NavFuse Unscented Kalman Filter Class Design Description

Parker Barrett

February 6th, 2023

1 Class Overview

- Header File: NavFuse/include/filter/UnscentedKalmanFilter.hpp
- Implementation: NavFuse/src/filter/UnscentedKalmanFilter.cpp

The NavFuse Unscented Kalman Filter class contains functions for Unscented Kalman Filter initialization, prediction and measurement updates. The Unscented Kalman Filter class is derived from the Base Filter class.

2 Derived Class Members

2.1 UnscentedKalmanFilter::filterInitialize()

- Inputs
  - Eigen::VectorXd x0: nx1 dimensional vector containing the initial filter state
  - Eigen::MatrixXd P0: nxn dimensional matrix containing the initial filter covariance
- Outputs
  - No Outputs
- Algorithm
  - The private class member UnscentedKalmanFilter::filterState is initialized to the value of the input state vector, x0
  - The private class member UnscentedKalmanFilter::filterCovariance is initialized to the value of the input covariance matrix, P0

2.2 UnscentedKalmanFilter::filterCovariance

- Data Type: Eigen::MatrixXd
- Description: nxn dimensional matrix which stores the most up to date filter covariance estimate

2.3 UnscentedKalmanFilter::filterState

- Data Type: Eigen::VectorXd
- Description: nx1 dimensional vector which stores the most up to date filter state estimate

2.4 UnscentedKalmanFilter::getCovariance()

- Inputs
  - No Inputs
- Outputs
  - Eigen::MatrixXd Pk: nxn dimensional filter covariance matrix
- Algorithm
  - Return the current filter covariance matrix

2.5 UnscentedKalmanFilter::getState()

- Inputs
  - No Inputs
- Outputs
  - Eigen::VectorXd xk: nx1 dimensional filter state vector
- Algorithm
  - Return the current filter state vector

3 Public Class Members

The following sub-sections describe the inputs, outputs and internal algorithms used in the public interface of the Unscented Kalman Filter class.

3.1 UnscentedKalmanFilter::filterUkfPredict()

- Inputs
  - Eigen::VectorXd Wi: (2n+1)x1 dimensional vector of sigma point weights
  - Eigen::MatrixXd yi: nx(2n+1) dimensional matrix with each column corresponding to one sigma vector passed through the full nonlinear process dynamics - $\mathbf{y}_i = f(\mathbf{x}_i)$
  - Eigen::MatrixXd Qk: nxn dimensional discrete time process noise matrix
- Outputs
  - No Outputs
- Algorithm
  - The state vector mean is predicted via the weighted average of the sigma points: $\mathbf{x}_{k+1} = \sum_{i=0}^{2n} \mathbf{w}_i \mathbf{y}_i$
  - The covariance matrix is predicted via $\mathbf{P}_{k+1} = \sum_{i=0}^{2n} \mathbf{w}_i (\mathbf{y}_i - \mathbf{x}_{k+1})(\mathbf{y}_i - \mathbf{x}_{k+1})^T + \mathbf{Q}_k$

3.2 UnscentedKalmanFilter::filterUkfUpdate()

- Inputs
  - Eigen::MatrixXd yi: nx(2n+1) dimensional matrix with each column corresponding to one sigma vector passed through the full nonlinear process dynamics - $\mathbf{y}_i = f(\mathbf{x}_i)$
  - Eigen::VectorXd Wi: (2n+1)x1 dimensional vector of sigma point weights
  - Eigen::VectorXd zi: mx(2n+1) dimensional matrix of predicted measurements at each sigma point
  - Eigen::MatrixXd zk: mx1 dimensional measurement vector
  - Eigen::MatrixXd Rk: mxm dimensional discrete time measurement noise matrix
- Outputs
  - No Outputs
- Algorithm
  - The predicted measurement is computed as a weighted average of the predicted measurements at each sigma point by: $\hat{\mathbf{z}} = \sum_{i=0}^{2n} \mathbf{w}_i \mathbf{z}_i$
  - The measurement covariance is computed as: $\mathbf{S} = \sum_{i=0}^{2n} \mathbf{w}_i (\mathbf{z}_i - \hat{\mathbf{z}})(\mathbf{z}_i - \hat{\mathbf{z}})^T + \mathbf{R}_k$
  - The cross-correlation between the measurement and state is computed by: $\mathbf{T} = \sum_{i=0}^{2n} \mathbf{w}_i (\mathbf{y}_i - \mathbf{x}_{k+1})(\mathbf{z}_i - \hat{\mathbf{z}})^T$
  - The Kalman Gain is computed by: $\mathbf{K} = \mathbf{T}\mathbf{S}^{-1}$
  - The measurement residual is computed as: $\nu = \mathbf{z}_k - \hat{\mathbf{z}}$
  - The state estimate is updated by the standard Kalman Filter Update Equation: $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{K}\nu$
  - The covariance is updated by: $\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{K}\mathbf{S}\mathbf{K}^T$

References

[1] H.S.Chadha, “The Unscented Kalman Filter: Anything EKF can do I can do it better” https://towardsdatascience.com/the-unscented-kalman-filter-anything-ekf-can-do-i-can-do-it-better-ce7c773cf88d.