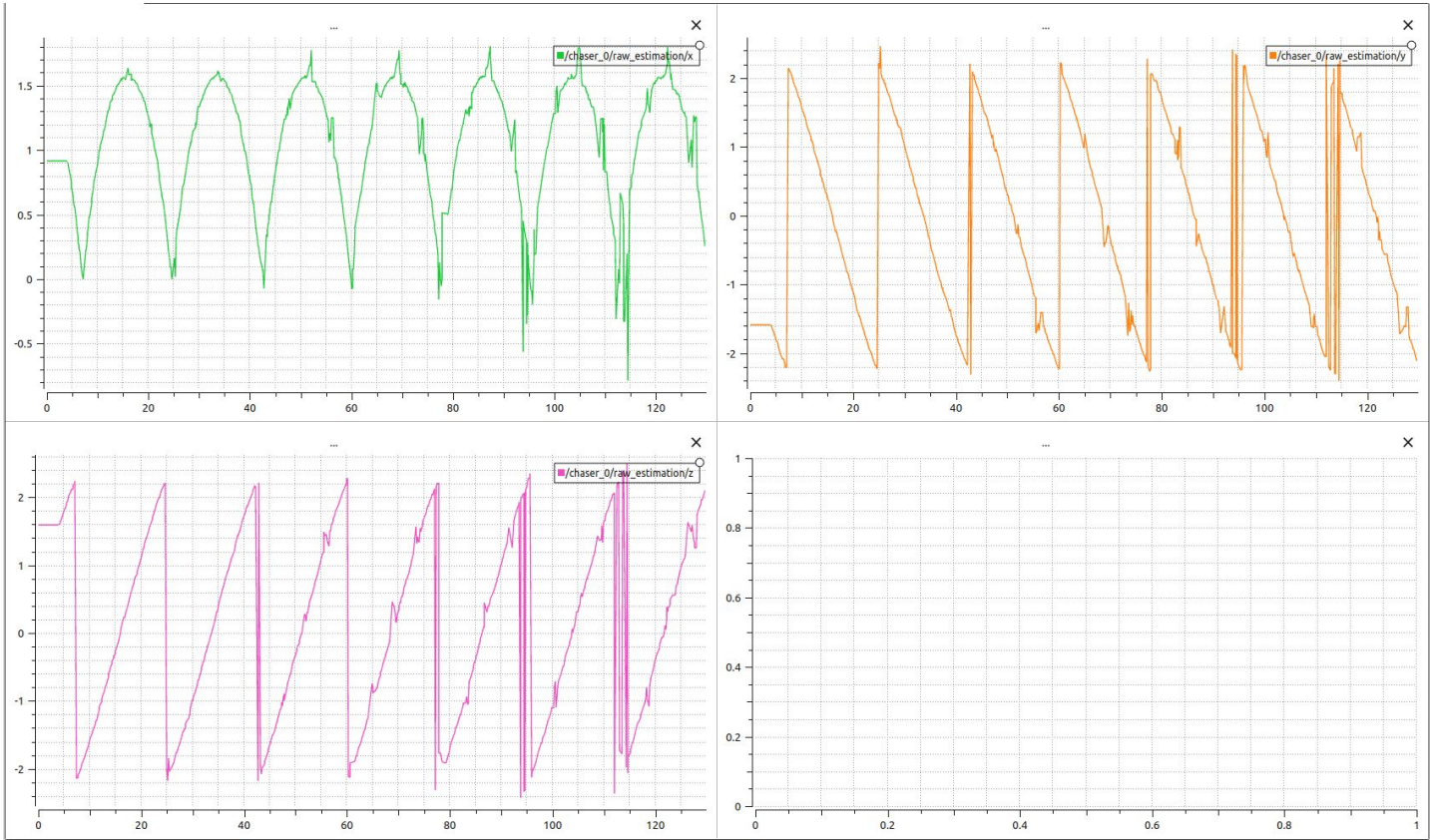
- Covariance Matrix P = 1e4
- Process Noise Q = 1e-2
- Measurement Noise R = 1e-4

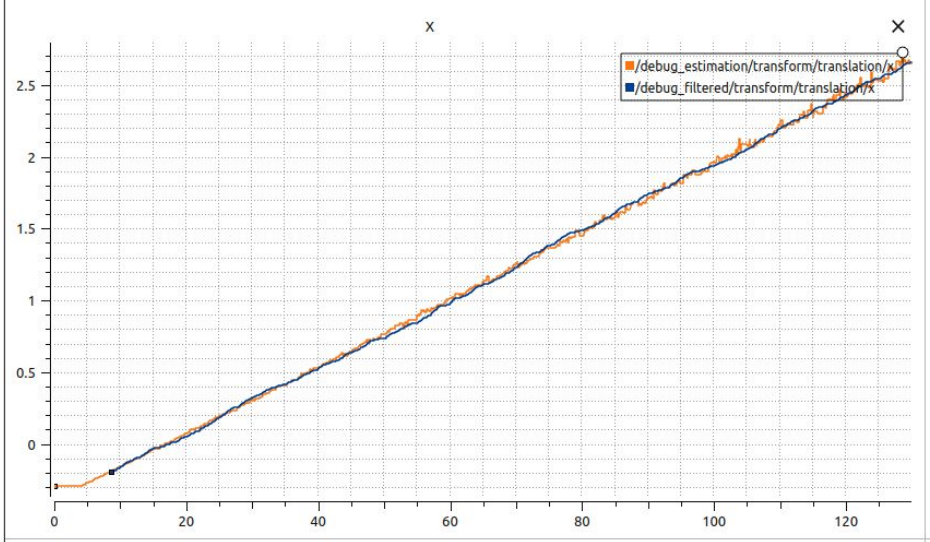
Raw estimation output

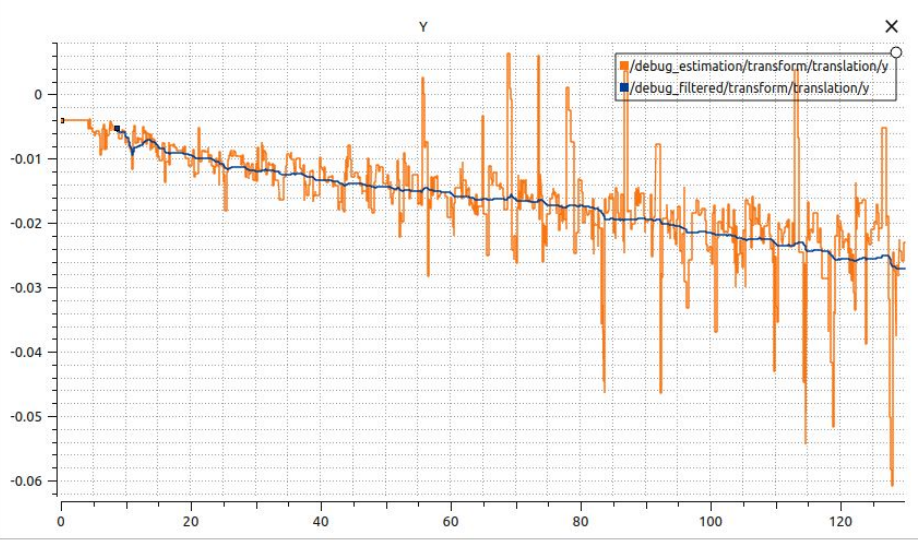
radians - seconds

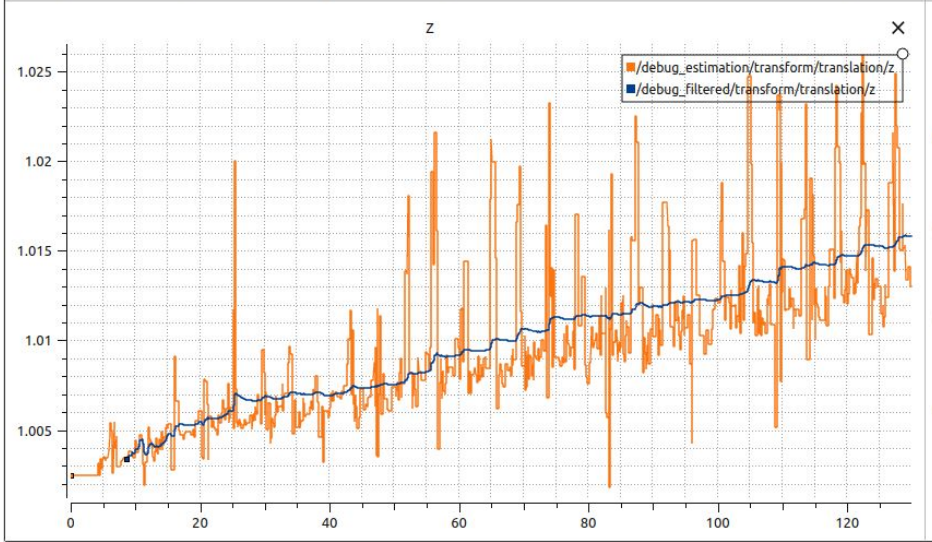
In respect to the **camera_optical** frame

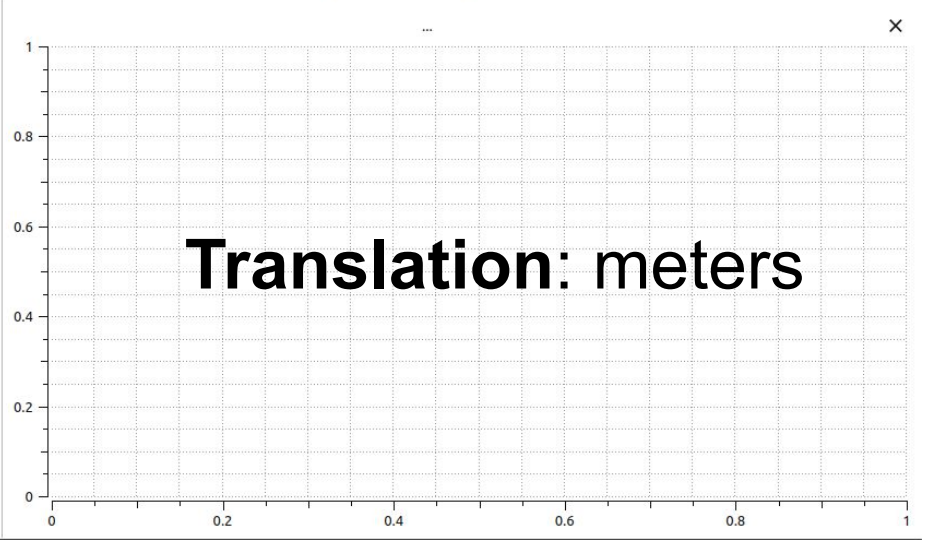
solvePnP returns a rotation vector with these 3 values that is then converted into Rodrigues matrix and then quaternion



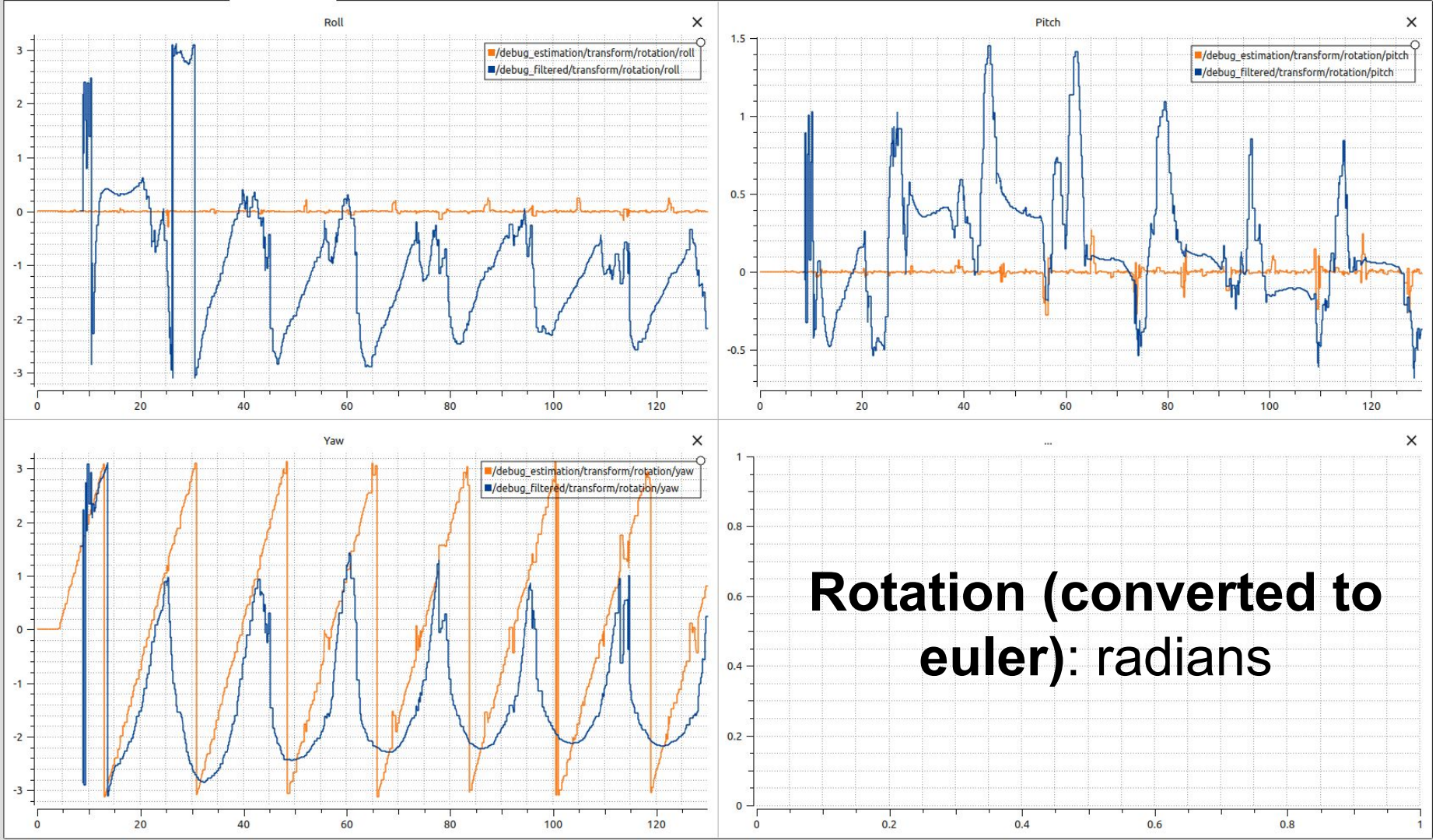


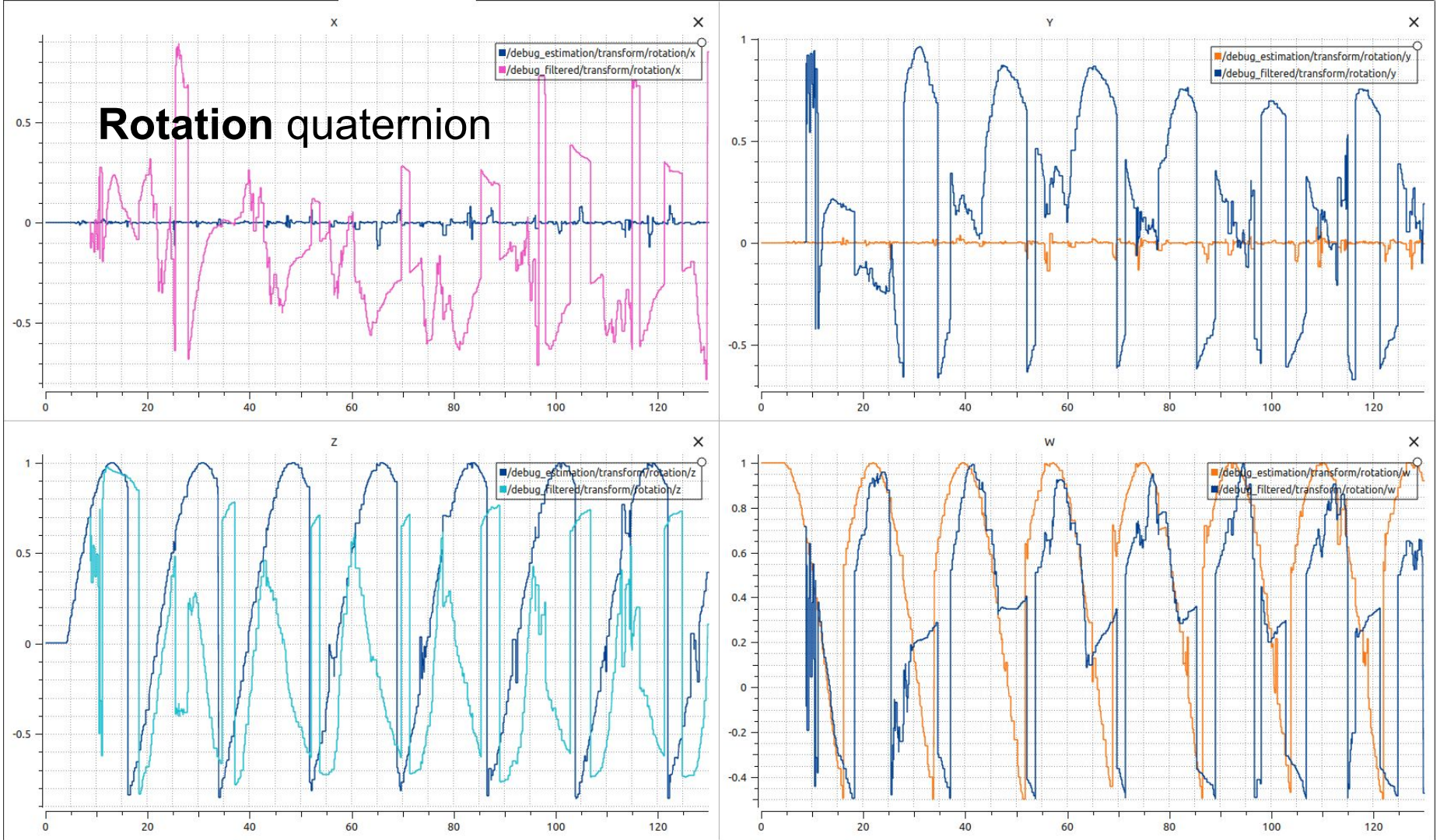






Translation: meters


Roll
/debug_estimation/transform/rotation/roll
/debug_filtered/transform/rotation/roll
Pitch
/debug_estimation/transform/rotation/pitch
/debug_filtered/transform/rotation/pitch
Yaw
/debug_estimation/transform/rotation/yaw
/debug_filtered/transform/rotation/yaw
Rotation (converted to euler): radians


x
/debug_estimation/transform/rotation/x
/debug_filtered/transform/rotation/x
Rotation quaternion
Y
/debug_estimation/transform/rotation/y
/debug_filtered/transform/rotation/y
z
/debug_estimation/transform/rotation/z
/debug_filtered/transform/rotation/z
w
/debug_estimation/transform/rotation/w
/debug_filtered/transform/rotation/w

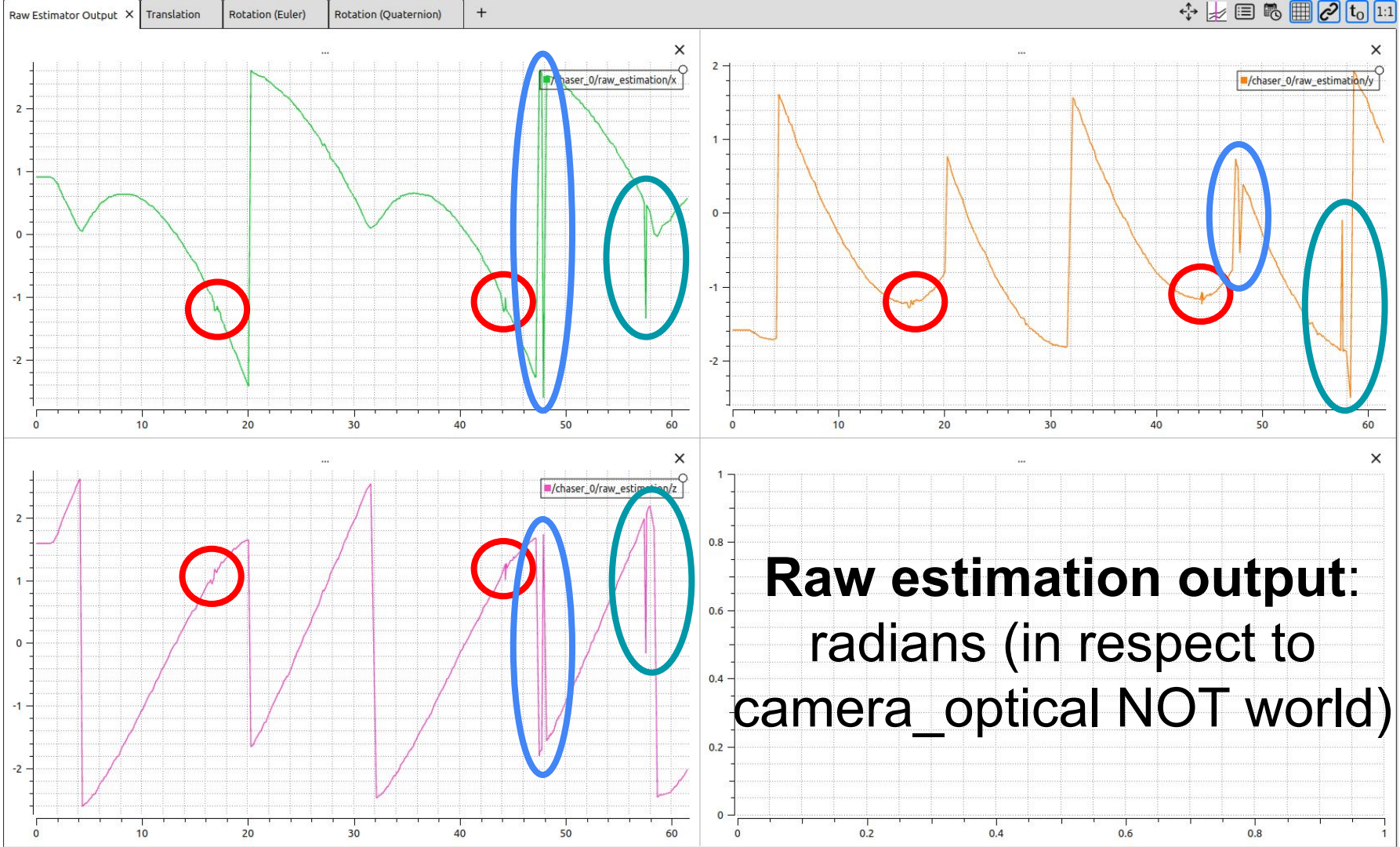
Results

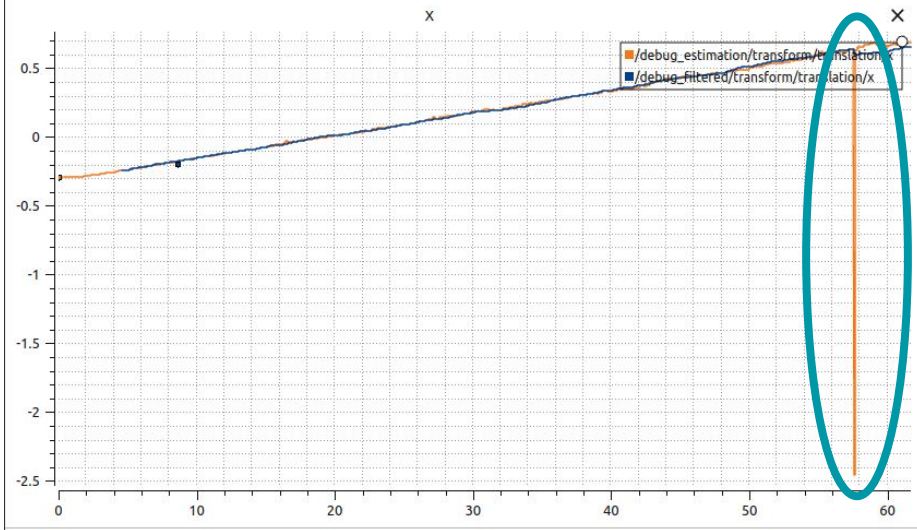
- When getting the data from the **tf tree** with respect to the world frame most of the chattering disappears.
- Singularities when making a full rotation persist
- lookup_transform most likely has a way to fix the chattering and most singularities and discontinuities, and give a final transformation from the **estimated_pose** frame to the **world** frame
- In this scenario the target is only moving and rotating in 1 axis. Maybe when there are rotations in all axis the problem persists
- NOTE: The kalman node filters data in respect to the **camera_optical** frame and NOT **world**. This is why the estimation suffers from the same problem as last time

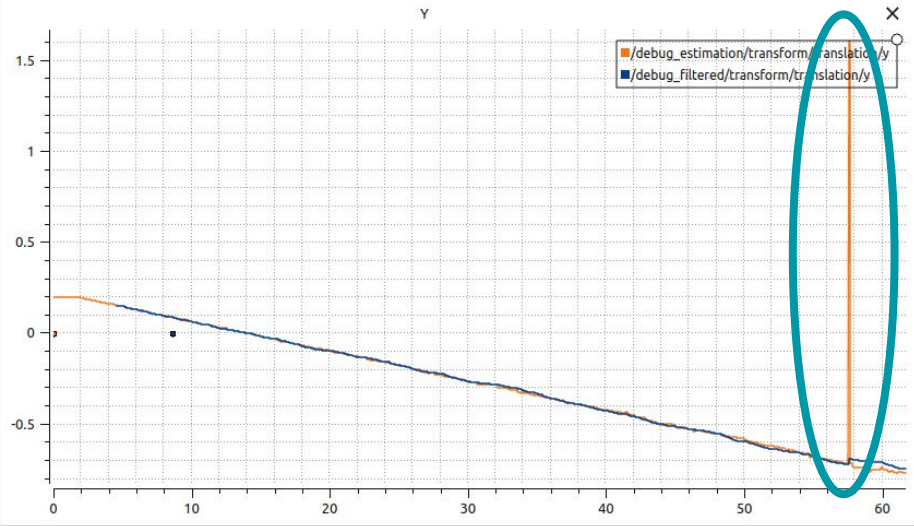
Simulation kalman_test_5

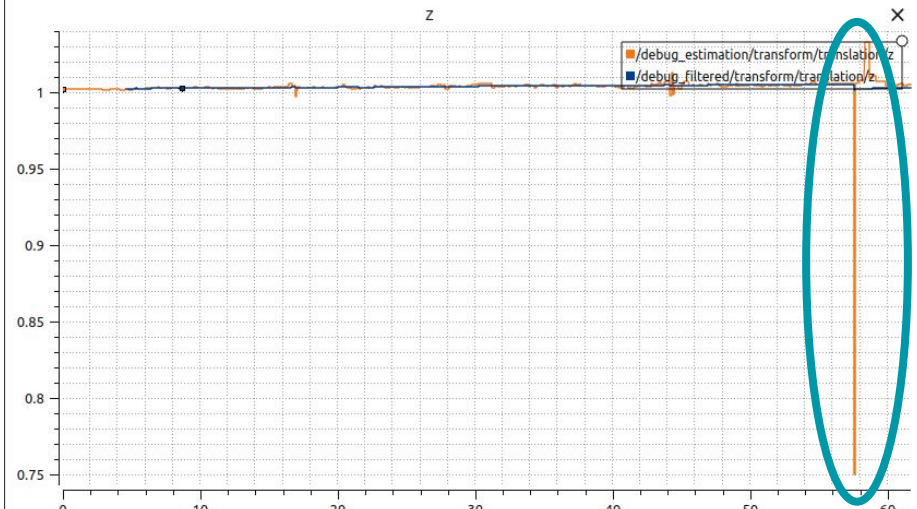
chaser_0 is looking at the target from an angle. The target is moving with constant velocities in the +X and -Y **world** axis while rotating around **all of its** axis. Initial pose of target is xyz = [-0.3, 0.2, 1], rpy = [0, 0, 0] in the world frame

- Covariance Matrix P = 1e4
- Process Noise Q = 1e-2
- Measurement Noise R = 1e-4

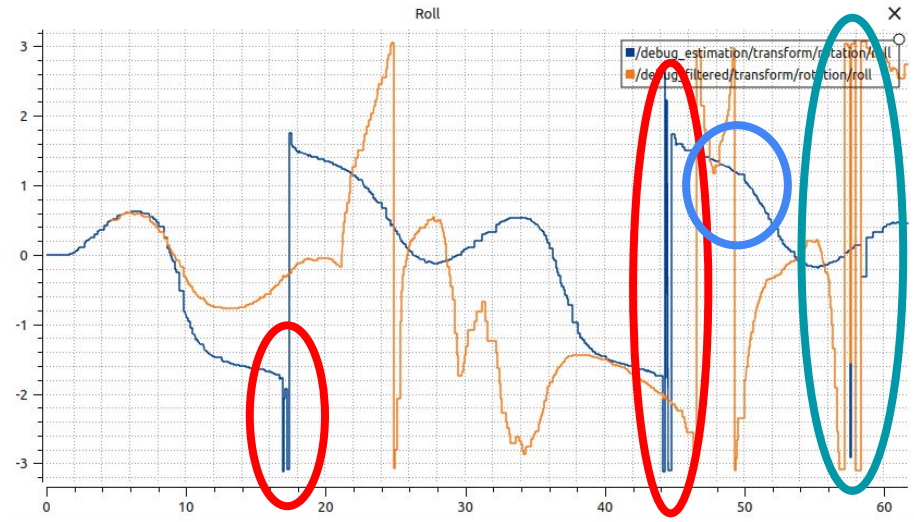

Raw Estimator Output
Translation
Rotation (Euler)
Rotation (Quaternion)
/chaser_0/raw_estimation/x
/chaser_0/raw_estimation/y
/chaser_0/raw_estimation/z
Raw estimation output:
radians (in respect to
camera_optical NOT world)

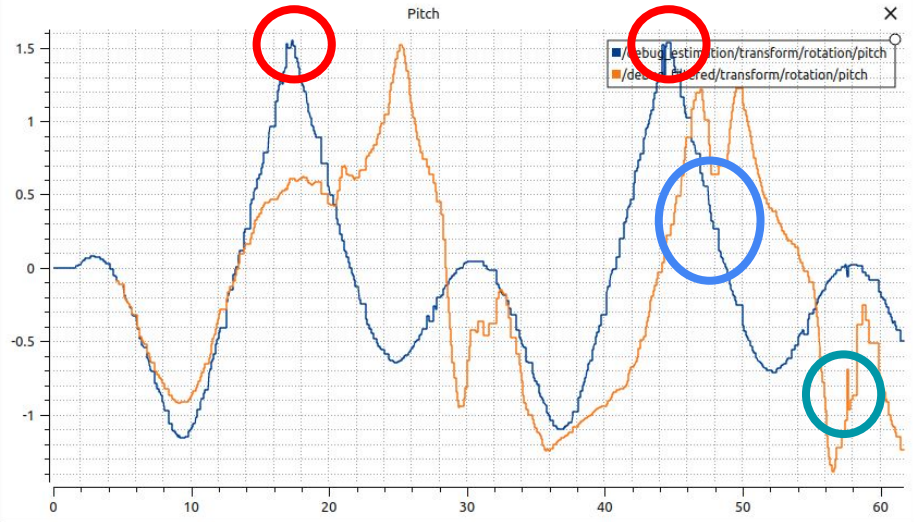


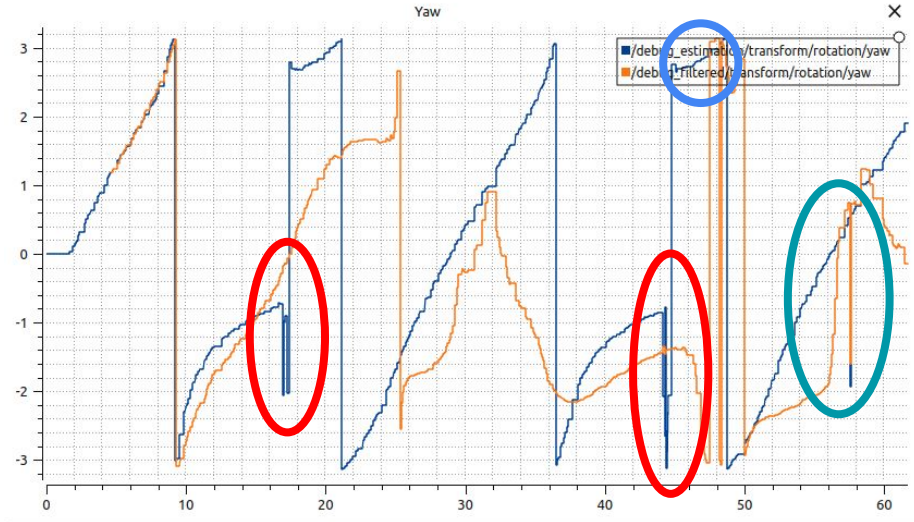




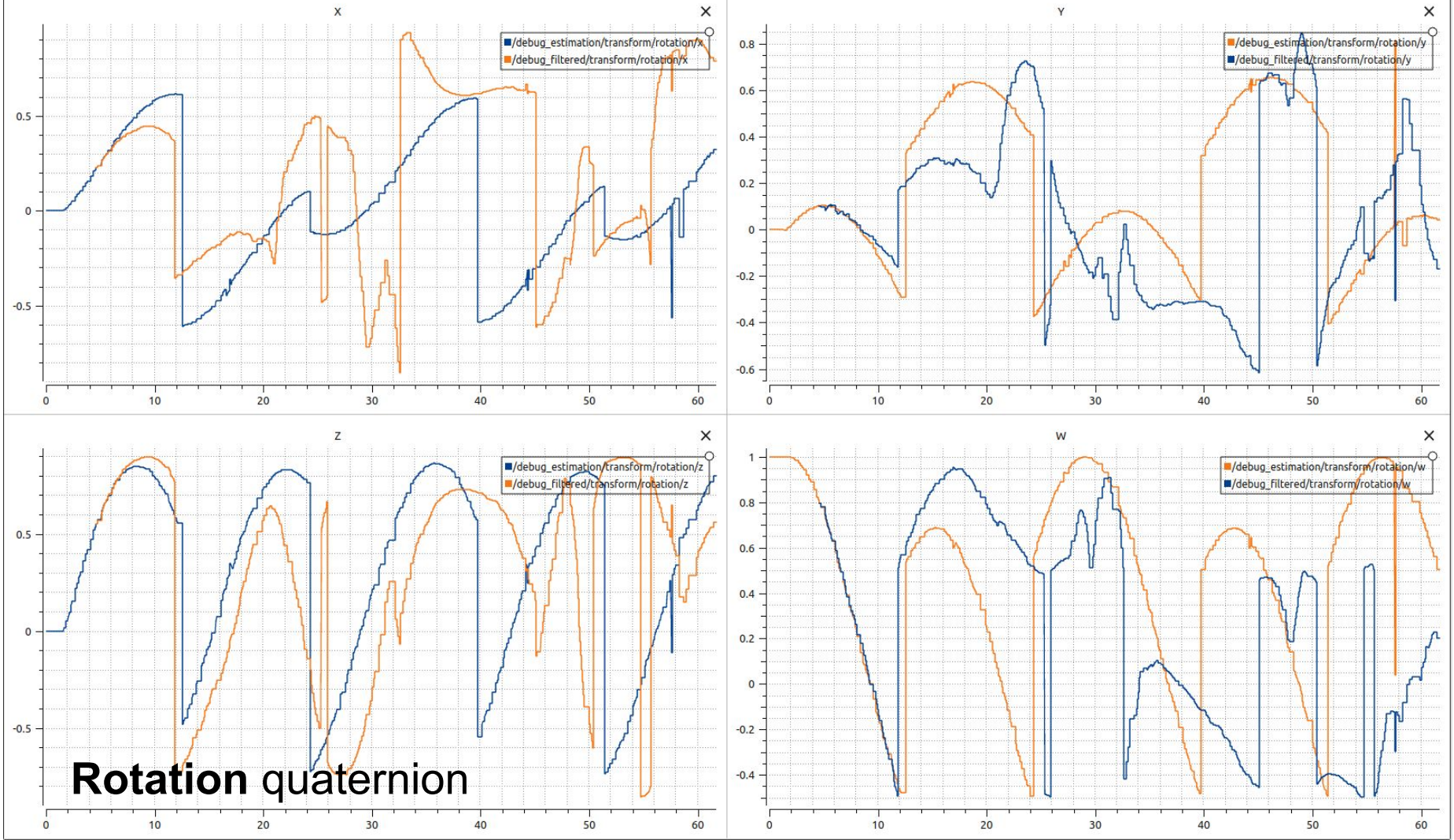
Translation: meters







Rotation (converted to euler): radians


X
/debug_estimation/transform/rotation/x
/debug_filtered/transform/rotation/x
Y
/debug_estimation/transform/rotation/y
/debug_filtered/transform/rotation/y
Z
/debug_estimation/transform/rotation/z
/debug_filtered/transform/rotation/z
W
/debug_estimation/transform/rotation/w
/debug_filtered/transform/rotation/w
Rotation quaternion

Results

Thankfully the improvements from using the tf tree and pose in respect to the **world** frame persist in more complicated scenarios.

- Big stuttering in the final angle estimates are caused from tiny deviations in the raw estimation output. Discontinuities (change in direction) in one axis match stuttering in the other two (singularities?) (marked in red)
- Big stuttering in the raw estimation output does NOT produce ANY visible stuttering in the final output (marked in blue)
- Around t=55s camera visual starts getting lost and there are faulty measurements in the translation estimation too. Stuttering in the orientation there is caused by this (marked in green)

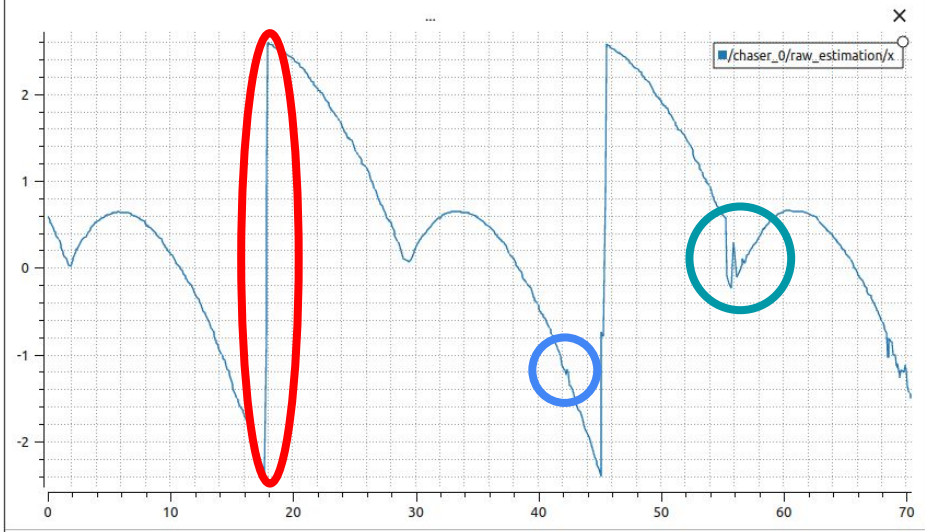
Ideas from improvement

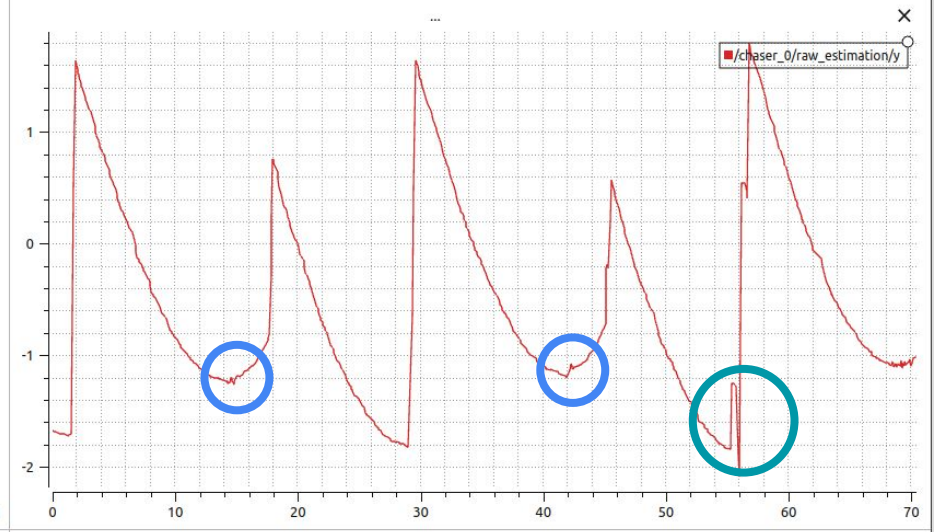
- Rolling average filter on the estimators raw output (might also improve translation)
- In the KF node, get data from the tf tree in respect to the **world** frame and filter that. This might be a bit slower.
- Outlier detection and removal

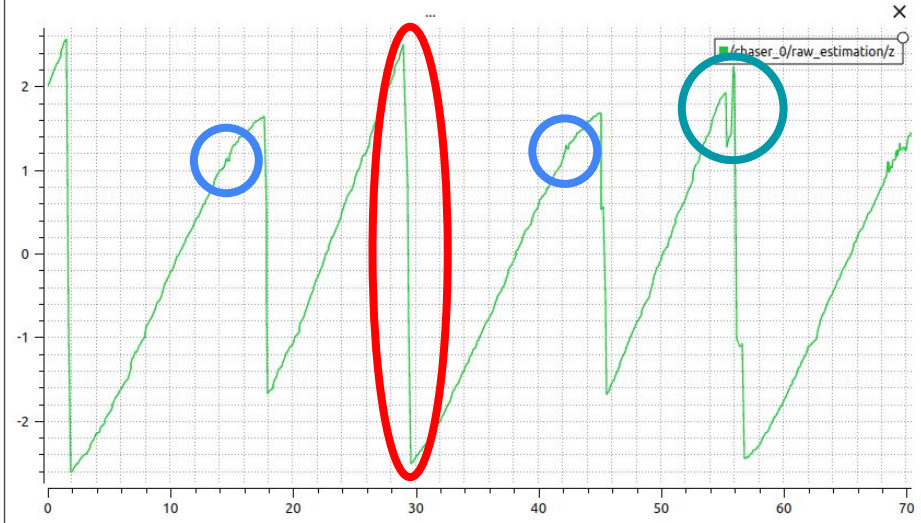
Simulation avg_test_1

Same scenario as before. Added a rolling average filter of size 3

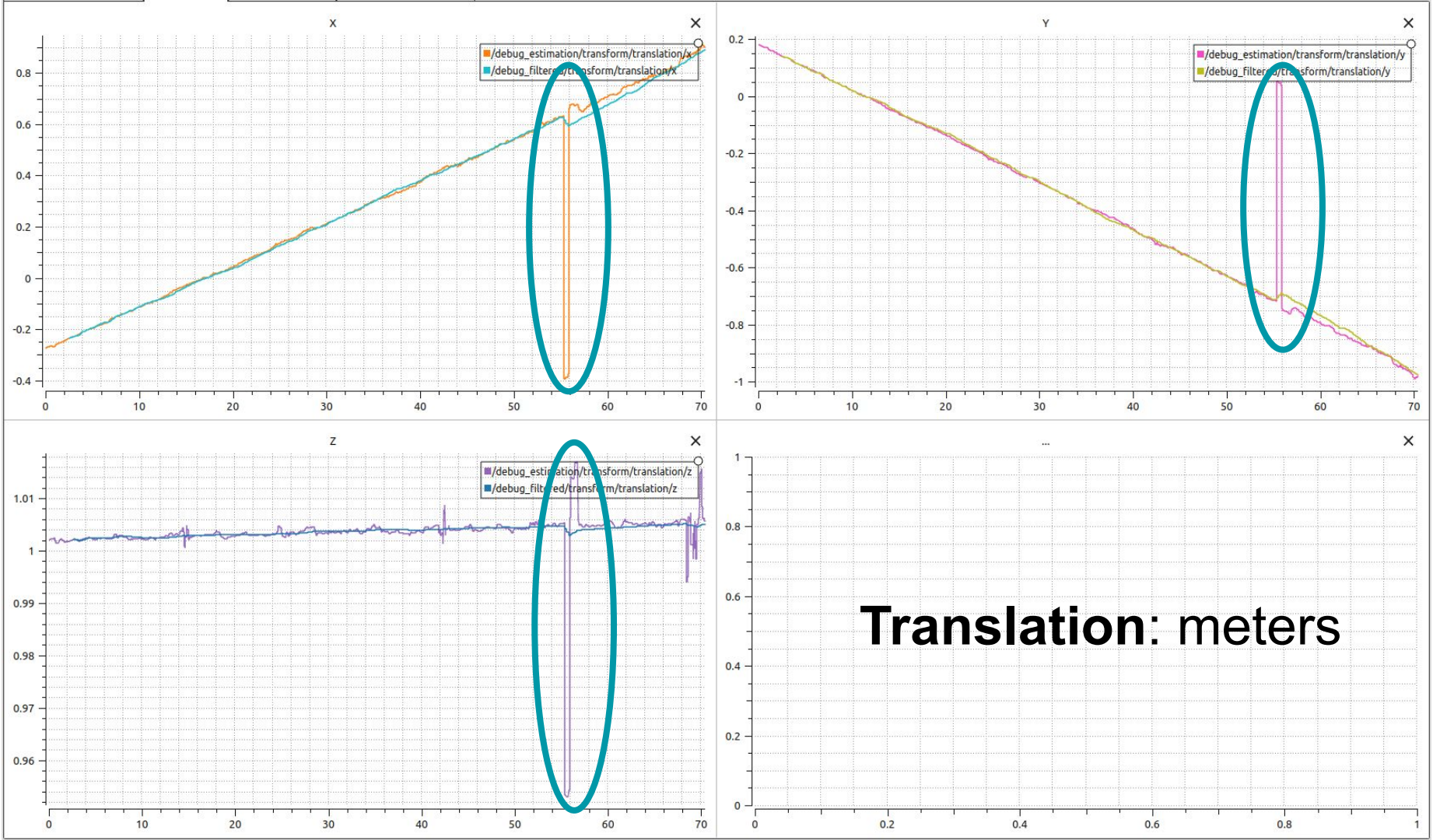
(The following plots do not contain filtered data. For the time being they provide no useful debugging information)

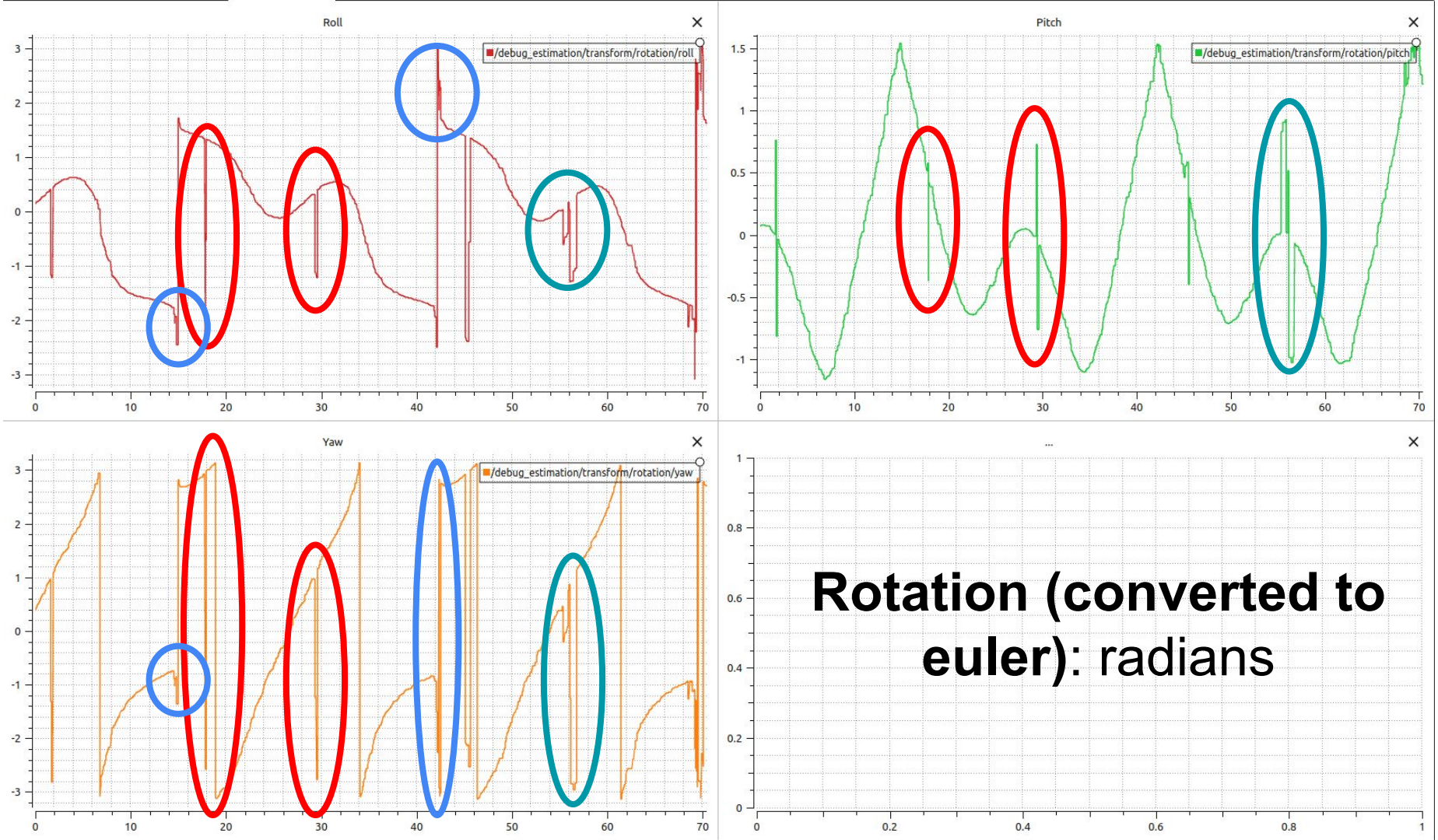


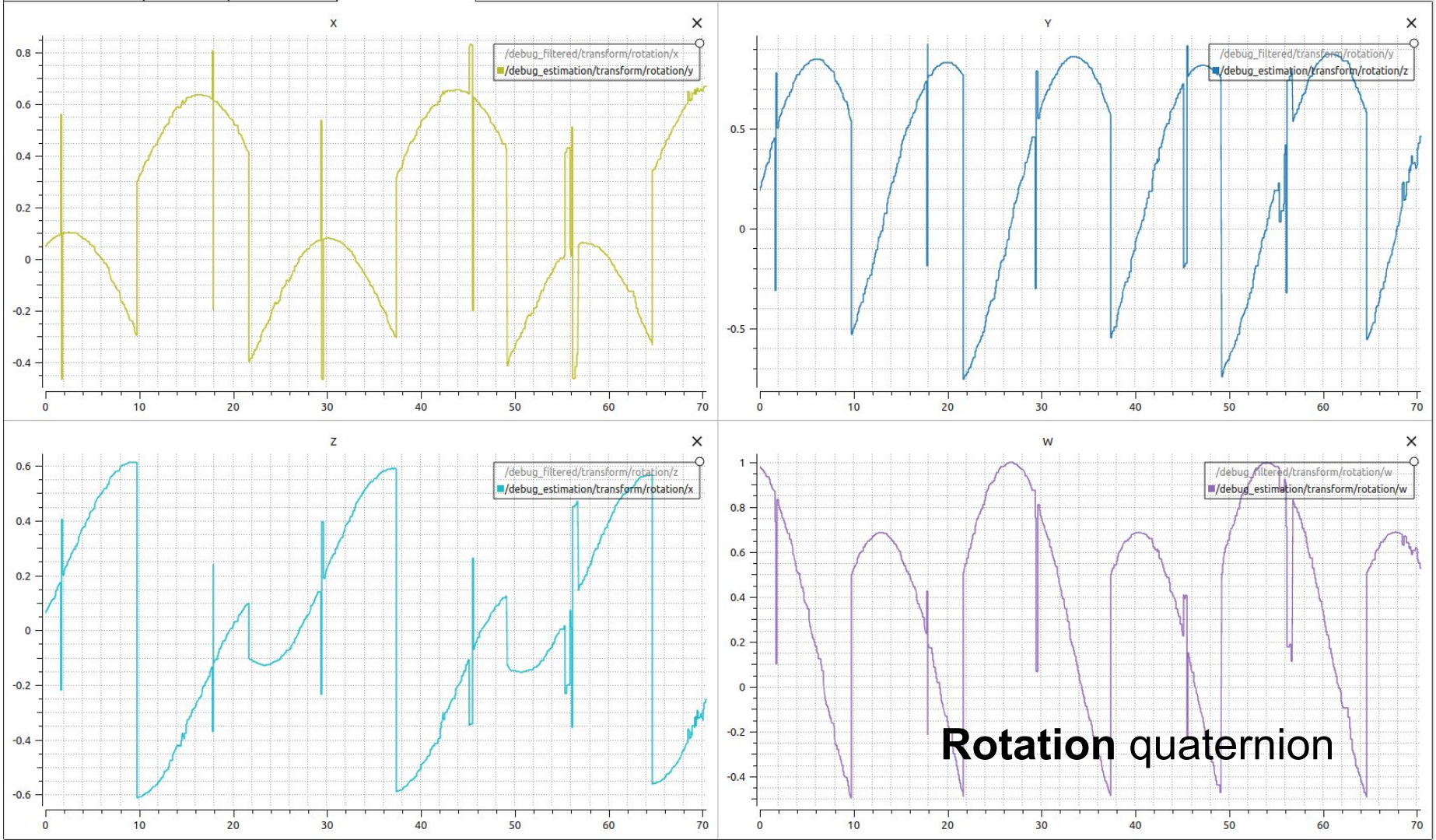




Raw estimation output: radians (in respect to camera_optical NOT world)


X
/debug_estimation/transform/translation/x
/debug_filtered/transform/translation/x
Y
/debug_estimation/transform/translation/y
/debug_filtered/transform/translation/y
Z
/debug_estimation/transform/translation/z
/debug_filtered/transform/translation/z
...
Translation: meters


Roll
/debug_estimation/transform/rotation/roll
Pitch
/debug_estimation/transform/rotation/pitch
Yaw
/debug_estimation/transform/rotation/yaw
Rotation (converted to euler): radians


x
/debug_filtered/transform/rotation/x
/debug_estimation/transform/rotation/y
y
/debug_filtered/transform/rotation/y
/debug_estimation/transform/rotation/z
z
/debug_filtered/transform/rotation/z
/debug_estimation/transform/rotation/x
w
/debug_filtered/transform/rotation/w
/debug_estimation/transform/rotation/w
Rotation quaternion

Results

- Even though the results may seem worst most of the chattering is removed. The peaks observed only happen where there are singularities in the raw estimation (examples in red). This was not observed in the previous simulations and so it might be *fixable by turning off the rolling average filter when there are big changes in the estimation (detecting singularity)*
- There is some chattering where the tiny noise in the raw estimation survived the filter. Should be fixable by increasing the filter length (marked in blue)
- Of course we can again ignore the point where the camera missed completely missed the markers (marked in green)

Ideas for improvement

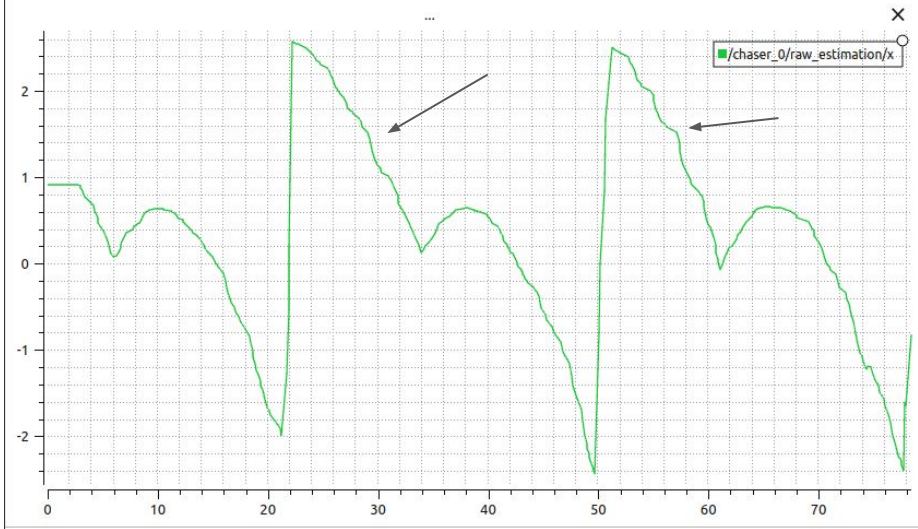
- Increase size of rolling average
- Apply rolling average to the calculated quaternion and NOT the raw output of the estimator
- **Singularities are still a problem**

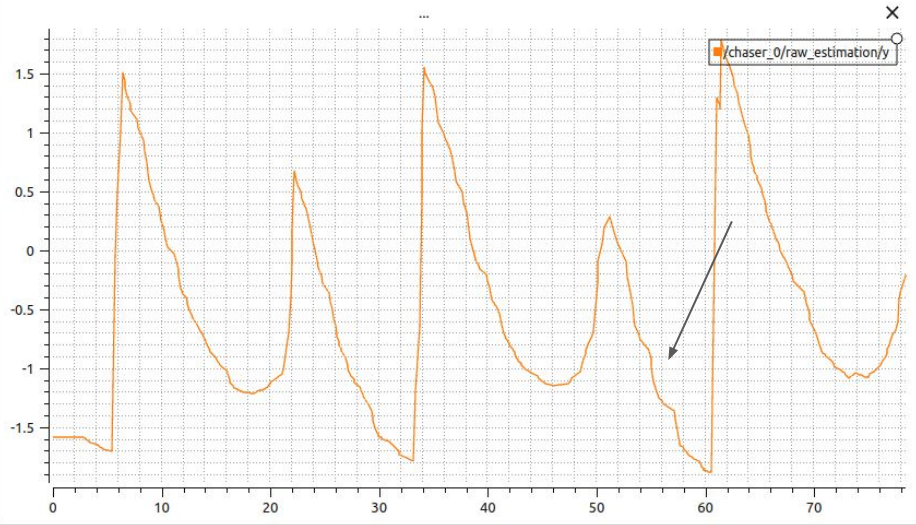
Simulation avg_test_2

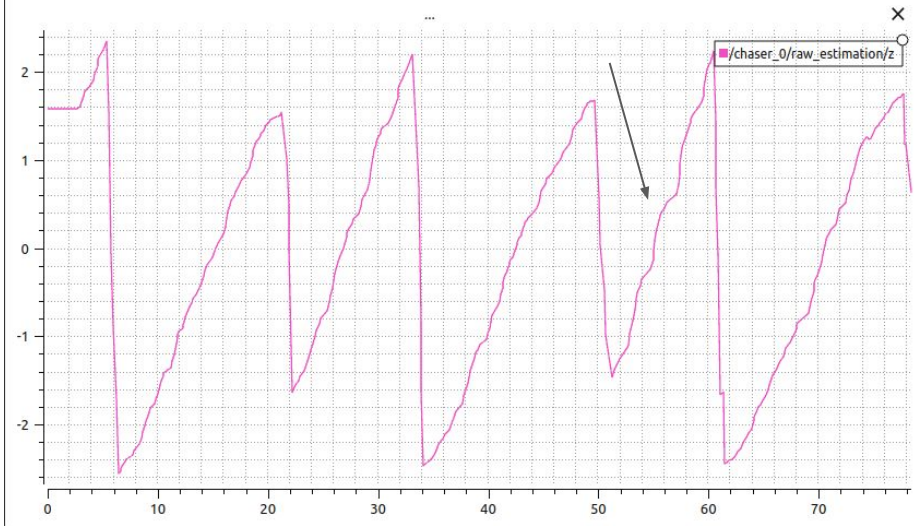
Same scenario as before.

Increased rolling average filter size to 6

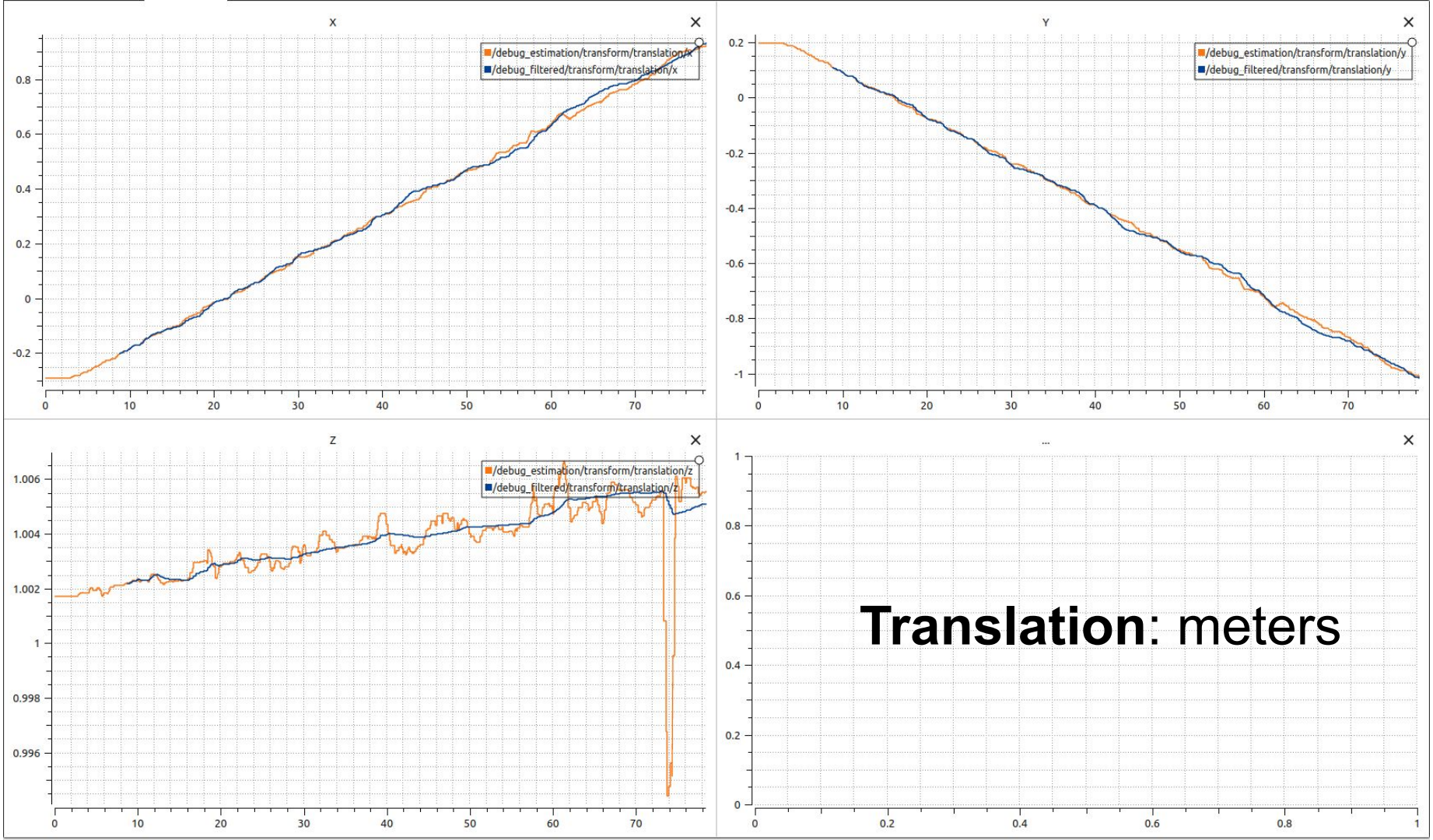
The KF now takes the pose estimation in relation to the **world** system from the tf tree using lookup_transform

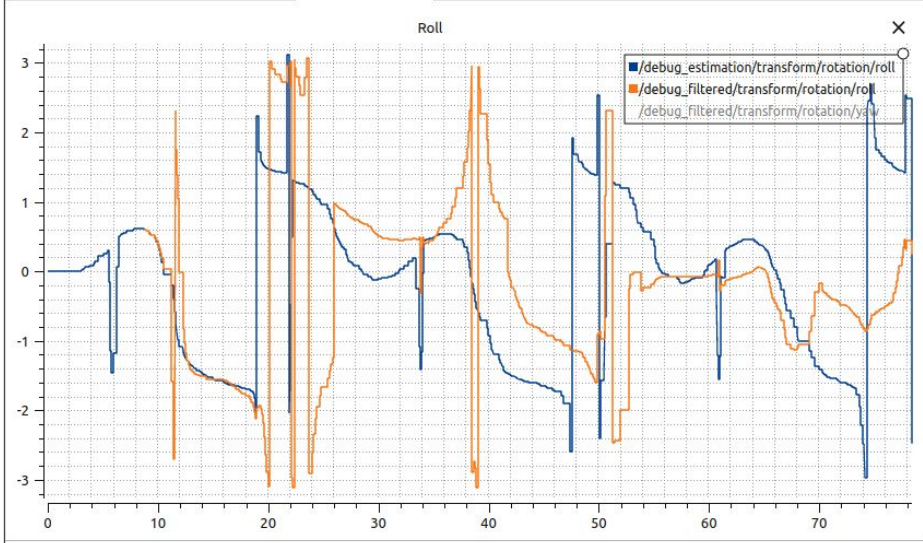


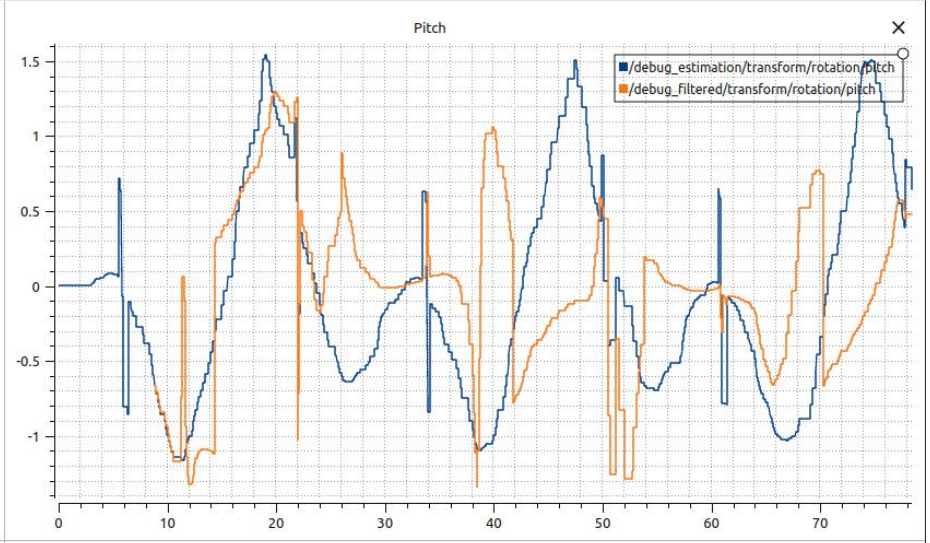


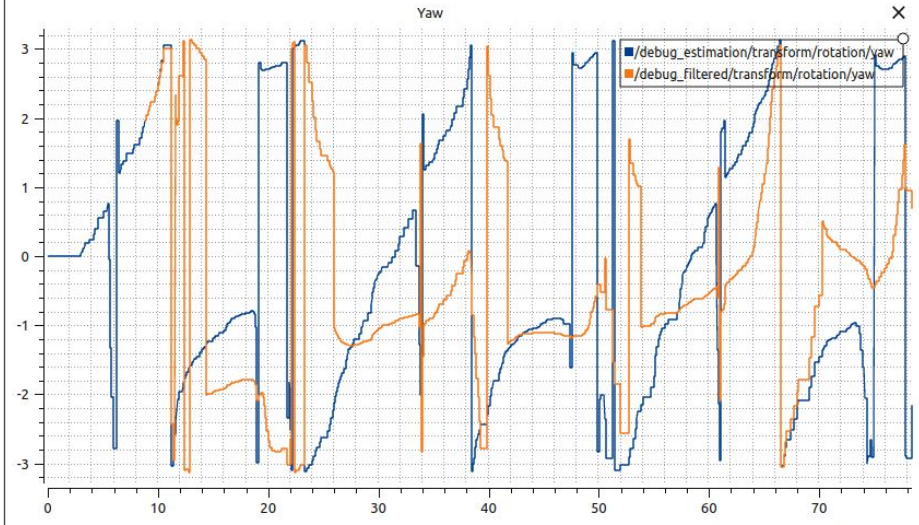


Raw estimation output: radians (in respect to camera_optical NOT world)


X
/debug_estimation/transform/translation/x
/debug_filtered/transform/translation/x
Y
/debug_estimation/transform/translation/y
/debug_filtered/transform/translation/y
Z
/debug_estimation/transform/translation/z
/debug_filtered/transform/translation/z
Translation: meters

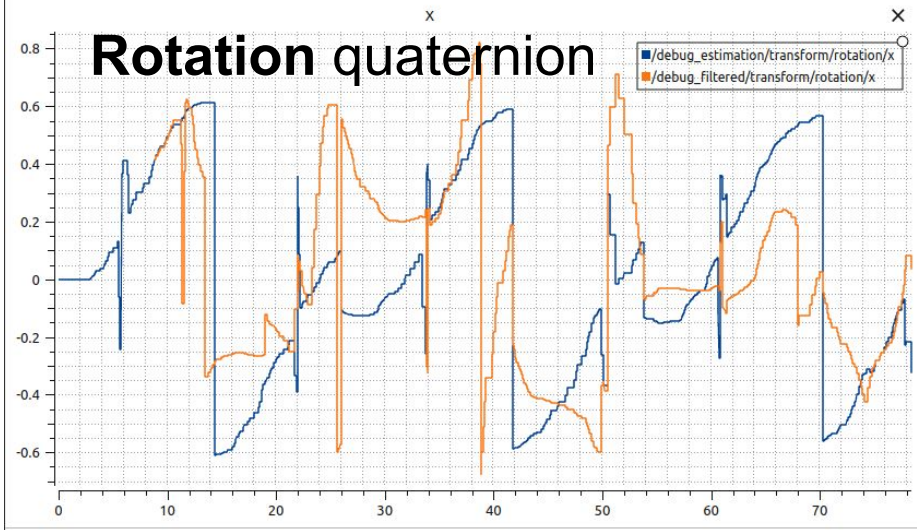


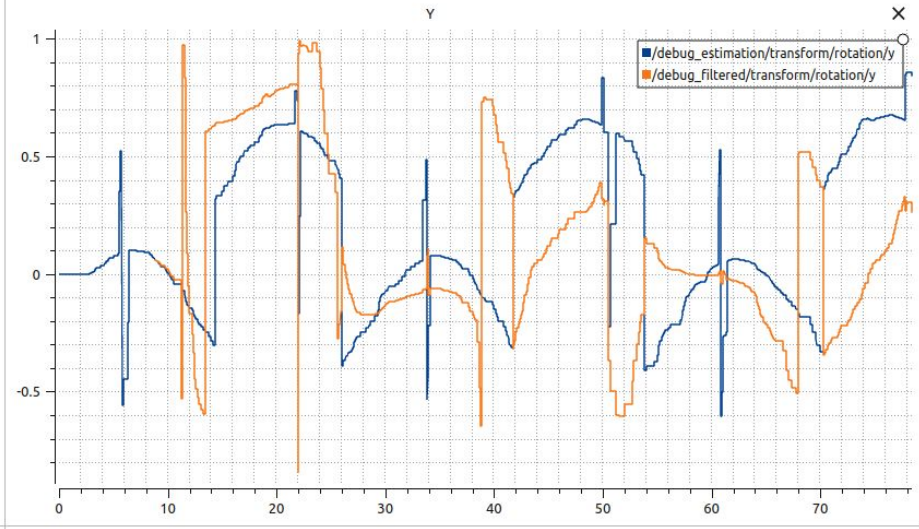


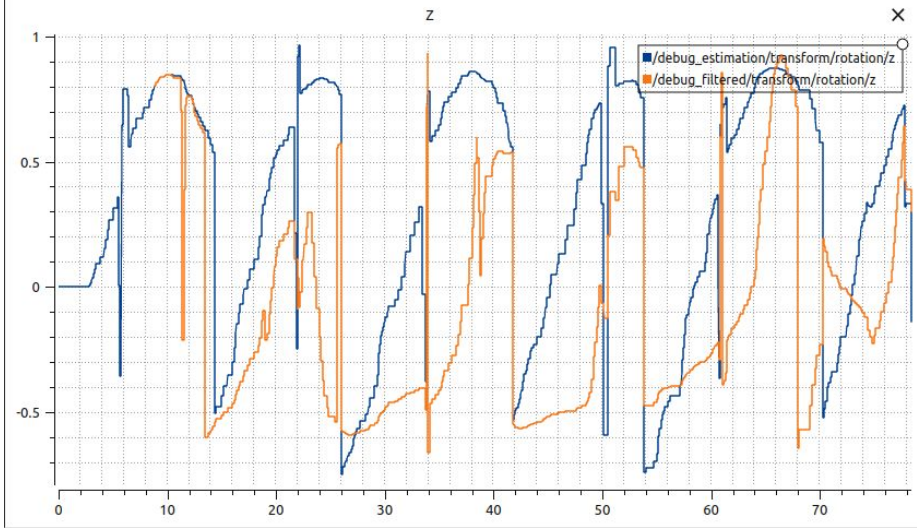


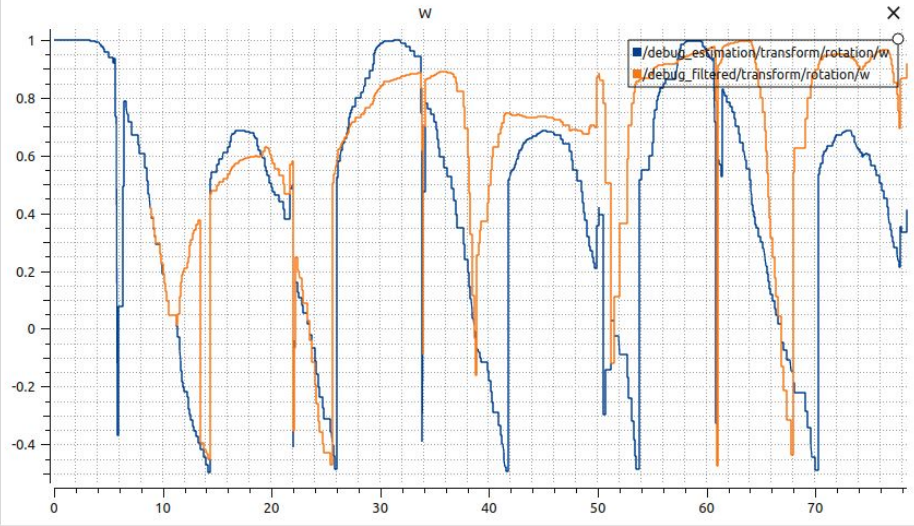
Rotation (converted to euler): radians

Rotation quaternion









Results

- Significant improvements both to estimated and filtered pose
- Increased average window created some lumps in the raw data (examples shown with arrows)
- Chattering is almost gone
- Singularities are still a problem (might improve with the suggestion on p.29)

Index

- **Chaser**: the controlled spacecraft mounted with a camera trying to dock the **target**.
- **Target**: the uncontrolled/uncooperative spacecraft with imprinted markers.
- Kalman gains:
  - **P**: Covariance matrix
  - **R**: Measurement noise
  - **Q**: Process noise