ENSEMBLE FILTERING FOR STATE ESTIMATION,

NICHOLAS GURNARD [NGURNARD@SEAS], PRANAV SHAH [PRANAVPS@SEAS], AADIT PATEL [AADITP@SEAS] ,

ABSTRACT. In time critical systems, it is essential that the state estimation pipeline is quick and accurate. Current methods often struggle to achieve both. In this project we attempt to rectify this problem by constructing an ensemble filtering method by combining the state estimates from multiple filters to get a better overall estimate. The ensemble consists of fast, weak learners that can compete with an accurate, but slow filtering algorithm. There are 3 methods to form the ensemble: a simple average, a perceptron network, and a dense neural network with loss functions of MSE loss and a custom loss function. We demonstrate that the performance of the ensemble, in specific the dense neural network, is better than the individual performance of each filter tested on the EuRoC dataset.

1. INTRODUCTION

State estimation of a robot is critical for any robotics pipeline, where the Kalman filter variants are the backbone of most modern state estimation algorithms. However, the dynamics of a system are often hard to model, and thus a single Kalman filter variant may not be able to capture all of the intricacies of the dynamics. There are many variations of the Kalman filter that attempt to reduce assumptions about the system dynamics, for example the Unscented Kalman Filter (UKF) and Error State Kalman Filter (ESKF), however still rely on a simplification of the true model dynamics [10]. Previous work has introduced the idea of using multiple Kalman filters together in order to limit model simplification and assumptions [9] [4] [2]. Additionally, in a system that experiences rapid changes in its state, for example aggressive drone flight, the Kalman filter can be slow in recovering the true state once there is a major deviation [10].

When tasked with a real system, the speed of the filter is often a bottleneck that causes the system to fail [1] [11]. However, these same aggressive systems also rely on excellent state estimation to avoid failure. Fast, but less accurate, filtering algorithms are often the replacement for slow, accurate algorithms [11]. In this project, we introduce the idea of combining fast and less accurate filters together to get a better state estimate while still saving on computation time.

The proposed algorithm has 2 main applications: time critical systems that require an ensemble of fast but less accurate filters, and delay tolerant systems that would allow for an ensemble of several slow but extremely accurate filters. This project borrows the ensemble learning idea from field of machine learning, where several weak learners are combined together in an ensemble to create a powerful unbiased estimator such as random forests, which can then be passed through a neural network to capture the nonlinearity of the dynamics [12] [13].

1.1. **Contributions.**

We demonstrate that the performance of the ensemble is better than the performance of each individual filter for system state estimation. We present 3 methods to combine the filters, namely simple averaging, a single layer perceptron, and a neural network with loss functions of MSE loss and custom loss function.

2. BACKGROUND

Kalman filters are a family of algorithms that use a series of sensor measurements observed over time and produce state estimates for the system. The Kalman filter and its variants broadly work in 2 stages, namely propagation step and update step. In the propagation step, we use the estimate of the state at time k and estimate the state at time $k+1$ before the observation arrives. Once the observation is received, the update step is executed and the estimate of the state for time $k+1$ is refined. This process is repeated recursively for each new observation received.

If the system has linear dynamics, linear observation models, and Gaussian noise in the sensor readings, the Kalman filter is the most efficient state estimator. But as in most practical systems, the system dynamics and/or the observation models are nonlinear which lead to degraded performance on the vanilla Kalman filters. There are many variations like UKF and Extended Kalman Filters (EKF) which approximate the nonlinear dynamics of the system to a linear function and then use the propagate and update equations from the Kalman filter algorithm. Since these variations approximate the nonlinearity in the system dynamics, they are often noisy in their estimates. Different filters use different techniques such has Taylor series approximation, sigma points transformation etc. for the non-linear to linear approximation.

The inspiration for ensemble systems comes from the idea of random forests. In random forest algorithm, multiple decision trees are combined to produce a prediction output. Each decision tree in the ensemble is a weak learner but their combination produces an efficient result that is unbiased. This idea can be extended to filtering systems as well. Modern robotics systems require fast and accurate filtering system for efficient state estimation and trajectory tracking. An accurate but slow filter and a fast but inaccurate filter can lead to poor trajectory tracking and produce drifts in the robots movement. Thus, in this project we develop an ensemble based filtering system that uses multiple fast filters to produce an accurate estimate of the system's state.

3. Related Work

Multi stage filtering is an effective way to combine measurements from various sensors. This idea is demonstrated in [1] [6] [7]. [1] demonstrates a 3 stage system to combine the sensor measurements from a stereo camera and IMU. It's implementation follows that of [6] where the IMU sensor data is used to propagate the state of the system and when the camera measurements are received the state update step is performed. [7] implements the idea in [6] to run on-board a quadrotor. This implementation is made highly optimized for a edge-device to run full time.

There are several variations of Kalman filters to approximate the non-linearity in the system dynamics. [2] implements an unscented Kalman filter to estimate the orientation of a non-linear system. [3] and [4] implement an extended Kalman filter and error state Kalman filter respectively. [4] implementation is VIO based while that of [2] is only inertial odometry based filtering system. [5] describes a complimentary filter approach to solving the filtering problem for non-linear systems. The idea of using multiple filters to give a more robust estimate is introduced in [9].

4. Approach

4.1. **The Filter Variants Implemented to Incorporate in the Ensemble:** In this project we implement and ensemble the UKF [2], ESKF [4] and Complementary Filter (CF) [5]. We focus on orientation estimation since it is often the hardest part of the state to estimate. The datasets used for all analyses in this project come from the EuRoC MAV dataset [8]. The EuRoC MAV dataset has 5 datasets of varied complexity levels of drone flight. 2 datasets have a simple environment with less aggressive maneuvers labeled "easy", 1 dataset labeled "medium" and 2 datasets labeled "hard".

4.2. **Training the Perceptron and Neural Network:** The datasets are split 60-40 into training and tests sets. Machine Hall (MH) 01 (easy), MH-03 (medium) and MH-05 (hard) were used for training the ensemble network and MH-02 (easy) and MH-04 (hard) were used for testing.

The filters assume that the origin of the world frame is the initial pose of the drone at t_0, which is different from the Vicon (ground truth) world frame. In order to correctly compare against the Vicon system, a corrective translation and rotation was applied to all of the states of the filters based on the Vicon initial pose. The corrected estimates from the 3 filters are saved to later incorporate into the ensemble for training.

The orientation estimates from the 3 ensemble filters were stacked together along with the ground truth after aligning the timestamps of the filter outputs. Aligning the filter outputs is critical since the ESKF uses vision measurements and IMU propogation while the CF and UKF rely on the IMU only. Timestamp matching is performed by subtracting one timestamp of the filter from all the timestamps in the ground truth readings and then finding the argmin of the absolute difference to find the index of the reading to consider. We end up with 3 datasets of shape $(N \times 4)$, the 4th column being the ground truth labels, which will be used to train 3 perceptron networks and 3 neural networks, one each for roll, pitch and yaw. Each perceptron network takes 3 inputs at each timestep and produces 1 linear estimate. Each neural network takes 3 inputs at each timestamp and are passed through a hidden layer of dimension 5 with a ReLU activation function, which are then fully connected into a single estimate. The predictions are compared with the ground truth estimates using mean-squared-error (MSE) loss or custom loss function, which is then used for backpropagation and updating the weights. Since orientation estimates, namely roll, pitch, and yaw angles, are nonlinear, the choice of loss functions evaluate a linear distance loss (MSE) and an nonlinear function that considers the double-cover with angle wrap around. In particular, the loss between angles 359° and 1° should reflect an absolute distance of 2° and not 358°. The custom loss function takes the following equation:

$$loss = \left|((\theta_{estimate} - \theta_{truth}) + 180) \bmod (360) - 180\right| \tag{1}$$

The optimizer we have used is the Adam optimizer which takes in arguments for learning rate and a regularizing weight-decay parameter.

4.3. **Testing and Validation of the Ensemble:** The training dataset is split into training and validation datasets with a 80-20 split. Mini-batches from the training dataset are used to the train the network. Once the network is trained, it is verified with the verification set and saved offline for testing. The saved models are then tested on the datasets MH-02 and MH-04. The performance of the model is evaluated by computing MSE loss and the loss described in Equation 1 between the predicted states and the ground truth values. The states obtained from each filter, the ensemble state estimate, the simple average, and the ground truth were then be plotted together for evaluation.

5. Experimental Results

The following figures illustrate the results for the dense neural network ensemble with MSE loss, where ESKF is the green line, UKF is the pink line, CF is the red line, the ground truth is the black line, and the network output is the blue line.

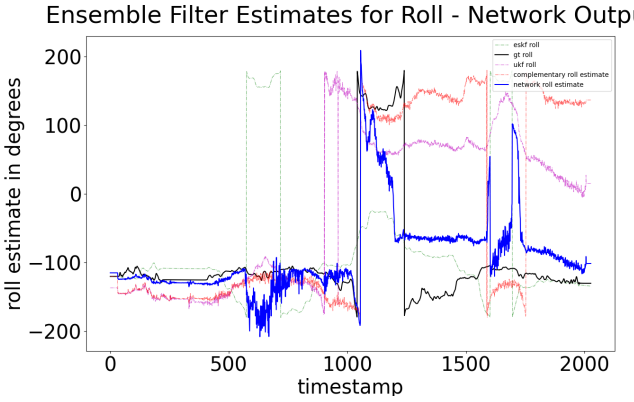

FIGURE 1. Roll Estimates for Dataset 4

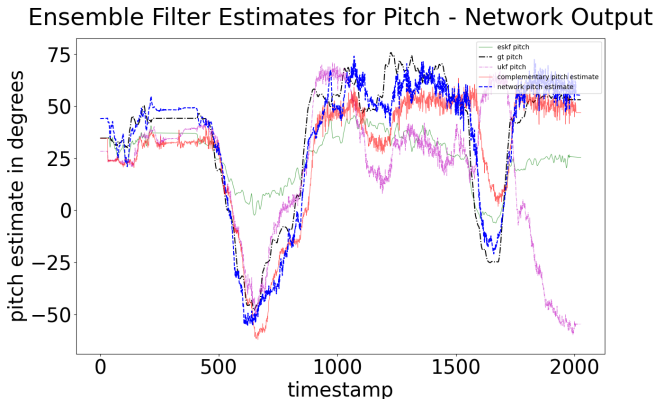

FIGURE 2. Pitch Estimates for Dataset 4

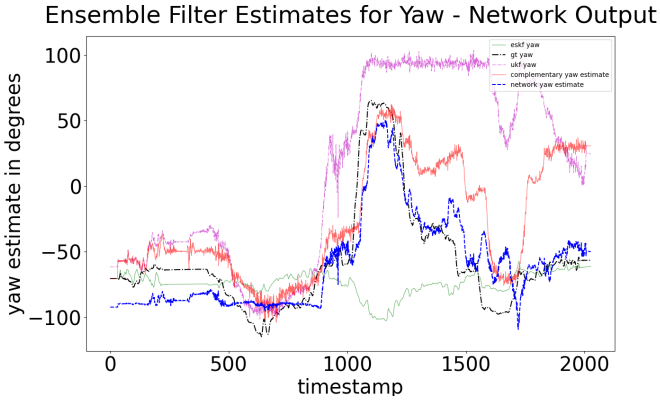

FIGURE 3. Yaw Estimates for Dataset 4

The following figures illustrate the results for simple average, dense NN with MSE loss and ground truth, where simple avearge is the red line, dense NN is the blue line and the ground truth is the black line.

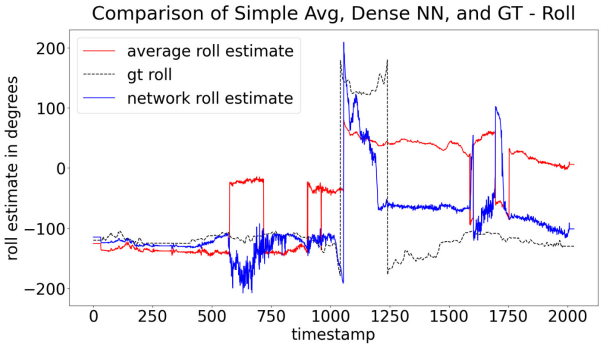

FIGURE 4. Simple average, Dense NN and GT comparision for Dataset 4 for Roll

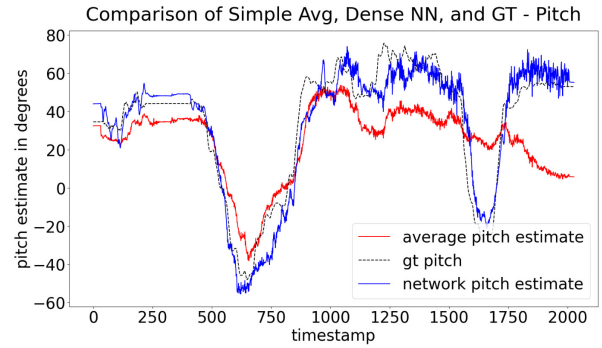

FIGURE 5. Simple average, Dense NN and GT comparision for Dataset 4 for Pitch

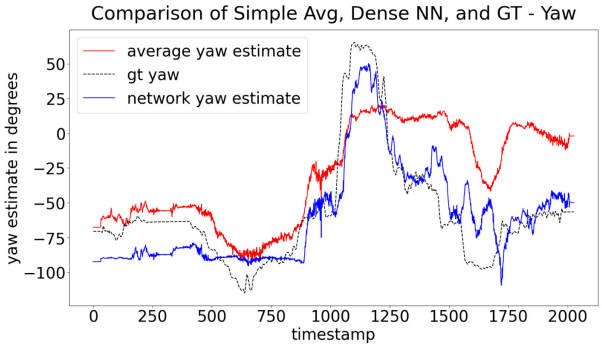

FIGURE 6. Simple average, Dense NN and GT comparision for Dataset 4 for Yaw

RMSE between filter estimates and ground truth for roll, pitch and yaw - MH-02 DATASET			
Filter Name	RMSE for roll	RMSE for pitch	RMSE for yaw
UKF	142.51	21.70	137.07
ESKF	140.04	25.11	97.78
CF	146.53	28.79	109.26
Ensemble Filter (Simple Average)	128.14	13.75	98.17
Ensemble Filter (Perceptron - MSE Loss)	114.75	18.34	95.43
Ensemble Filter (Dense - MSE Loss)	113.59	11.22	130.89
Ensemble Filter (Dense - Custom Loss)	136.02	11.23	83.46

TABLE 1. Summary of results - MH-02

RMSE between filter estimates and ground truth for roll, pitch and yaw - MH-04 DATASET			
Filter Name	RMSE for roll	RMSE for pitch	RMSE for yaw
UKF	144.98	42.37	87.84
ESKF	115.31	23.46	49.68
CF	157.72	17.68	43.34
Ensemble Filter (Simple Average)	108.25	23.67	42.27
Ensemble Filter (Perceptron - MSE Loss)	86.78	16.01	35.42
Ensemble Filter (Dense - MSE Loss)	64.88	10.18	20.75
Ensemble Filter (Dense - Custom Loss)	98.81	11.70	41.93

TABLE 2. Summary of results - MH-04

6. DISCUSSION

Three methods were employed and tested to get the ensemble estimate from the three filters. The methods employed were simple average, perceptron network and a dense neural network. Simple average worked well in scenarios where one filter would be under-estimating the states while the other filters would be over-estimating the states. Cases when all the filters are either under-estimating or over-estimating, the simple average would not perform well. Therefore, simple averaging was not good for generalization, but roughly followed the correct trend.

This result motivated us to build a perceptron network, one each for roll, pitch and yaw. A linear activation function was used in the perceptron network in order to get a weighted average along with an Adam optimizer. The perceptron was able to generalize better compared to the simple average and handle the situation better when all the filters are either under-estimating or over-estimating. The only place where it would fail to work is when the quaternion flips, which is the same as the aforementioned double-cover of angles (angle wrap-around). From the instance the network is faced with an angle wrap-around issue, the error cascades to all the estimates after that instant and results in a poor ensemble estimation of states.

In order to deal with the set-backs experienced with the simple average and the perceptron, a few different approaches were tried to deal with angle wrap-around. One of the approaches was to try a different loss function. The loss functions that we compared were the MSE loss function and the custom loss function in Equation 1. The purpose of the custom loss function has been discussed in section 4.2. Upon evaluation, it was observed that the MSE loss performed better than the custom loss function since the angle wrap-around has not been dealt with in the ground truth data and hence, it was not required of the loss function to deal with this. The network is required to learn to fit a line as closely as possible to the ground truth and hence the MSE loss function performed better at this task.

Another solution to deal with the setbacks was to create a dense neural network. The architecture of the dense network has also been discussed in section 4.2. Since this was a complex network compared to a simple perceptron, it was able to capture the nonlinearity of anglular data and therefore generalize better to then predict estimates with a lower margin of error. It was still unable to handle the angle wrap-around perfectly, but even after the angle wrap-around instance, the error would not cascade to the future estimates and hence would be able to predict estimates relatively well for the future timesteps. For datasets that experienced no angle wrap around, the performance of the dense neural network was extremely good as seen in Figure 2 and Tables 1 2.

In Tables 1 2, it can be seen that the perceptron and dense neural networks generalized best for roll and pitch. However, yaw experienced the most instances of angle wrap-around and therefore generalized poorly. On the other hand, the custom loss function designed to handle angle wrap-around generalized best for yaw, but struggled with roll and pitch. Using the MSE loss function for roll and pitch and the custom loss function for yaw is expected to yield the best results.

Since the main challenge for network generalization was angle wrap-around, it can be expected that modifying each filtering algorithm to preprocess the double cover (i.e. unwrap the angles) would yeild optimal results.

REFERENCES

[1] Cen, Ruping, et al. "A low-cost visual inertial odometry for mobile vehicle based on double stage Kalman filter." Signal Processing 197 (2022): 108537.

[2] Kraft, Edgar. "A quaternion-based unscented Kalman filter for orientation tracking." Proceedings of the sixth international conference of information fusion. Vol. 1. No. 1. IEEE Cairns, 2003.

[3] Bloesch, Michael, et al. "Robust visual inertial odometry using a direct EKF-based approach." 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2015.

[4] Sola, Joan. "Quaternion kinematics for the error-state Kalman filter." arXiv preprint arXiv:1711.02508 (2017).

[5] Valenti, Roberto G., Ivan Dryanovski, and Jizhong Xiao. "Keeping a good attitude: A quaternion-based orientation filter for IMUs and MARGs." Sensors 15.8 (2015): 19302-19330.

[6] Mourikis, Anastasios I., and Stergios I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation." ICRA. Vol. 2. 2007.

[7] Sun, Ke, et al. "Robust stereo visual inertial odometry for fast autonomous flight." IEEE Robotics and Automation Letters 3.2 (2018): 965-972.

[8] Burri, Michael, et al. "The EuRoC micro aerial vehicle datasets." The International Journal of Robotics Research 35.10 (2016): 1157-1163.

[9] Drolet, Louis, François Michaud, and Jean Côté. "Adaptable sensor fusion using multiple Kalman filters." Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113). Vol. 2. IEEE, 2000.

[10] Chaudhari, P. "ESE650 Course Lecture Notes" `https://pratikac.github.io/pub/21_ese650.pdf` (2021).

[11] Taylor,CJP. "MEAM620 Course Lecture Notes" `https://alliance.seas.upenn.edu/~meam620/wiki/` (2022).

[12] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[13] Haykin, Simon, and Richard Lippmann. "Neural networks, a comprehensive foundation." International journal of neural systems 5.4 (1994): 363-364.