Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-inertialUKF

INERTIAL UNSCENTED FILTER

Prepared by	T. Teil

Status: First Version
Scope/Contents
This module filters incoming star tracker measurements and reaction wheel data in order to get the best possible inertial attitude estimate. Measurements can be coming in from several camera heads.

Rev	Change Description	By	Date
1.0	First documentation	T. Teil	03/02/2019

Contents

---

1 Model Description

This module implements a square-root unscented Kalman Filter in order to achieve it's best state estimate. The estimated state is the attitude (MRPs) and the spacecraft rotation rate in the body frame.

1.1 Filter Setup

The equations and algorithm for the square root uKF are given in "inertialUKF_DesignBasis.pdf" [1] alongside this document.

The filter is therefore derived with the states being $\boldsymbol{X} = \begin{bmatrix}\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}} & \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}\end{bmatrix}^T$

The dynamics of the filter are given in Equations (1). τ is the total torque read in by the wheels.

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}[B]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \tag{1}$$

$$\dot{\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}} = [I]^{-1}\tau \tag{2}$$

The following square-root uKF coefficients are used: $\alpha = 0.02$, and $\beta = 2$.

1.2 Measurements

The measurement model is given in equation 4. Since the input MRP may or may not be in the same "shadow" set as the state estimate, they are assured to be in the same representation. This prevents from getting residuals of 360°.

This is done following these steps:

- Current state estimate and measurements turned to quaternions
- State estimate is transposed (scaled by -1)
- Both quaternions are added and the sum turned to an MRP
- If the sum is greater than one the MRPs were not in the same representation and the measurement is shadowed

$$\boldsymbol{G}_i(\boldsymbol{X}) = \boldsymbol{\sigma} \tag{3}$$

1.3 Measurements

The measurement model is given in equation 4. Since the input MRP may or may not be in the same "shadow" set as the state estimate, they are assured to be in the same representation. This prevents from getting residuals of 360°.

This is done following these steps:

- Current state estimate and measurements turned to quaternions
- State estimate is transposed (scaled by -1)
- Both quaternions are added and the sum turned to an MRP
- If the sum is greater than one the MRPs were not in the same representation and the measurement is shadowed

$$\boldsymbol{G}_i(\boldsymbol{X}) = \boldsymbol{\sigma} \tag{4}$$

2 Module Functions

- **Read ST Messages:** Read in the messages from all available star trackers and orders them with respect to time of measurement.
- **Inertial UKF Agg Gyr Data:** Aggregate the input gyro data into a combined total quaternion rotation to push the state forward. This information is stored in the main data structure for use in the propagation routines.
- **Inertial UKF Time Update:** Performs the filter time update as defined in the baseline algorithm
- **Inertial UKF Meas Update:** Performs the filter measurement update as defined in the baseline algorithm
- **Inertial UKF Meas Model:** Predicts the measurements given current state and measurement model $\boldsymbol{G}$
- **inertial State Prop:** Integrates the state given the $\boldsymbol{F}$ dynamics of the system

3 Module Assumptions and Limitations

The assumptions of this module are all tied in to the underling assumptions and limitations to a working filter. In order for a proper convergence of the filter, the dynamics need to be representative of the actual spacecraft perturbations. Depending on the tuning of the filter (process noise value and measurement noise value), the robustness of the solution will be weighed against it's precision.

The number of measurements and the frequency of their availability also influences the general performance.

4 Test Description and Success Criteria

4.1 Test 0: Individual Methods Tests

The first test in this suite runs methods individually:

- Read STMessages: This test sends 3 ST messages in the wrong order (1.25s, 1s, 0.5s), and tests that the method organizes them chronologically relative to their timeTag.
- Clean Update: This test calls the Clean method and ensures previous sBar, covariance, and state values replaced the potentially erroneous current values.
- Faulty Measurement Update: This test gives a negative wM vector and ensures that the badUpdate is triggered
- Faulty Time Update: The same test is run on the Time update
- Wheel acceleration in the inertial Prpagation: This test compares the output array when the wheel acceleration is computed with expected values

4.2 Test 1: StatePropInertialAttitude

This test runs a pure propagation test. The states are set to a fixed value and integrated with the filter. This shows filter stability in the simple case and a very low tolerance for error is permitted (1E-10).

4.3 Test 2: StatePropRateInertialAttitude

This test runs a pure propagation test while adding gyro data. This ensures those messages are being read and used properly.

4.4 Test 3: StateUpdateInertialAttitude

Given no dynamics, this test runs the filter by giving two star tracker messages. For the first half of the simulation, the measurements both read attitude values of $\boldsymbol{\sigma} = \begin{bmatrix} 0.3 & 0.4 & 0.5 \end{bmatrix}$. After 1000s, the measurement changes to $\boldsymbol{\sigma} = \begin{bmatrix} 1.2 & 0 & 0 \end{bmatrix}$.

Figures 1 and 2 show the results for the states and covariance values.

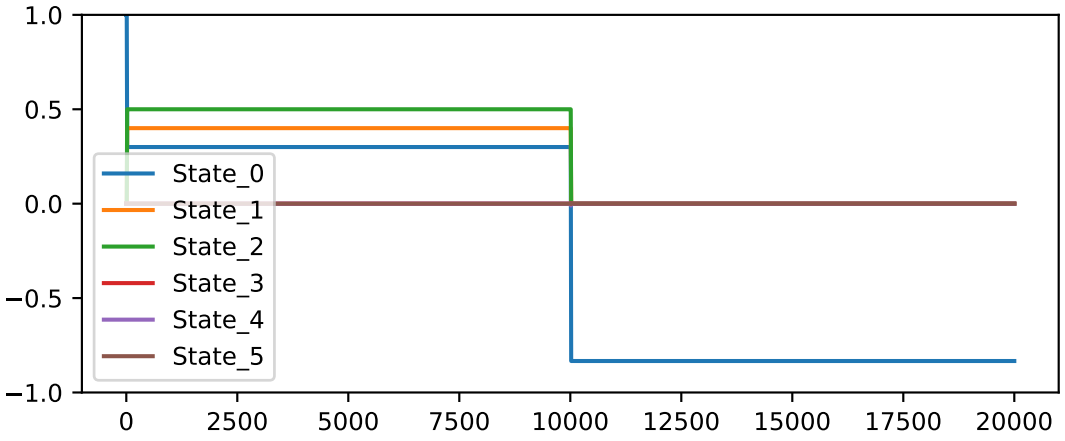

Fig. 1: Test 1 State convergence

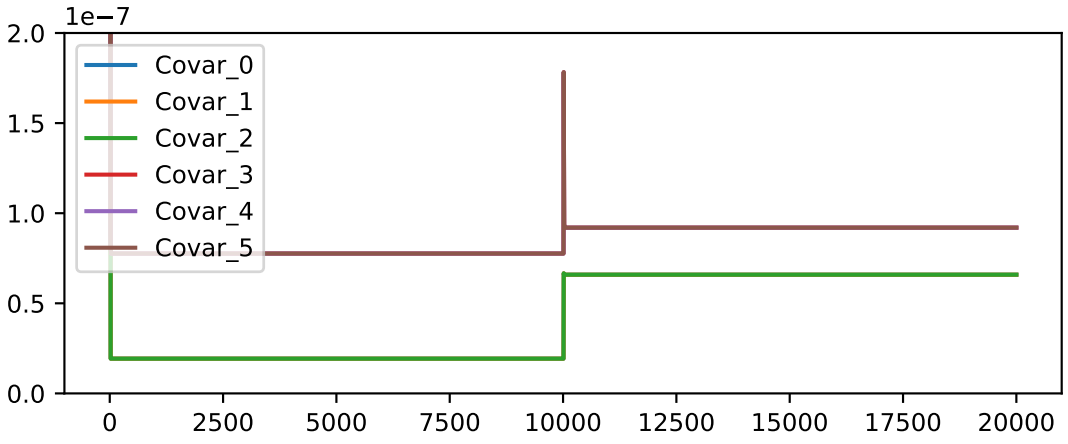

Fig. 2: Test 1 Covariance convergence

4.5 Test 4: StateUpdateRWInertialAttitude

This last test runs the filter by giving two star tracker messages as well as reaction wheels. For the first half of the simulation, the measurements both read attitude values of $\boldsymbol{\sigma} = \begin{bmatrix} 0.3 & 0.4 & 0.5 \end{bmatrix}$. After 1000s, the measurement changes to $\boldsymbol{\sigma} = \begin{bmatrix} 1.2 & 0 & 0 \end{bmatrix}$. Wheel speeds are set to $\boldsymbol{w}_s = \begin{bmatrix} 0.1 & 0.01 & 0.1 \end{bmatrix}$.

Figures 1 and 2 show the results for the states and covariance values.

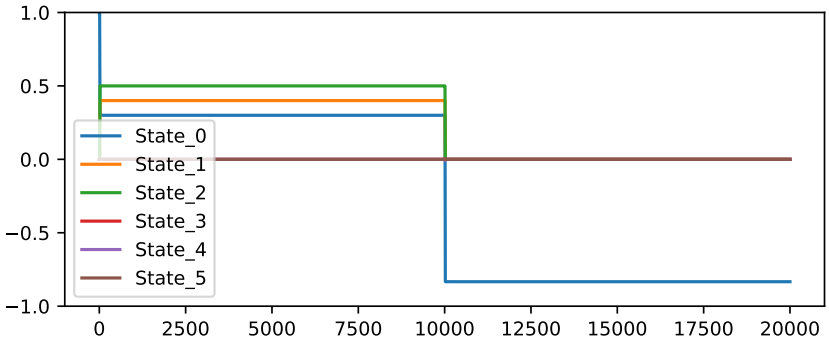

Fig. 3: Test 3 State convergence

5 Test Parameters

Output Value Tested	States Tolerated Error	Covariance Tolerated Error
Test 0	1e-10	N/A
Test 1	1e-10	N/A
Test 2	0.001	Increase
Test 3	1e-05	Increase
Test 4	1e-05	N/A

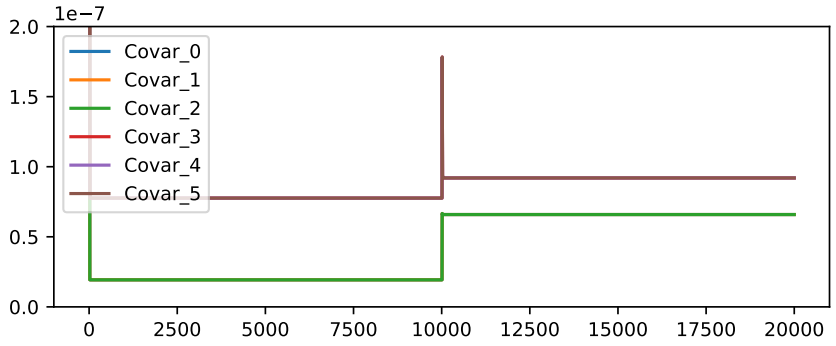

Fig. 4: Test 3 Covariance convergence

6 Test Results

Table 2: Test results

Check	Pass/Fail
Test 0	PASSED
Test 1	PASSED
Test 2	PASSED
Test 3	PASSED
Test 4	PASSED

7 User Guide

7.1 Filter Set-up, initialization, and I/O

In order for the filter to run, the user must set a few parameters:

- The unscented filter has 3 parameters that need to be set, and are best as:
```
filterObject.alpha = 0.02
filterObject.beta = 2.0
filterObject.kappa = 0.0
```

- The star trackers:
```
ST1Data = inertialUKF.STMessage()
ST1Data.stInMsgName = "star_stracker_1_data"
ST1Data.noise = [0.00017 * 0.00017, 0.0, 0.0,
0.0, 0.00017 * 0.00017, 0.0,
0.0, 0.0, 0.00017 * 0.00017]
ST2Data = inertialUKF.STMessage()
ST2Data.stInMsgName = "star_tracker_2_data"
ST2Data.noise = [0.00017 * 0.00017, 0.0, 0.0,
0.0, 0.00017 * 0.00017, 0.0,
0.0, 0.0, 0.00017 * 0.00017]
STList = [ST1Data, ST2Data]
filterObject.STDatasStruct.STMessages = STList
filterObject.STDatasStruct.numST = len(STList)
```

- The initial covariance:
```
Filter.covar =
[1., 0.0, 0.0, 0.0, 0.0,
0.0, 1., 0.0, 0.0, 0.0,
0.0, 0.0, 1., 0.0, 0.0,
0.0, 0.0, 0.0, 0.02, 0.0,
0.0, 0.0, 0.0, 0.0, 0.02]
```
- The initial state :
```
Filter.state =[0.0, 0.0, 1.0, 0.0, 0.0]
```
- The low pass filter for the accelerometers :
```
lpDataUse = inertialUKF.LowPassFilterData()
lpDataUse.hStep = 0.5
lpDataUse.omegCutoff = 15.0/(2.0*math.pi)
filterObject.gyroFilt = [lpDataUse, lpDataUse, lpDataUse]
```

The messages must also be set as such:

- `filterObject.navStateOutMsgName = "inertial_state_estimate"`
- `filterObject.filtDataOutMsgName = "inertial_filter_data"`
- `filterObject.massPropsInMsgName = "adcs_config_data"`
- `filterObject.rwSpeedsInMsgName = "reactionwheel_output_states"`
- `filterObject.rwParamsInMsgName = "rwa_config_data_parsed"`
- `filterObject.gyrBuffInMsgName = "gyro_buffer_data"`

REFERENCES

[1] R. van der Merwe. The square-root unscented kalman filter for state and parameter-estimation. Acoustics, Speech, and Signal Processing, 2001.