ADAPTING BAYESIAN FILTERS FOR USE OF ANGULAR DATA

CANNON TUTTLE, CURTIS EVANS, SPENCER ASHTON, TYLER SANDERS

Abstract. This paper attempts to adapt linear filtering methods to work with angular data in order to predict angle of arrival of human voices. The data used for this project was both simulated and real data. We used various Bayesian filtering techniques on a continuous state space model to optimize angle estimation. We had success with a particle filter and a circular Kalman filter.

1. Problem Statement and Motivation

Imagine a construction site with heavy machinery, such as excavators, dump trucks, etc., in operation. The machine operator typically communicates with designated spotters via radio, but this leaves any other workers on the site unable to interface directly with the machine operators. A BYU acoustics project seeks to remedy this with a sound system that can pick up on the sound of human voice from outside the machine, filter out the noise, and reproduce the speech, including its relative direction to the machine, for the machine operator. This will augment the safety of those outside the machinery by increasing the awareness of the machine operators.

In order to replicate the directionality of the sound, the angle of arrival of the sound must be accurately estimated over time. Currently, this angle in relation to the center of the microphone array (see Figure 1) is calculated using pressure measurements and triangulation. According to the researchers, a sufficiently good angle approximation for this application is within $\pm 15^\circ$ of the true angle. However, the noisy environment makes raw triangulation insufficiently accurate. Somewhat rudimentary filtering has been implemented to improve stability, but so far the results of the calculation are still unreliable.

Our aim is to improve the robustness of the angle estimation in order to allow the researchers to more accurately reproduce the sound directionality. Previous work has shown that an unmodified Kalman Filter does not work well on angular data[1]. Various distributions, including a wrapped normal distribution and the Von Mises distribution have been successfully implemented to be used with Kalman type filters. However, these adaptations are complex and not easily implemented, as they include approximating the distributions with a wrapped Dirac mixture [1].

Date: April 10, 2024.

FIGURE 1. Microphone array used in data collection.

We modeled the angle of arrival evolution process as a continuous state space model and employed various Bayesian filtering techniques in order to optimize the angle estimation. This included adapting Kalman filters, normally used with data defined for all real numbers and gaussian distributed noise, to work with circular data that are defined on $[0, 2\pi]$ with the wrapping property at the boundary. We also adapted a Particle filter, a type of Bayesian filter that allows for the use of an arbitrary distribution, for use with circular probability distributions.

2. DATA

The data used for this project included both data given to us by the research team and simulated data. The original data is in the form of microphone pressure measurements that we then ran through a triangulation algorithm, provided to us by the research team, to produce angle measurements in time. Due to symmetry, each pair of microphones produces two possible angles. These can each be taken directly as possible measurements, producing 42 measurements per time step in our case, or all synthesized in order to produce a single angle measurement per time step. The provided triangulation algorithm produces data for both cases. However, the provided data didn't include certain extreme scenarios relevant to this application, including harsh discontinuities, frequent angle wraparounds, etc. To account for these scenarios, we produced representative synthetic data by plotting a

"true" angle, then adding gaussian noise to it to be used as the angle measurement. We also reserved real data to test our filter on after tuning our filter parameters.

3. Methods

We have tried several methods to compute the hidden angle measurement. Through this paper we use degrees in plots and examples, but all the measurements and computations were done in radians. We tried the following methods:

3.1. **State Space Model.** First we had to set up our continuous state space model. We needed angular velocity in our state space so to do that we do a finite difference approximation where $\theta'_t \approx \frac{\theta - \theta_{t-1}}{\Delta t}$. Now that we have angular velocity we can use it to make a simple forward Euler step in our state space. With the finite difference we need to save θ_{t-1} in the state space. This yields the following setup

$$\mathbf{x}_t = \begin{pmatrix} \theta_t \\ \theta_{t-1} \\ \theta' \end{pmatrix}, \; F = \begin{pmatrix} 1 & 0 & \Delta t \\ 1 & 0 & 0 \\ \frac{1}{\Delta t} & \frac{1}{\Delta t} & 0 \end{pmatrix}, \; H = \begin{pmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \end{pmatrix}$$

where H is of dimension $n \times 3$ for n observed angles. We don't have any control in our situation so our state space is

$$\mathbf{x}_t = F\mathbf{x}_{t-1} + \mathbf{w}_t,$$

$$\mathbf{z}_t = H\mathbf{x}_t + \mathbf{v}_t$$

.

3.2. **Kalman Filter.** We have two different approaches for Kalman Filters. Both originated from the Volume 3 Textbook [2] and Lab Manual [3]. For the rest of this paper we will call the Kalman filter illustrated in the textbooks as the naïve Kalman filter and the one we altered as the circular Kalman filter.

3.2.1. *Naïve Kalman Filter.* We first implemented an naive Kalman filter based on our state space model, specifically using the efficient form based on Kalman Gain. In the update step of this form, the updated state estimation is computed as follows, where K_t is Kalman Gain and the state is updated as:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t|t-1} + K_t(\mathbf{z}_t - H\hat{\mathbf{x}}_{t|t-1}).$$

The quantity $\mathbf{z}_t - H\hat{\mathbf{x}}_{t|t-1}$ is the difference between the observation $\mathbf{z}_t$ and expected observation given the predicted state $H\hat{\mathbf{x}}_{t|t-1}$.

The naïve Kalman filter is computing the arithmetic difference while the circular Kalman filter is computing a circular distance by taking into the fact that no two angles are more than 180° apart. This is illustraded with a couple of examples in the table below

Observed	Expected Observation	Arithmetic Diff	Circular Diff
15°	355°	-340°	20°
355°	15°	340°	-20°
120°	90°	30°	30°

3.2.2. *Circular Kalman Filter.* We adapted the Kalman filter for angular data by changing the update step to account for circular arithmetic. Specifically, we changed the calculation of the difference between true observation and predicted observation to ensure that the difference calculated was never greater than 180° (i.e. measuring the shortest distance around the circle), while preserving sign to account for their position in relation to each other. This computes the circular distance as illustrated in the table. Refer to Algorithm 1 to get an idea how this was implemented.

Algorithm 1 Altered Update Step

$\mathbf{v} \leftarrow \mathbf{z}_t - H\mathbf{x}_t$
if $|\mathbf{v}| \geq \pi$ **and** $\mathbf{v} \leq 0$ **then**
  $\mathbf{v} \leftarrow \mathbf{v} + 2\pi$
else if $|\mathbf{v}| \geq \pi$ **and** $\mathbf{v} \geq 0$ **then**
  $\mathbf{v} \leftarrow \mathbf{v} - 2\pi$
end if
$\tilde{\mathbf{y}}_t = \mathbf{v}$

3.3. **Particle Filter.** The second method we employed was another type of Bayesian filter, the particle filter [4]. The particle filter employs a large number of possible current states, each referred to as a "particle". Each particle has an associated weight, interpreted as the probability that particle accurately reflects the hidden state, with all weights summing to 1. As the source can possibly come in at any angle, we initialize the particles with uniformly distributed states and weights. At each time step, the filter propagates all of the particles forward in time according to the state space model, including adding state noise in the update. Then, using the data, it updates the weights of each particle by computing the posterior probabilities given data. Finally, it computes the state estimate as the weighted average of the states of all the particles.

In the update step, the filter computes $P(\theta|\text{data}) \propto P(\text{data}|\theta)P(\theta)$ for each particle. The prior $P(\theta)$ is given by the particle's previous weight, while $P(\text{data}|\theta)$ is calculated according to the probability distribution defined for the measurement noise. One of the benefits of the particle filter is that, unlike the normal distribution requirement of the Kalman filter, any arbitrary noise distribution may be employed. In this case, we used the Von Mises distribution (see Figure 2, right side). This closely approximates a Normal distribution that is "wrapped" around the angle boundary between 0 and 2π. Thus, no other consideration of angle wrapping must be made for

the particle filter. After the posterior weights have been calculated, they are again normalized by their sum so they represent a probability distribution.

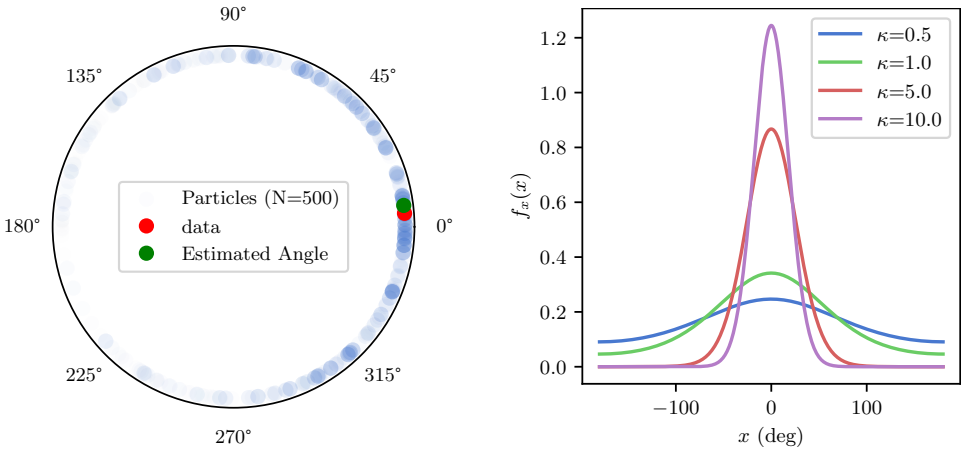

FIGURE 2. On the left, a particle filter example at a given timestep. The particles are colored according to weight, with the heaviest weights clustering around the data angle. On the right, the Von Mises distribution for various shape parameters (κ).

Particle filters suffer from what is called the "Degeneracy Problem", in which the weights become concentrated in a small number of the total particles. This is solved by resampling, a process in which low probability particles are replaced with higher probability particles. Detailing this problem and its solution is beyond our scope, but we used the `systematic_resample` function from `filterpy` to accomplish resampling.

As the particle filter keeps track of a large number of possible states which are distributed around the circle, it is much more flexible to nonlinear behavior. This is desirable in our case, as the source location may possibly reappear at any location in the state space. Unfortunately, the particle filter performs a similar number of computations for each individual particle as the Kalman filter performs in total. As we are using `num_particles`, our particle filter takes `n_particles` times as long to compute each step.

The Particle filter is defined with the following hyperparameters:

- `K1` defines the analogous variance term for the distribution of the state noise.
- `K2` defines the same for the measurements noise.
- `N_particles` defines the number of particles the model employs.
- `N_eff_particles` defines the minimum number of "effective" particles allowed before particles are resampled.

Similar to the Kalman filter, we tuned the hyperparameters `K1`, `K2`, `N_particles`, and `N_eff_particles` by performing a gridsearch over many

different possible values. A filter was constructed for each hyperparameter combination, with its performance tested across several different datasets. We then chose the hyperparameters that produced minimum average angular error over the different datasets. A similar method was implemented for a robotics research paper as well see [6].

4. Results and Analysis

Here we will first examine our success circular-izing Kalman filters (see Figure 3). Then, we will consider the speed of our filters, followed by their accuracy (see Figures 4 and 5). After that, we will review our unsuccessful attempt to combine many simultaneous angle observations into one computed angle (see Figure 6). Finally, we will show results of hyperparameter tuning our Particle filter (see Figure 7).

4.1. **Circular vs Naïve.** First, we compared the performance of the naive and circular Kalman filters in tracking a source moving continuously in a circle. The circular Kalman filter modification resulted in a marked improvement in behavior across the boundary between 360° and 0° as compared to the naive filter (see Figure 3). The Naive filter tended toward the arithmetic mean of 180° for walking in a circle, while the circular Kalman filter accurately tracked the source around the entire circle.

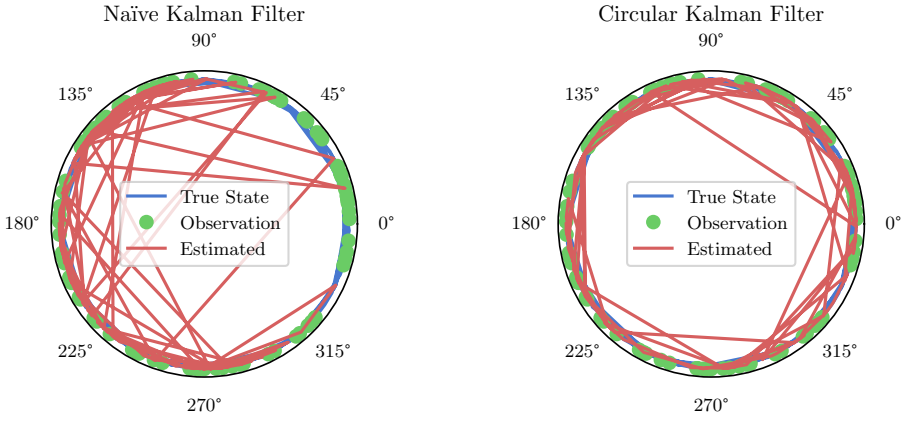

Figure 3. The naïve and circular Kalman Filter

4.2. **Speed.** The Kalman filters ran in about 3 to 5 miliseconds on about 1 second worth of data. The Particle filter took about 125 miliseconds on that dataset, so in total about 30 times longer than the Kalman filters. This means that all of our methods can be run in real time.

4.3. **Accuracy.** We used average absolute angle error as our primary evaluation metric. This took the absolute value of the circular distance between the true state and predicted state at each time step and averaged it across time for the whole dataset. The performance of each filter varied depending on the dataset.

4.3.1. *White Noise at Constant Angle.* On the White Noise dataset, the Particle filter was the best filter, with about half the error of the Kalman filters. The error for the Kalman filters was almost exactly the same (.367 for Noncircular and .369 for Circular), which makes sense because the wraparound algorithm in the Circular Kalman filter was not used, making it behave exactly like the unmodified Kalman filter.

4.3.2. *Constant Angle with Engine Noise.* The Particle filter performed the best again on this data set, however the improvement over the Kalman filters was not as large as it was with the White Noise data set. The error of the Particle filter was $\approx 25\%$ lower than the unmodified Kalman filter and was $\approx 40\%$ lower than the Circular Kalman filter. It is interesting to note that the unmodified Kalman filter performed significantly better than the Circular Kalman filter on this data set.

4.3.3. *Switching.* This is an artificial dataset created to simulate jumping between angles across $0°$. The Naïve Kalman filter performed poorly on this dataset. The Circular Kalman and Particle filters performed much better, with the Particle filter being twice as accurate (E = 0.199 vs E = .392) as the Circular Kalman filter.

4.3.4. *With Jump.* The Naïve Kalman filter performs better on this data set vs the Switching data set, but the results of the other two filters are the same.

4.3.5. *Wraparound.* The results of this dataset are the same as the previous dataset.

4.3.6. *The Best Filter.* The Circular Kalman and Particle filters both had consistent errors no matter the data set. The real trade off is whether we care more about speed or performance. The Particle filter was around twice as accurate, but it did take about 30 times longer to run. If computational efficiency is important, go with the Circular Kalman filter. If loacation accuracy is important, go with the Particle filter. The improvement in accuracy in the Particle filter is due to the lower variance compared with the Circular Kalman filter, as both responded to changes in angle measurements very quickly.

The researchers shared that due to limitations on reproducing angle-shifted sound accurately, that performance accurate to within approximately $15°$ of accuracy is desirable. Of our algorithms, only the Particle filter consistently was within $15°$ of accuracy.

4.4. **Filtering Multiple Angle Estimates.** Since our Kalman Particle observation matrix H allows for an arbitrary amount of simultaneous angles, we tried combining the pairwise microphone angle calculations simply by passing them all into our filters. See Figure 6. While there was some correlation, this model performed worse than the single angles. We realized that we have an unmet prerequisite to using Kalman Filter theory: our variation is not symmetric. However, if the microphone array were to be symmetric, this method would become viable again.

4.5. **Hyperparameter tuning of Most Accurate Filter.** The particle filter has several model parameters as mentioned previously. We performed a grid search to find the best filter for the engine dataset, see Figure 7.

5. Ethical Considerations

This project seeks to augment the safety of those outside the machinery by increasing the awareness of the machine operators. However, before deployment, this model must be shown to improve safety as much as or more than it increased the safety perception of the operators. Risk compensation "is a theory which suggests that people typically adjust their behavior in response to perceived levels of risk, becoming more careful where they sense greater risk and less careful if they feel more protected" [5]. This should be accounted for before deploying these methods in the field to ensure that the model performs robustly enough to truly enhance overall safety, despite risk compensation. Furthermore, it should be emphasized that operators should still use sight and radio communication to identify nearby individuals, and that people near machinery must still exercise caution.

6. Conclusion

Our goal was to improve robustness of the angle estimation by adapting filtering methods for use with circular data. We derived a very simple modification to the Kalman filter that allows it to work for circular data, and we implemented a particle filter using circular probability distributions. We found the particle filter and the circular Kalman filter suitable for angular data. The particle filter handles nonlinearities well but comes at a computational cost, while the circular Kalman filter is computationally efficient but suffers in accuracy in cases of strong nonlinearity. Due to the wraparound property of angular data, the unmodified Kalman filter is highly unsuitable for angular data. The particle filter and circular Kalman filter were both suitable, and can be adapted depending on the project's needs.

For the BYU Acoustics project specifically, if the computational cost of the filtering step is a concern, we would recommend use of the circular Kalman filter for this project due to its low cost and accurate performance except in cases of strong nonlinearity. If more computational headroom is allowed, the particle filter would be ideal, as it performed well even in cases of nonlinearities. Furthermore, the utility of these methods extends to any field

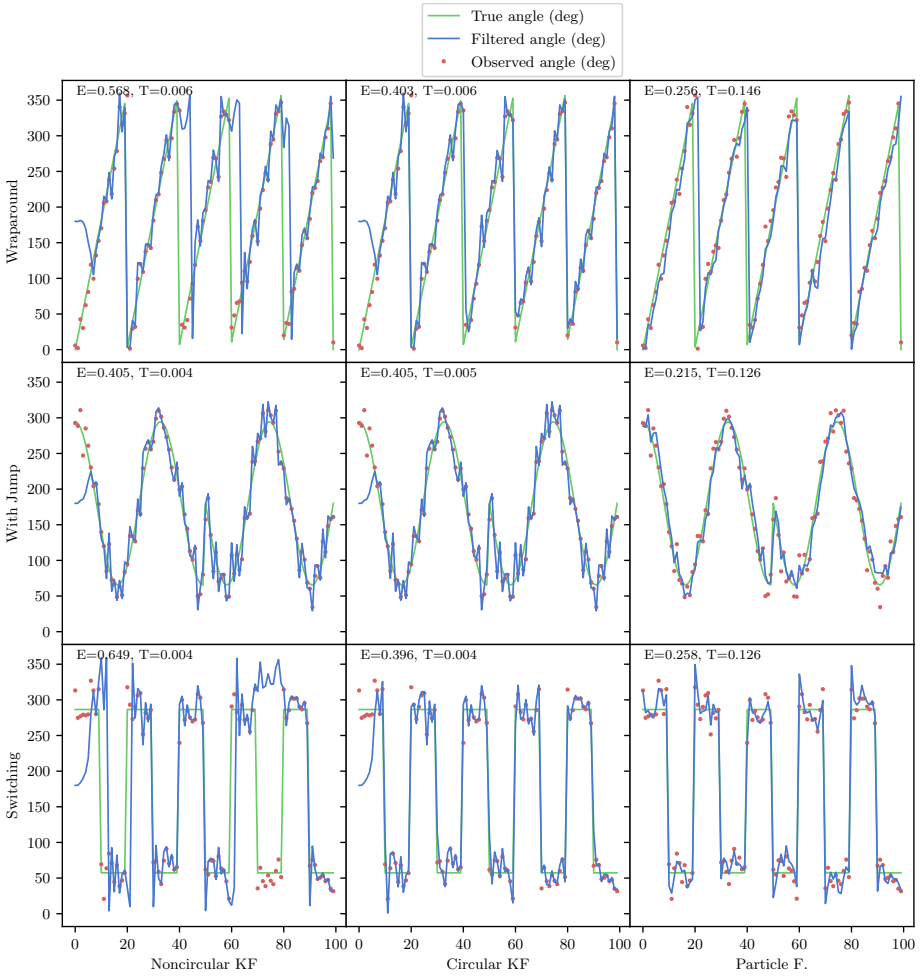

FIGURE 4. All filters, run on simulated datasets. Each row is a dataset, and each column is a type of filter. The red dots represent observations of the green true angle. The observations alone are filtered to produce the blue estimate angles.

in which angle measurements are taken, including robotics, navigation, and astronomy. The hyperparameters of our methods can easily be modified to adjust accuracy, flexibility, and computational cost depending on what is needed for the given scenario.

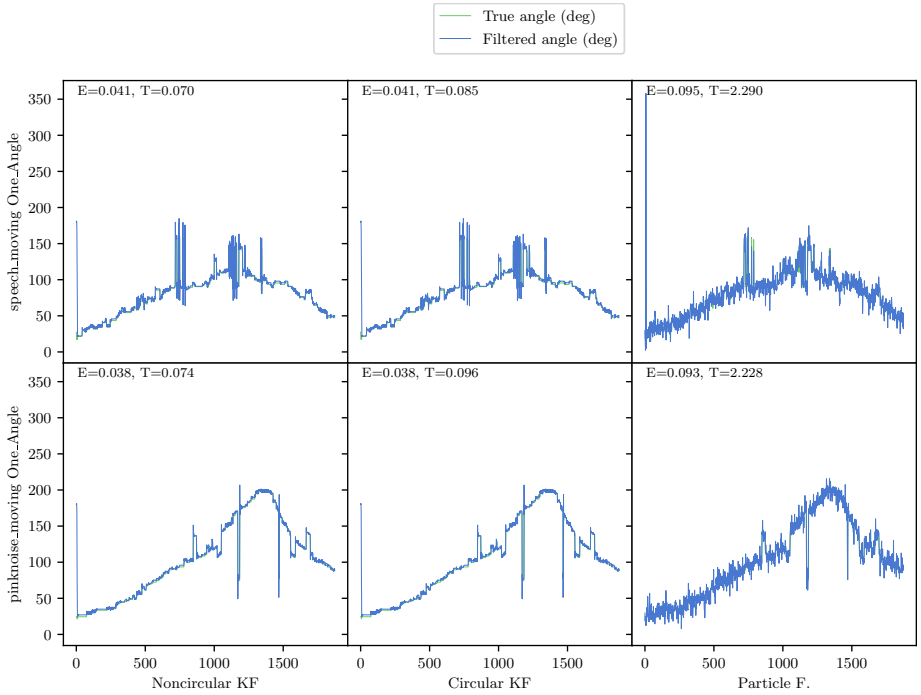

FIGURE 5. All filters run on angles calculated from real microphone-captured sound. Each row is a dataset, and each column is a type of filter. The true angle (green) is being filtered (blue).

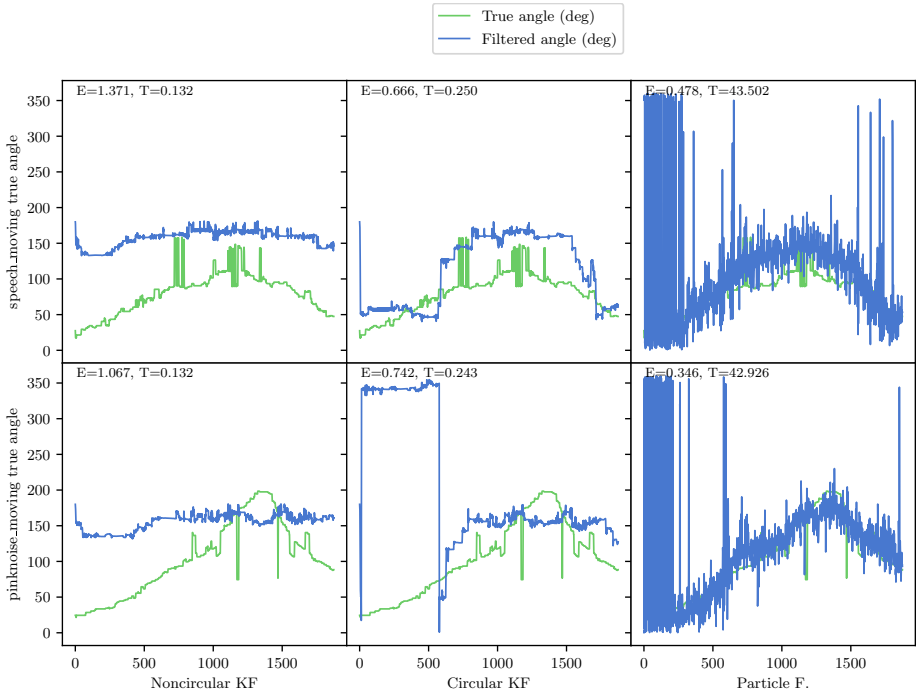

FIGURE 6. All filters, run on real data with multiple simultaneous angle measurements. Each row is a dataset, and each column is a type of filter. The true angle (green) is calculated separately, while the filtered angle (blue) actually comes from multiple individual angle measurements being fed through the single filter.

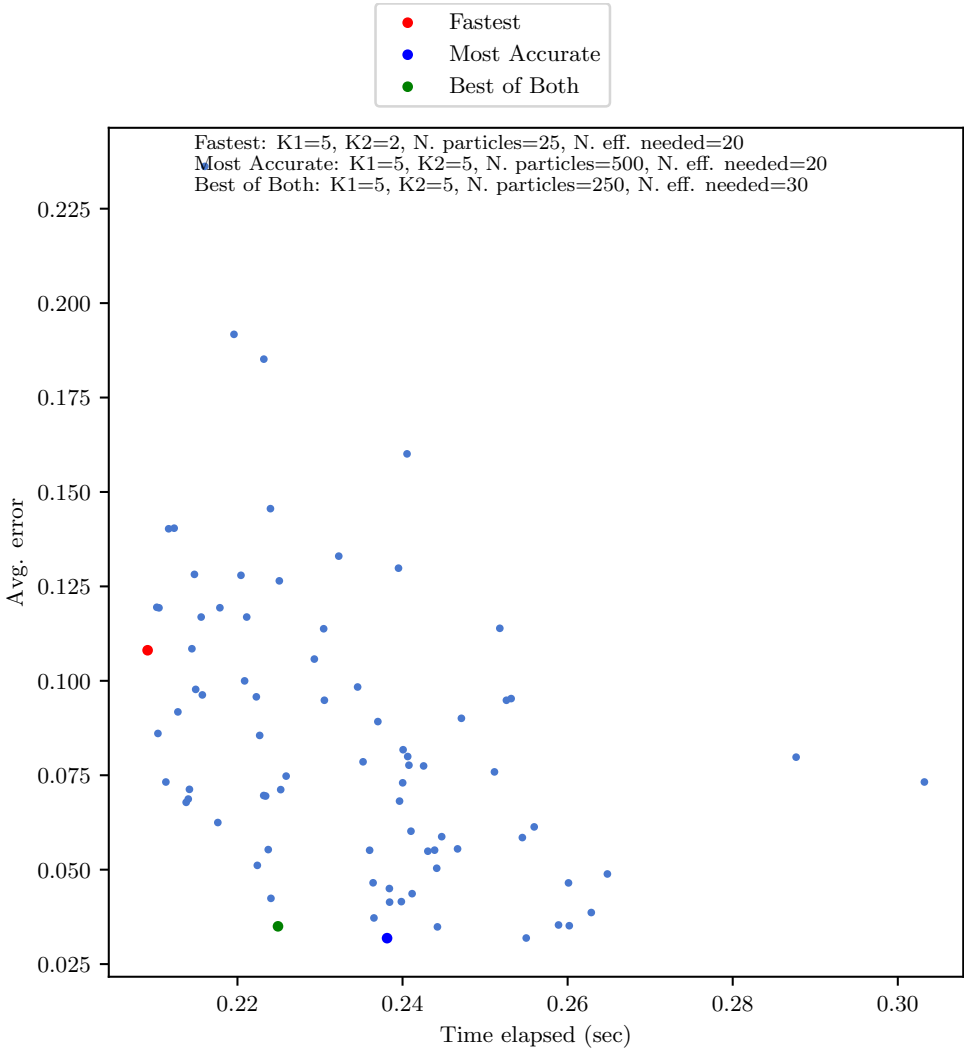

FIGURE 7. Hyperparameter tuning of the Particle filter.

References

[1] G. Kurz, I. Gilitschenski and U. D. Hanebeck, "Recursive nonlinear filtering for angular data based on circular distributions," *2013 American Control Conference*, Washington, DC, USA, 2013, pp. 5439-5445, doi: 10.1109/ACC.2013.6580688.

[2] The ACME Volume 3 Textbook

[3] J. Humpherys and T. Jarvis, "Labs for Foundations of Applied Mathematics, Volume 3, Modeling with Uncertainty and Data."

[4] Labbe, Roger. “Kalman and Bayesian Filters in Python”.

[5] Masson, Maxime; Lamoureux, Julie; de Guise, Elaine (October 2019). "Self-reported risk-taking and sensation-seeking behavior predict helmet wear amongst Canadian ski and snowboard instructors". *Canadian Journal of Behavioural Science.* 52 (2): 121–130. doi:10.1037/cbs0000153. S2CID 210359660.

[6] I. Marković and I. Petrović, "Speaker localization and tracking with a microphone array on a mobile robot using von Mises distribution and particle filtering," *Robotics and Autonomous Systems*, Volume 58, Issue 11, 2010, pp. 1185-1196, doi: 10.1016/j.robot.2010.08.001