ANGLE APPROXIMATION FROM PRESSURE MEASUREMENTS

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

A BYU Acoustics research project seeks to reproduce the filtered sound signal from a source near heavy machinery, within the cab for the operator. In essence, the machine operator should hear the reproduced voice of someone outside the cab speaking clearly from the direction they stand in relation to the machine. This will augment the safety of those outside the machinery by increasing the awareness of the machine operators. Our aim is to improve the robustness of the angle estimation in order to allow the researchers to more accurately reproduce the sound directionality.

Currently, the angle in relation to the center of the microphone array (see Figure 1) is calculated using triangulation. Somewhat rudimentary filtering has been implemented to improve stability, but so far the results of the calculation are still unreliable. We modeled the angle of arrival evolution process as a continuous state space hidden Markov model and employed various Bayesian filtering techniques in order to optimize the angle estimation. This included adapting filters normally used with data defined for all real numbers and gaussian distributed noise to circular data that are defined on $[0, 2\pi]$ with the wrapping property at the boundary.

Previous work has shown that an unmodified Kalman Filter does not work well on angular data, and our experience working with this data supports that conclusion. Various distributions, including a wrapped normal distribution and the Von Mises distribution have been adapted to be used with the Kalman Filter successfully. However, these adaptations are complex and not easily implemented, as they include approximating the distributions with a wrapped Dirac mixture [1]. Particle filters are a type of filtering algorithm that allow for the use of an arbitrary distribution and have been used with circular distributions before.

FIGURE 1. Microphone array used in data collection.

2. DATA

The data used for this project was both simulated data and real data given to us by the research team. The research team didn't provide all the data we needed to test our methods, so we decided to simulate our own data. The situations we needed to test were a stationary angle, moving angle, and discontinuous angle. The stationary angle would be someone standing still

talking to the machine operator, the moving angle would be someone walking around the machine while talking continuously, and the discontinuous angle would be someone walking around the machine while talking intermittently. The first situation we got data from the research team and the later two situations we needed to simulate. We are waiting for real data to test the last two with our methods.

3. Methods

We have tried several methods to compute the hidden angle measurement. We tried the following methods...

3.1. **Leaky Filter.** The leaky filter is a simple filter that just takes the last n samples and finds the average to use as the estimate of the true state. This is the filter currently being used on the project and the results have not been incredibly useful.

3.2. **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

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

.

3.3. **Kalman Filter.** We have two different approaches for Kalman Filters. Both originated from the Volume 3 Textbook [2] and Lab Manual [3].

3.3.1. *Naïve Kalman Filter.* To understand how angle measurements worked in the Kalman Filter we tried it with simulated data done in the lab manuals and class. We discovered a problem with the update step in angle wraparound from the angles in the range of

$$2\pi - \epsilon \leq \theta \leq 2\pi + \epsilon$$

for some $\epsilon \geq 0$. We noticed that the more noise we added the larged this ϵ got. See Figure 2 to get a visual.

We discovered that the problem was occurring in the update step where

$$\tilde{\mathbf{y}}_t = \mathbf{z}_t - H\hat{\mathbf{x}}_{t|t-1}.$$

A toy example would be if we are looking at an observed angle $\mathbf{z}_i = 15°$ and a predicted angle of $(H\mathbf{x})_i = 355°$ then $\mathbf{z}_i - (H\mathbf{x})_i = 15° - 355° = -340°$ and we desire $20°$. We could simply do this by noticing that $-340 \equiv 20$ (mod 360), however, if we said $\mathbf{z}_i = 355°$ and $(H\mathbf{x})_i = 15°$ then $\mathbf{z}_i - (H\mathbf{x})_i = 355° - 15° = 340°$ which isn't what we want. We want the Kalman filter to think of this difference as $-20°$.

3.3.2. *Circular Kalman Filter.* We solved this circular issue by taking wraparound into account. See Algorithm 1. This algorithm ensures that if a distance between two angles crosses the $0°$ threshold, both the short way round and the far way are considered, and the smaller is chosen. This resulted in an improvement across boundaries, see Figure 2.
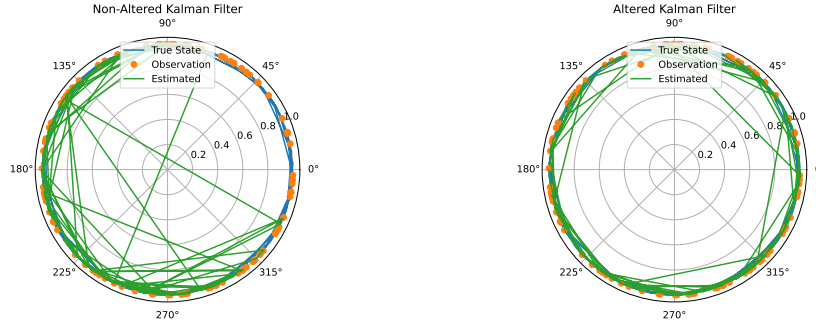

FIGURE 2. The non-altered and altered Kalman Filter

Algorithm 1 Process to Fix Wraparound

$\mathbf{v} \leftarrow \mathbf{z}_t - H\mathbf{x}_t$
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
$\tilde{\mathbf{y}}_t = \mathbf{v}$

3.4. **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 $P(\theta|\text{data}) \approx P(\text{data}|\theta)P(\theta)$ 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 3, 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.
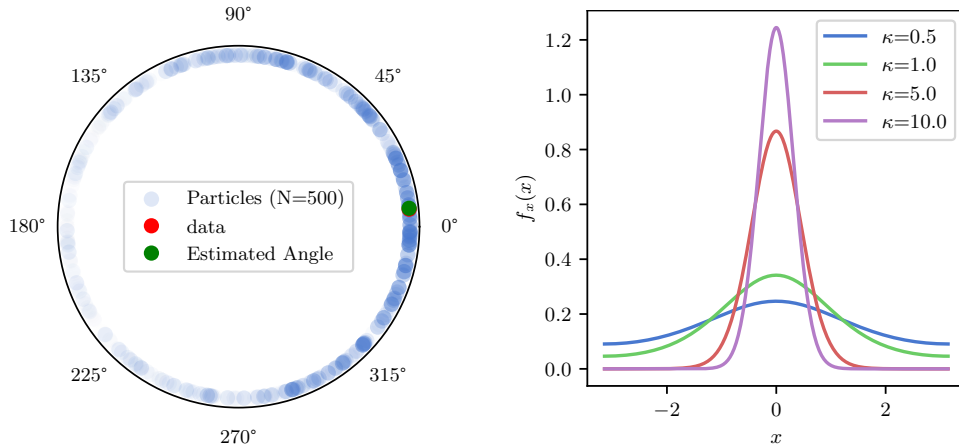

FIGURE 3. On the left, a particle filter example at a given timestep. The particles are colored according to weight, with the heaviest weights clustering around the data angle. On the right, the Von Mises distribution.

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, and
- `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 is a breakdown of the speed and accuracy of each filter (see Figure 4).

4.1. **Speed.** The Kalman filters rivaled the Leaky filter in speed. For instance, the average time for the Leaky filter to run was about 3 or 4 miliseconds, while the Kalman filters ran in about 4 or 5 miliseconds on about 2 miliseconds worth of data. The Particle filter took about 30 times longer than the Kalman filters.

4.2. **Accuracy.** In terms of accuracy the results varied per dataset.

4.2.1. *White Noise.* The Leaky filter was three times better on the White Noise dataset compared to the Particle filter, which was the second best filter. This is unsurprising since for a constant angle over time, the noise is normally distributed around the angle, which is a fixed mean. Thus, the MLE is a simple average, which is what the Leaky Filter is calculating.

4.2.2. *Real Engine.* Quantitatively, the Leaky filter again performed best, but the graphs tell a different story. A large part of this data set is at a fixed angle, but when it does change, the Leaky filter is the slowest to catch up. The other three filters have higher variances, but adjust much faster. The particle filter performed the most robustly, with about half the error of the Kalman filters.

4.2.3. *Switching.* This is an artificial dataset created to simulate jumping between angles across $0°$. Both the Leaky filter and Naïve Kalman filters performed poorly on this dataset. The Circular Kalman and Particle filters performed much better, with the Particle filter being twice as accurate (E = 0.453 vs E = .243).

4.2.4. *With Jump.* Here we see the true pitfall of the Leaky filter, not being able to track at all. The other filters all performed better, with the Particle filter again being the most accurate.

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

4.2.6. *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.

4.3. **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 5.

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.
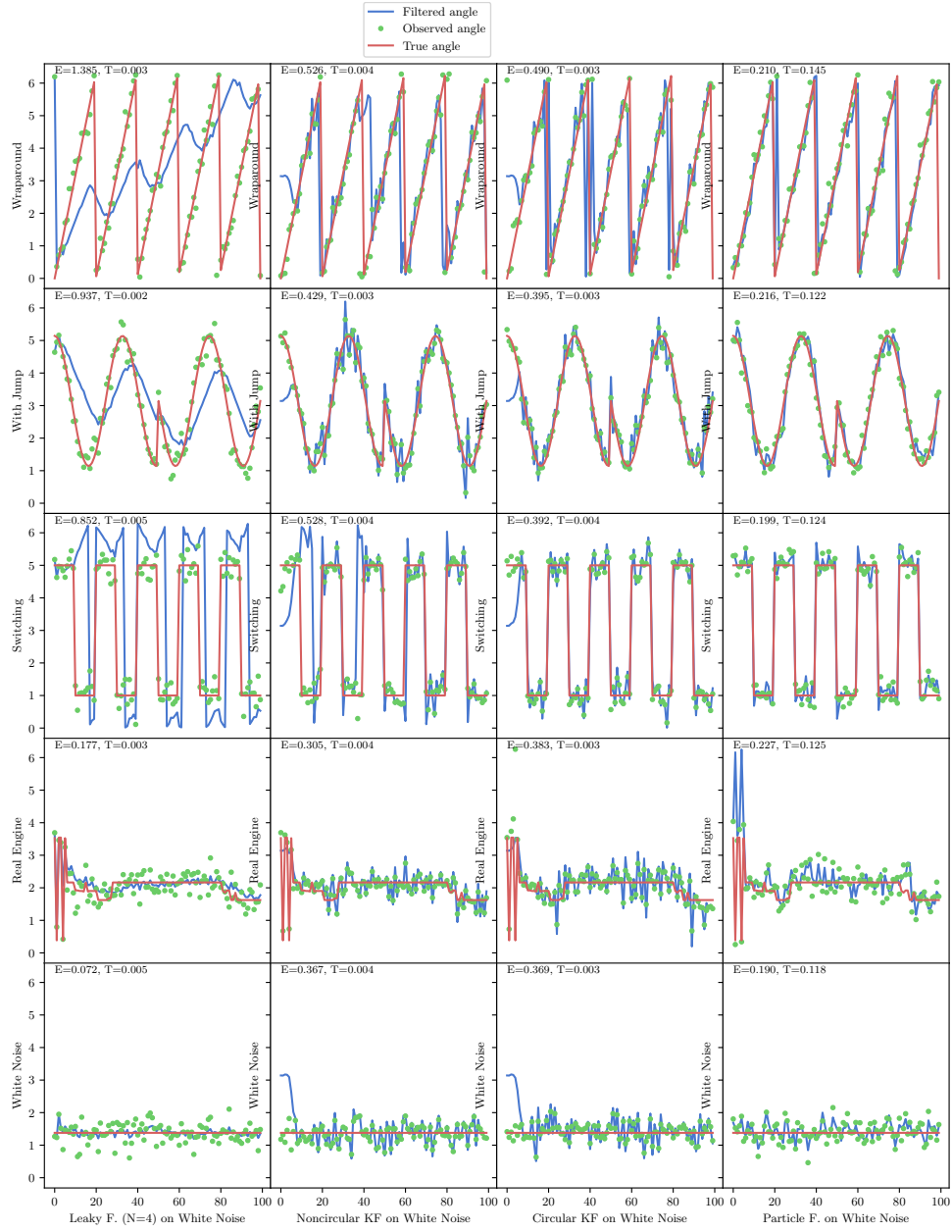
FIGURE 4. All filters, run on all datasets. Each row is a dataset, and each column is a type of filter. The green dots represent observations of the red true angle. The observations alone are filtered to produce the blue estimate angles.
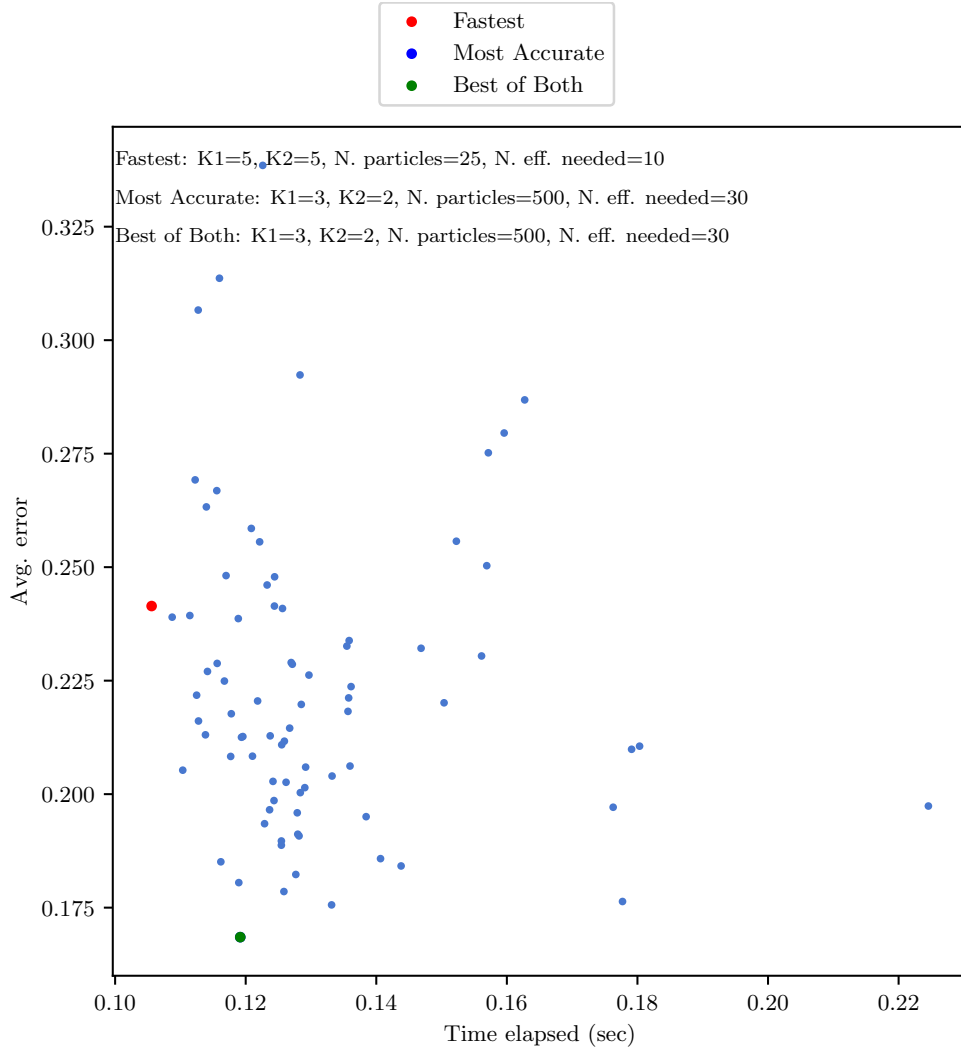
FIGURE 5. Hyperparameter tuning of the Particle filter.

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 leaky filter and unmodified Kalman filter are 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 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.

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