Deep Likelihood Learning for 2-D Orientation Estimation Using a Fourier Filter

Florian Pfaff, **Kailai Li**, and **Uwe D. Hanebeck**
Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology (KIT), Germany
florian.pfaff@kit.edu, kailai.li@kit.edu, uwe.hanebeck@kit.edu

***Abstract*—Filters for circular manifolds are well suited to estimate the orientation of 2-D objects over time. However, manually deriving measurement models for camera data is generally infeasible. Therefore, we propose loss terms that help train neural networks to output Fourier coefficients for a trigonometric polynomial. The square of the trigonometric polynomial then constitutes the likelihood function used in the filter. Particular focus is put on ensuring that rotational symmetries are properly considered in the likelihood. In an evaluation, we train a network with one of the loss terms on artificial data. The filter shows good estimation quality. While the uncertainty of the filter does not perfectly align with the actual errors, the expected and actual errors are clearly correlated.**

***Index Terms*—Deep learning, likelihood learning, trigonometric polynomial**

I. Introduction

Orientation estimation is a widespread estimation problem in biology and robotics, with applications including speaker tracking [1] and phase demodulation [2]. Tracking problems may involve image data, such as when estimating the orientation of a robot on the floor or the orientation of vehicles or planes on satellite images. While recursive Bayesian estimators can be used to solve this problem, likelihood functions are required, which are hard to provide for image data. Deep learning is well suited to deriving orientations from image data, but current approaches [3] generally lack the consideration of uncertainties.

For estimating the 2-D orientation of an object, we propose to extract likelihoods instead of only specific values from image data, as illustrated in Fig. 1. We also show that symmetries can result in ambiguities of the object's orientation. Based on the likelihoods, we are able to run a filter and thus provide both estimates and corresponding uncertainties. Furthermore, we can fuse information over multiple time steps even when the orientation of the object changes.

In this paper, the likelihoods are derived in a parametric form suitable for the update step of the Fourier square root filter [4] (SqFF). The prediction steps of the SqFF are executed according to a suitable system model. The SqFF is inherently suited to multimodal estimation problems, unlike the filters proposed in [1] and [5], which are based on wrapped normal or von Mises distributions [6, Sec. 2.2]. Allowing for multimodalities helps to handle ambiguities in the image data.

A state-of-the-art approach to provide uncertainties is to learn and output variances [7], which is not a promising option for circular domains due to the nonlinearity of the manifold. However, there are existing works considering other periodic domains. In [8], a loss based on the Bingham distribution is presented for providing uncertainties in the space of unit quaternions for 3-D orientation estimation. While the Bingham distribution allows for high probability mass along entire lines or rings on the unit hypersphere, it is not inherently capable of modeling densities with high probability mass in specific areas. An approach involving Bingham mixtures that can consider this case was proposed in [9]. In contrast to these approaches, we consider a family of densities that can be inherently multimodal without the requirement to involve any mixtures.

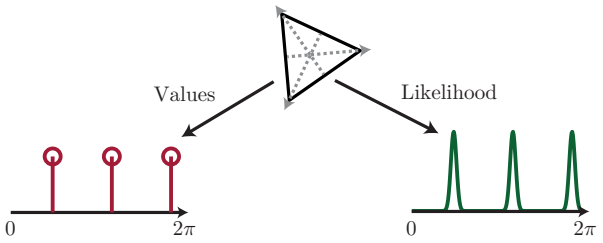


Fig. 1. Image of a triangle and possible orientations or likelihoods that may be derived from the image.

A work related to ours is [10], in which a neural network is used to process image data to provide inputs to a UKF. Instead of a UKF, we use the SqFF as a filter that is perfectly tailored to the considered domain. To provide likelihoods that are compatible with our filter, the network is designed to yield the coefficients of a trigonometric polynomial [11] (a Fourier series with a finite number of non-zero coefficients) describing the square root of the density.

The remainder of the paper is structured as follows. In Sec. II, we provide the basics about trigonometric polynomials, the SqFF, and parameterizations for trigonometric polynomials that are non-redundant. Knowledge of these topics is required for understanding the loss terms proposed in Sec. III. In Sec. IV, we explain how we trained a network based on one of the loss terms. We further describe a tracking scenario and evaluate the estimates and uncertainties provided by the filter. In the last section, we provide a conclusion and an outlook.

II. Filtering and Likelihoods Based on Trigonometric Polynomials

In the first subsection of this section, we provide the details on both real and complex trigonometric polynomials. This serves as the basis for the explanation of the relevant operations of the SqFF in the second subsection. In our networks, we use

non-redundant parameterizations of trigonometric polynomials. These are addressed along with the required operations for the loss term in the third subsection.

A. Basics of Trigonometric Polynomials

In our papers on the Fourier filters, such as [4], we only consider complex trigonometric polynomials. For the filters, the representation used is more of a technical implementation detail than an important aspect. For designing loss terms for training neural networks, we should consider everything that has an impact on the computational graph for the loss term.

A complex trigonometric polynomial is parameterized by a complex Fourier coefficient vector $\underline{c}$ and is given by [11, Volume I, Section I.1]

$$s^{\text{complex}}_{k_{\max}}(x) = \sum_{k=-k_{\max}}^{k_{\max}} c_k \mathrm{e}^{ikx} \ . \tag{1}$$

Complex trigonometric polynomials can be used to represent both real and complex functions. For real functions, a real trigonometric polynomial in the form of

$$s^{\text{real}}_{k_{\max}}(x) = \frac{1}{2}a_0 + \sum_{k=1}^{k_{\max}} \left(a_k \cos(kx) + b_k \sin(kx)\right) \ , \tag{2}$$

which is parameterized by two real Fourier coefficient vectors $\underline{a}$ and $\underline{b}$, can be used. $k_{\max}$ is referred to as the order of the trigonometric polynomial.

For a given function on $[0, 2\pi)$, the complex Fourier coefficients can be determined according to

$$c_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)\mathrm{e}^{-ikx}dx \ , \tag{3}$$

and the formulae

$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos(kx)\,dx \ , \quad b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(kx)\,dx$$

can be used to obtain the real Fourier coefficients [11, Volume I, Section I.4]. Since not all functions can be represented using a sum of sines and cosines, one generally only obtains an approximation of the original function when considering a trigonometric polynomial of specified order.

It is possible to convert the coefficient vector for a complex trigonometric polynomial to the coefficient vector for a real trigonometric polynomial. When generating the Fourier coefficient vectors according to

$$a_k = 2\mathcal{R}(c_k),\ b_k = -2\mathcal{I}(c_k) \ , \tag{4}$$

the real trigonometric polynomial is equivalent to the original complex trigonometric polynomial. Only the coefficients with positive indices are required because $c_k = \bar{c}_{-k}$ holds for real functions, with $\bar{c}$ denoting the complex conjugate of c. The conversion from real to complex coefficients is possible using

$$c_0 = \frac{a_0}{2} \ , \ c_k = \frac{a_k - ib_k}{2} \text{ for } k > 0 \ , \ c_k = \bar{c}_{-k} \text{ for } k < 0 \ .$$

B. Quick Overview of the Fourier Filters

In this subsection, we address the representations used in the Fourier filters and explain its update step. We also provide the motivation for using square roots in the Fourier square root filter and why it is sufficient for the network to output a scaled likelihood instead of the true likelihood. We do not address the prediction step as it is not the focus of this paper and refer the reader to [4], [12] for possible prediction steps.

1) Density Approximation: In the Fourier identity filter [4], densities are directly approximated using trigonometric polynomials. A parameterized density is given by

$$f^{\text{id}}(x; \underline{c}^{\text{id}}) = \sum_{k=-k_{\max}}^{k_{\max}} c_k^{\text{id}} \mathrm{e}^{ikx} \ . \tag{5}$$

While the Fourier identity filter has been shown to yield good estimates, it has the disadvantage that the trigonometric polynomial may have negative function values, which should not be allowed for probability densities. Negative function values also pose a problem to the loss terms considered in this paper. Thus, this representation is not considered any further.

In the SqFF, the nonnegativity of the density is ensured by approximating the square root of the density using a trigonometric polynomial, leading to the parametric family of densities

$$f^{\text{sqrt}}(x; \underline{c}^{\text{sqrt}}) = \left(\sum_{k=-k_{\max}}^{k_{\max}} c_k^{\text{sqrt}} \mathrm{e}^{ikx}\right)^2 \ . \tag{6}$$

Due to the squaring operation involved, the function values are guaranteed to be nonnegative. It is also possible to use these representations for likelihood functions when no normalization is enforced.

2) Update Step: In this paper, we assume we obtain a measurement $\hat{\mathbf{Z}}_t$ that is an image (or, with multiple color channels, a tensor). We also assume that the likelihood $f^{\boldsymbol{Z}|\boldsymbol{x}}(\hat{\mathbf{Z}}_t|x)$ is time invariant, i.e., the information about the system contained in the image only depends on the image data and not on the current time step.

Our update step is based on Bayes' rule. If we have a prior density $f^{\text{p}}_t(x_t|\hat{\mathbf{Z}}_1, \ldots, \hat{\mathbf{Z}}_{t-1})$ that incorporates only information of the measurements in previous time steps, we can obtain the posterior density that incorporates the information of the current measurement $\hat{\mathbf{Z}}_t$ according to

$$\begin{aligned} &f^{\text{e}}_t(x_t|\hat{\mathbf{Z}}_1, \ldots, \hat{\mathbf{Z}}_t) \\ &\quad = \frac{f^{\boldsymbol{Z}|\boldsymbol{x}}(\hat{\mathbf{Z}}_t|x_t) f^{\text{p}}_t(x_t|\hat{\mathbf{Z}}_1, \ldots, \hat{\mathbf{Z}}_{t-1})}{\int_0^{2\pi} f^{\boldsymbol{Z}|\boldsymbol{x}}(\hat{\mathbf{Z}}_t|x_t) f^{\text{p}}_t(x_t|\hat{\mathbf{Z}}_1, \ldots, \hat{\mathbf{Z}}_{t-1})\,dx_t} \\ &\quad \propto f^{\boldsymbol{Z}|\boldsymbol{x}}(\hat{\mathbf{Z}}_t|x_t) f^{\text{p}}_t(x_t|\hat{\mathbf{Z}}_1, \ldots, \hat{\mathbf{Z}}_{t-1}) \ , \end{aligned} \tag{7}$$

in which $\propto$ indicates that the term at the end only differs from the other terms by a non-zero constant. Because the posterior density is a valid density function and hence integrates to one, we can conclude from (7) that we can obtain the posterior density by multiplying the prior density by the likelihood and normalizing the result afterward. We obtain the same result when using the likelihood scaled by a positive factor instead of the actual likelihood. This will be useful later.

To perform the update step based on Fourier coefficient vectors, we use that the convolution of two complex Fourier coefficient vectors yields the Fourier coefficient vector for the multiplication of the two trigonometric polynomials. Because

$$\sqrt{f_t^{\mathrm{p}}(x_t) f^{\boldsymbol{Z}|\boldsymbol{x}}(\hat{\mathbf{Z}}_t|x_t)} = \sqrt{f_t^{\mathrm{p}}(x_t)}\sqrt{f^{\boldsymbol{Z}|\boldsymbol{x}}(\hat{\mathbf{Z}}_t|x_t)}$$

holds, we can directly use a discrete convolution (denoted by $*$) of the coefficient vectors for the prior density and likelihood to obtain the coefficient vector for the unnormalized posterior density before the normalization step according to

$$\underset{\circ}{\underline{c}}_t^{\mathrm{e,sqrt}} = \underline{c}_t^{\mathrm{p,sqrt}} * \underline{c}_t^{\mathrm{L,sqrt}} \ .$$

This vector is longer than the original vector since the convolution operation introduces additional coefficients. To prevent an increase in the required computation and memory over time, we truncate the coefficient vector $\underset{\circ}{\underline{c}}_t^{\mathrm{e,sqrt}}$ to the size of the vector $\underline{c}_t^{\mathrm{p,sqrt}}$ and call the result $\breve{\underline{c}}_t^{\mathrm{e,sqrt}}$. The vector $\breve{\underline{c}}_t^{\mathrm{e,sqrt}}$ still represents a potentially unnormalized density.

The normalization step can be provided based on the formula for the integral of a trigonometric polynomial. The integral of the square of a trigonometric polynomial with the coefficient vector $\breve{\underline{c}}_t^{\mathrm{e,sqrt}}$ over $[0, 2\pi)$ is $2\pi \left\|\breve{\underline{c}}_t^{\mathrm{e,sqrt}}\right\|^2$, which can be proven using the Parseval relation [13, Section 2.7.1]. When scaling the Fourier coefficient vector, the integral is squared by the square of the constant because we can pull the square of the constant out of the squared norm. Thus, by dividing the coefficient vector by the square root of the integral, we can ensure that the squared norm multiplied by 2π is 1. Therefore, we obtain the coefficient vector $\underline{c}_t^{\mathrm{e,sqrt}}$ representing a normalized density using

$$\underline{c}_t^{\mathrm{e,sqrt}} = \frac{1}{\sqrt{2\pi}\left\|\breve{\underline{c}}_t^{\mathrm{e,sqrt}}\right\|} \breve{\underline{c}}_t^{\mathrm{e,sqrt}} \ .$$

C. Non-Redundant Representations with Required Operations

We now consider parameterizations of trigonometric polynomials that involve no redundancies. Such vectors should be the output of our neural network. As we see later, we need to be able to integrate over likelihoods and normalize them. To emphasize that we consider potentially scaled, unnormalized likelihoods and not densities, we shall use $\breve{\mathcal{L}}$ instead of f from now on.

For a non-redundant parameterization of complex trigonometric polynomials, we consider only the part of the coefficient vector with the indices 0 to $k_{\max}$, which we write as $\underline{c}_{0:k_{\max}}^{\mathrm{sqrt}}$. Based on this vector, the parametric likelihood that only requires elements in $\underline{c}_{0:k_{\max}}^{\mathrm{sqrt}}$ but is otherwise equivalent to the square of the original trigonometric polynomial involving all coefficients is given by

$$\breve{\mathcal{L}}_{\mathrm{complex}}(x; \underline{c}_{0:k_{\max}}^{\mathrm{sqrt}}) = \left(\sum_{k=0}^{k_{\max}} c_k^{\mathrm{sqrt}} \mathrm{e}^{ikx} + \sum_{k=-k_{\max}}^{-1} \bar{c}_{-k}^{\mathrm{sqrt}} \mathrm{e}^{ikx}\right)^2. \quad (8)$$

To obtain the integral of the likelihood based on only the coefficients in $\underline{c}_{0:k_{\max}}^{\mathrm{sqrt}}$, we can use

$$2\pi \left\|\underline{c}^{\mathrm{sqrt}}\right\|^2 = 2\pi \left((c_0^{\mathrm{sqrt}})^2 + 2\left\|\underline{c}_{1:k_{\max}}^{\mathrm{sqrt}}\right\|^2\right) \ . \quad (9)$$

The normalization can be performed by dividing the vector $\underline{c}_{0:k_{\max}}^{\mathrm{sqrt}}$ by the square root of the integral.

Real trigonometric polynomials can also be used to avoid redundancies. The parametric likelihood is then

$$\breve{\mathcal{L}}_{\mathrm{real}}(x; \underline{a}^{\mathrm{sqrt}}, \underline{b}^{\mathrm{sqrt}}) = \left(\frac{1}{2}a_0^{\mathrm{sqrt}} + \sum_{k=1}^{k_{\max}} \left(a_k^{\mathrm{sqrt}} \cos(kx) + b_k^{\mathrm{sqrt}} \sin(kx)\right)\right)^2 \ . \quad (10)$$

The formula for the integral can be derived by combining the integration formula (9) with the rules for converting a complex Fourier coefficient vector to two real Fourier coefficient vectors (4). This results in the formula

$$\int_0^{2\pi} \breve{\mathcal{L}}_{\mathrm{real}}(x; \underline{a}^{\mathrm{sqrt}}, \underline{b}^{\mathrm{sqrt}})\, dx = \pi\left(\frac{(a_0^{\mathrm{sqrt}})^2}{2} + \left\|\underline{a}_{1:k_{\max}}^{\mathrm{sqrt}}\right\|^2 + \left\|\underline{b}^{\mathrm{sqrt}}\right\|^2\right). \quad (11)$$

From this formula, it is evident that if we multiply all entries in the vectors by a constant, we can pull out the square of that constant. Thus, we can normalize the likelihood by dividing all coefficients by the square root of the right-hand side of (11).

III. Loss Terms for Training and Validation

We assume there is a measurement function h mapping the state, in our case an angle, to an image. We assume that h is deterministic, which is an adequate assumption for simulations or when there is no measurement noise. For a fixed measurement $\hat{\mathbf{Z}}$, the likelihood function is then given by $f^{\boldsymbol{Z},\boldsymbol{x}}(\hat{\mathbf{Z}}|x) = \delta(\hat{\mathbf{Z}} - h(x))$, with δ being the Dirac measure. If h is invertible and the images are not ambiguous, we have a single Dirac in state space. If there are ambiguities due to symmetries, $f^{\boldsymbol{Z},\boldsymbol{x}}(\hat{\mathbf{Z}}|x)$ is a mixture of Diracs instead. While we will use the assumption that the true likelihood can be seen as a mixture of Diracs for one of our loss terms, this does not even hold true for simulation data. Due to its finite resolution and color information, the image cannot encode every real number, and thus, h is not invertible. Therefore, the potential true states can, in general, not be reconstructed with arbitrary precision based on a single measurement.

Due to the errors involved, the network should not simply output one or multiple angles. Providing these angles as certain inputs to our filter may lead to having probability mass only at these points, although the true value may not be any of these points. Hence, we want to derive an entire likelihood with nonnegative function values. By using a likelihood, we assume the error to be fully stochastic. While (especially in simulations) the error may be deterministic and caused by discretization only, assuming the error to be stochastic makes it possible to apply Bayesian filters. Furthermore, real scenarios generally involve stochastic noise in the form of sensor noise.

The network $\underline{n}_{\underline{\omega}}$ with parameters $\underline{\omega}$ that we train is supposed to map a measurement $\hat{\mathbf{Z}}$ to Fourier coefficients, i.e., $\underline{a}$ and $\underline{b}$ or $\underline{c}_{0:k_{\max}}$, that describe a continuous likelihood. The key hyperparameter of the network is the number of coefficients to provide, which determines the order of the trigonometric polynomial. A criterion for a good likelihood is that it should be high at all possible true angles (or angle, if there are no ambiguities) and low for angles that are far from the possible true angles.

First, we present a loss term that is based on maximizing the likelihood at the possible true angles. While this loss term did

not lead to fast convergence, it is a useful validation metric since it is not directly influenced by the order of the trigonometric polynomial and can thus be used to compare configurations with different hyperparameters. Second, we describe a loss term that is based on matching the (discrete) trigonometric moments of the mixture of Diracs at the possible true positions. The losses obtained for this loss term highly depend on the number of coefficients employed.

A. Likelihood-Based Loss Term

Our network $\underline{n}_{\underline{\omega}}$ converts the image $\hat{\mathbf{Z}}$ to Fourier coefficients (without redundancies) that describe the square root of the potentially scaled likelihood $\breve{\mathcal{L}}(x; \underline{n}_{\underline{\omega}}(\hat{\mathbf{Z}}))$. We start by addressing the case without ambiguities in the image data and denote the true state as $\tilde{x}$.

If we directly used $-\breve{\mathcal{L}}(\tilde{x}; \underline{n}_{\underline{\omega}}(\hat{\mathbf{Z}}))$ as the loss term, the network would have no incentive to adapt the coefficients so that the likelihood gets lower at any point. Thus, without any further adjustments, the parameter vector $\underline{\omega}$ may be changed by the training steps in ways that make the likelihood increase indefinitely.

For this reason, we derive the coefficients for the normalized likelihood, as explained in Sec. II-C. The application of the network followed by the normalization shall be denoted by $\underline{v}_{\underline{\omega}}$ and the normalized likelihood by $\mathcal{L}$. The normalization is not part of the network and is only applied for the calculation of the losses. For the filter, the likelihood (scaled by some positive constant) that is output by the network can be directly used by the SqFF since the result does not depend on scaling factors (see Sec. II-B2). Based on the normalized likelihood, we can define the loss term

$$L_{\text{asymm}}(\tilde{x}, \hat{\mathbf{Z}}, \underline{\omega}) = -\mathcal{L}(\tilde{x}; \underline{v}_{\underline{\omega}}(\hat{\mathbf{Z}}))$$

for cases without ambiguities.

For a batch of the training data, we calculate the loss as the average of the losses for the individual training samples. We recommend subdividing the training data into mini-batches to improve the training time. Furthermore, using mini-batches often improves the generalization capabilities of the trained network [14].

Now, we consider the case with ambiguities in which there are multiple possible true angles. We shall store these in a vector $\underline{\tilde{x}}$. We aim to have comparable losses across the samples in the training data. If the losses are not comparable, gradients can still be calculated. However, training samples that allow for greater improvements in the loss may be given more importance. Moreover, non-comparable losses would make it harder to keep track of the learning progress of the neural network.

To attain comparable losses, we take into account that we normalize the likelihood to one. Due to the smoothness of trigonometric polynomials, the integral over it is influenced by each mode. We now assume the area under the curve caused by any mode is proportional to the value at the peak. Under this assumption, the heights of the peaks are not independent of the number of modes since the normalized likelihood integrates to one. Rather, for m peaks at the m true states, each peak can only be $1/m$ of the height of the peak for a single true state. Under our assumptions

$$L_{\text{sum}}(\underline{\tilde{x}}, \hat{\mathbf{Z}}, \underline{\omega}) = -\sum_{j=1}^{m} \mathcal{L}(\tilde{x}_j; \underline{v}_{\underline{\omega}}(\hat{\mathbf{Z}})) \tag{12}$$

yields comparable losses regardless of the number of possible true states.

One issue with the loss term (12) is that the likelihood is not inherently enforced to be multimodal when there are multiple possible true angles. For example, for two possible angles that are opposite to each other, the network could always provide a likelihood that has a single mode in $[0, \pi)$ and never consider the other possible true state in $[\pi, 2\pi)$ without any major disadvantage in the obtained losses. This can harm the quality of the filter results.

Consider a tracking scenario in which the angle slowly increases toward π and at some point passes π. If the likelihood only focuses on the mode in $[0, \pi)$, the posterior density in the SqFF may have high density only close to π briefly before the value of π is exceeded. Once π is exceeded, the likelihood will be high at 0 and low at π. Thus, the prior density and the likelihood in the update step will be contradictory, which may lead to high uncertainties. In the worst-case scenario, the prior density and likelihood are identical up to a shift by π. In this case, a circular uniform distribution is obtained as the result of the update step. Conversely, if the likelihood is high at both possible true angles, the posterior density will be high at the true state even after it exceeds π, and thus, good tracking results can be attained.

With this in mind, we define a loss term that ensures high likelihood at all possible true angles. For this, we take the mth root of the likelihood and calculate the product of the likelihoods for all possible true angles. To account for the lower function values when there are multiple possible true angles, we multiply the result by m to obtain the loss term

$$L_{\text{prod}}(\underline{\tilde{x}}, \hat{\mathbf{Z}}, \underline{\omega}) = -m \prod_{j=1}^{m} \sqrt[m]{\mathcal{L}(\tilde{x}_j; \underline{v}_{\underline{\omega}}(\hat{\mathbf{Z}}))} \,. \tag{13}$$

A related loss term that has its minimum at the same value can be obtained by applying a logarithm to the part after the minus sign.

Without ambiguities, this loss term introduces little assumptions about the shape of the likelihood and is thus very versatile. However, with ambiguities, comparable losses are only achieved if the modes are spaced sufficiently far apart. Consider an example with very few Fourier coefficients. With an increasing number of possible true angles, the trigonometric polynomial will become unable to have low function values between the peaks and will start to resemble a circular uniform distribution. When further increasing the number of modes, the product on the right-hand side of (13) stays approximately constant while the factor m still increases. Thus, the losses may become multiple times as large as those for inputs without ambiguities.

However, if a 2-D shape is not circularly symmetric (which is a case that would need to be handled differently), the rotational symmetry must be an m-fold rotational symmetry. This implies that all possible true angles are $2\pi/m$ apart. Hence, if m is not too high and a sufficiently high number of coefficients is used so that the function value can go down between two

modes, the loss function (13) is usable. New challenges may arise for 3-D objects as rotational symmetries along one axis are widespread, which would require special treatment in the loss term.

If problems arise concerning non-comparable losses for different symmetries, one also has the option to generate mini-batches with a fixed mixing ratio of shapes with different symmetries. For this, a sufficiently large mini-batch size is required, and it should be possible to provide mini-batches with the chosen ratio without using some training samples much more often than others. One should also keep in mind that symmetries with higher variability in the losses may be given more importance.

Using L_{prod} may lead to large computation graphs through which gradients have to flow because it involves evaluating the trigonometric polynomial at all possible true angles. The loss term introduced in the next subsection leads to smaller computation graphs and also better convergence.

B. Hellinger Distance-Based Loss Term

The key idea for this loss term is to interpret the possible true angles $\underline{\tilde{x}}$ as a sample set and match moments of that sample set with a continuous density. This loss term can also be used if the shape is fully circularly symmetric. Using the formula

$$r_k^{\text{s}} = \frac{1}{m}\sum_{j=1}^{m} \mathrm{e}^{ik\tilde{x}_j} \ , \tag{14}$$

we obtain the kth trigonometric moment of the sample set [6, Section 1.3]. We now aim to provide a continuous density based on a trigonometric polynomial that matches these moments. This is straightforward since the trigonometric moment of a continuous density is defined as [6, Section 2.1]

$$r_k^{\text{c}} = \int_0^{2\pi} f(x)\mathrm{e}^{ikx}dx \ .$$

Thus, we can obtain the desired density using $c_k = \frac{1}{2\pi}\overline{r}_k^{\text{s}}$ for the complex Fourier coefficient vector or $a_k = \mathcal{R}(r_k^{\text{s}})/\pi$ and $b_k = \mathcal{I}(r_k^{\text{s}})/\pi$ for the real coefficient vectors. We can interpret the $1/m$ in (14) as the weights of the samples.

To obtain Fourier coefficients for the square root of the likelihood, we could take the square roots of the weights. However, since all samples are equally weighted, this would merely lead to scaling the moments by a fixed constant. As discussed previously, scaling the coefficient vector with a constant merely results in scaling the likelihood that is obtained by squaring the trigonometric polynomial. Since we normalize the likelihood for calculating the losses, we do not have to consider this. The derived Fourier coefficients can thus be directly used to describe a trigonometric polynomial for the square root of the likelihood.

The number of trigonometric polynomials we calculate based on $\underline{\tilde{x}}$ is always identical to the desired number of Fourier coefficients. In Fig. 2, we show the squared trigonometric polynomials for different numbers of coefficients. The true states are illustrated using weighted samples. As we can see, the likelihood is high at the possible true angles and is low in the middle between them, which is the desired outcome.

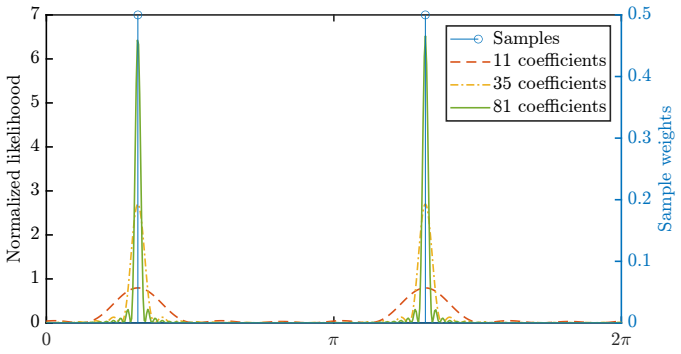


Fig. 2. A weighted sample set at the possible true angles is shown along with squared trigonometric polynomials for different numbers of coefficients. The coefficients are determined according to the trigonometric moments of the sample set.

To formalize the loss term, we now define a function g that extracts the trigonometric moments $r_0^{\text{s}}, \ldots, r_{k_{\max}}^{\text{s}}$ from the set of possible true states $\underline{\tilde{x}}$, converts them into the coefficients for the trigonometric polynomial, and scales them so that the squared trigonometric polynomial (which describes the likelihood) is normalized. Using a mean squared error loss between $\underline{v}_{\underline{\omega}}(\hat{\mathbf{Z}})$ and $g(\underline{\tilde{x}})$ as our loss term would neglect the semantic behind the coefficients. We want our resulting normalized function $\mathcal{L}(x; \underline{v}_{\underline{\omega}}(\hat{\mathbf{Z}}))$ to be close to the function $\mathcal{L}(x; g(\underline{\tilde{x}}))$. Since the two functions are normalized, we can use the Hellinger distance [15], which is a measure of distance between two density functions. The Hellinger distance involves a square root and a factor $1/2$, which we can ignore because they do not change where the distance has its minimum. Thus, we obtain

$$L_{\text{hel}}(\underline{\tilde{x}}, \hat{\mathbf{Z}}, \underline{\omega}) = \int_0^{2\pi} \left(\sqrt{\mathcal{L}(x; g(\underline{\tilde{x}}))} - \sqrt{\mathcal{L}(x; \underline{v}_{\underline{\omega}}(\hat{\mathbf{Z}}))} \right)^2 dx \tag{15}$$

as our loss term. The square in the integral must not be omitted to avoid that positive and negative parts cancel out.

The loss term can be calculated efficiently because $g(\underline{\tilde{x}})$ comprises the Fourier coefficients of $\sqrt{\mathcal{L}(x; g(\underline{\tilde{x}}))}$ and $\underline{v}_{\underline{\omega}}(\hat{\mathbf{Z}})$ of $\sqrt{\mathcal{L}(x; \underline{v}_{\underline{\omega}}(\hat{\mathbf{Z}}))}$. Subtracting one trigonometric polynomial from another is, as evident from the formulae (1) and (2), possible by subtracting the coefficients of the second from those of the first. This can be used to directly obtain the Fourier coefficients of $\sqrt{\mathcal{L}(x; g(\underline{\tilde{x}}))} - \sqrt{\mathcal{L}(x; \underline{v}_{\underline{\omega}}(\hat{\mathbf{Z}}))}$. The integral of the square of the trigonometric polynomial with the resulting parameters over $[0, 2\pi)$ can be determined according to the formulae (9) and (11) for complex and real trigonometric polynomials, respectively. When considering a mini-batch, we calculate the average of the losses as in the likelihood-based loss term.

The gradient has to flow from the loss to $\sqrt{\mathcal{L}(x; \underline{v}_{\underline{\omega}}(\hat{\mathbf{Z}}))}$ and then through the normalization step to the main part of the network. No gradient is needed for the likelihood $\mathcal{L}(x; g(\underline{\tilde{x}}))$ since this term is independent of the parameters of the network (it only depends on the hyperparameters). In the experiments in our evaluation, good convergence was obtained for the loss term (15). We further observed that reducing this loss also

(a) Arrow. (b) Semicircle. (c) Double arrow. (d) Triangle. (e) Pentagon.

Fig. 3. Classes used in the evaluation.

resulted in a reduction of the likelihood-based loss (13). A disadvantage of the loss term (15) is that the losses are not comparable when the hyperparameter describing the order of the trigonometric polynomial is changed.

IV. Evaluation

As a proof of concept, we considered a simple 2-D orientation estimation scenario[1]. We used an artificial data set and a simple network structure, which are explained in the first subsection. The tracking scenario used to evaluate the quality of the filter results is described in Sec. IV-B. In Sec. IV-C, we provide the evaluation criteria and results.

A. Training Data and Network

For the evaluation, we generated artificial images of 5 different shapes with a resolution of 24×24, which are illustrated in Fig. 3. While the arrow and the semicircle only have a single possible true orientation, the double arrow has two, the triangle has three, and the pentagon has five.

In Fig. 3, the images are shown in their base orientation, which we define as 0. For each of the classes, 1000 images are generated by rotating the base image by a random angle in $[0, 2\pi)$ using the `imrotate` function of Matlab. It was ensured that the background pixels are always white. A limitation of this data generation scheme is that since all generated images are based on the image with an orientation of 0, the image may be sharper for that orientation. However, we believe this effect is negligible. As all images are only black and white, only a single color channel was used. The images were not binarized, and thus, the intensity values can also contain information about the orientation of the shape. Thus, subpixel precision can be achieved. However, even considering this, the image resolution still limits the achievable precision.

In the evaluation scenario, which we describe in the next subsection, all data were generated during run time. Thus, it is guaranteed that none of the data used in the evaluation runs are contained in the training data. 75% of the 5000 images were used as training data, the remaining 25% as validation data. Since the training data comprises approximately 750 images per class, it is likely that similar images to those encountered in the evaluation runs have been observed by the network in the training phase.

In our processing pipeline, we standardized the images by calculating the mean and standard deviation over the whole training data, subtracting the mean from the images, and dividing the result by the standard deviation. In our proof of concept implementation, we used a modified LeNet-5 [16] as the basis for our network. The layer at the end that brings down the dimensionality from 84 to 10 was replaced with a fully connected layer with an output size that corresponds to the desired number of coefficients. No softmax was used.

As the training loss, the Hellinger distance-based loss, as introduced in Sec. III-B, was used. This loss and the likelihood-based loss were used as validation metrics. A possible strategy to choose the number of coefficients is to start with a relatively low number, which is increased until the likelihood-based loss term stops decreasing for the validation data. If ground truth data are available for a high number of runs of the considered tracking scenario, one may consider stopping earlier if the tracking results stop improving. In our evaluation, we use 81 coefficients. While using 101 coefficients led to a lower likelihood-based loss, the tracking results did not change significantly. Using 131 coefficients, even the likelihood-based loss did not improve further.

A single network was trained for all classes using PyTorch 1.9. A fixed learning rate of 0.0005 was used. The training was stopped after 30 epochs. The training took 7 minutes and 47 seconds on a server with an Intel Core i9-10980XE CPU, 128 GB of RAM, and four Nvidia GeForce RTX 2080 Ti graphics cards.

B. Evaluation Scenario

The scenario was simulated for ten time steps in each run. During the run, the shape was not changed. The initial state $\boldsymbol{x}_1$ was distributed according to a uniform distribution on the circle. Thus, no specific orientation was given any advantage. As the system model, we used a random walk adapted to the periodic manifold. The system state at time step $t+1$ was thus generated based on the state of the system $\boldsymbol{x}_t$ and the system noise $\boldsymbol{w}_t$ according to

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{w}_t \mod 2\pi \ .$$

The system noise was i.i.d. over time and von Mises distributed with parameters $\mu = 0$ and $\kappa = 10$. The measurements were generated by rotating the image according to the state.We used an SqFF with the same number of coefficients as the likelihood (81). The filter was initialized with a uniform distribution on the circle. We used the prediction step for the topology-aware identity system model with additive noise, as explained in [4]. The unnormalized likelihood $\breve{\mathcal{L}}(x; \underline{n}_{\hat{\underline{\omega}}}(\hat{\mathbf{Z}}))$ based on the trained network parameters $\hat{\underline{\omega}}$ was used as the likelihood for the update step (see Sec. II-B2).

C. Evaluation Results

In our evaluation of the estimation quality, we only considered the filter result at the last time step. We did not consider all time steps since the estimation qualities would have been correlated. Using only the result at the tenth time step also helps to reduce the effect of the initial prior density on the estimation quality of the filter. In experiments, the estimation quality did not change significantly after ten time steps.

As the basis for our assessment of the quality of the estimates derived from the filter, we used the angular distance

$$d(x, \tilde{x}) = \min(|x - \tilde{x}|, 2\pi - |x - \tilde{x}|) \ .$$

We adjusted this metric to also consider the symmetries of the different shapes. While the symmetries may not be known

[1] A pretrained network and all code are available at https://github.com/KIT-ISAS/FUSION21_2DLikelihoodFromImg.

in practice, considering them in the evaluation allows for a thorough assessment of the estimation quality. Based on one possible true orientation $\tilde{x}$, the other possible true orientations can be described by $\tilde{x}+\frac{2\pi j}{m} \mod 2\pi$ with $j \in \{0,\ldots,m-1\}$. We consider a distance that is the minimal distance to any of the possible true states. The formula is given by

$$d_{\text{symm}}(x,\tilde{x}) = \min_{j\in\{0,\ldots,m-1\}} \Bigg\{ \min\left(\left| x - \left(\tilde{x} + \frac{2\pi j}{m}\right)\right|, 2\pi - \left| x - \left(\tilde{x} + \frac{2\pi j}{m}\right)\right| \right) \Bigg\}. \tag{16}$$

It would have also been possible to add the offsets to x instead of $\tilde{x}$ without changing the resulting values. Note that this distance gives shapes with ambiguities an advantage. If we have no knowledge about the state and draw x from a uniform distribution on the circle, the expected distance will be $\pi/2$ without ambiguities and $\pi/(2m)$ with ambiguities.

When the rotational symmetry of the object is unknown, one can, e.g., use the maximum of the posterior density as an estimate. In our evaluation, we know the number of true orientations. Thus, we determined the value that minimized the expected error with respect to our distance measure (16). In the formula for the expected distance, the potential estimate $\bar{x}$ is fixed. The distances of all possibles states to $\bar{x}$ are weighted with the probability that the state is the true state according to the posterior density provided by the filter, and the weighted distances are integrated over $[0, 2\pi)$. Denoting the random variable at time step t with the posterior density $f_t^{\text{e}}(\cdot)$ as $\boldsymbol{x}_t^{\text{e}}$, we obtain the estimate according to

$$\begin{aligned}\hat{x}_t &= \arg\min_{\bar{x}\in\,[0,\,2\pi)} \mathrm{E}(d_{\text{symm}}(\bar{x}, \boldsymbol{x}_t^{\text{e}})) \\ &= \arg\min_{\bar{x}\in\,[0,\,2\pi)} \int_0^{2\pi} d_{\text{symm}}(\bar{x}, x_t) f_t^{\text{e}}(x_t|\hat{\mathbf{Z}}_1,\ldots,\hat{\mathbf{Z}}_t) dx_t\,.\end{aligned}$$

We will now describe the actual estimation errors and then compare the expected estimation errors with the actual errors.

1) Estimation Error: In Fig. 4, we show the results of 1000 runs for all the considered shapes using box plots. The red line in each box is at the median, and the box ranges from the 25% to the 75% quantile. The size of each box corresponds to the interquartile range. The whiskers extend to the value that is at most 1.5 times the interquartile range away from the median. The red crosses are outliers. Almost all outliers are shown in the plot, and at most three were omitted per shape to allow for better axis limits.

In almost all runs, the error is less than 0.05 radian (less than 3 degrees) for all the shapes. All the whiskers end below 0.026 radian, and thus, more than 75% of the errors are less than 1.5 degrees. The accuracy is not much higher for shapes with symmetries, which we believe is a result of the very high accuracy of the estimates.

2) Uncertainty Assessment: A key advantage to using a recursive Bayesian estimator with likelihoods generated by the network is that we can provide a whole density for the orientation. Now, we compare the expected error $\mathrm{E}(d_{\text{symm}}(\hat{x}_{10}, \boldsymbol{x}_{10}^{\text{e}}))$, i.e., the error we would expect on average without any knowledge of the true value (only knowledge about the symmetries is contained in the distance function) with the actual error $d_{\text{symm}}(\hat{x}_{10}, \tilde{x}_{10})$ at the 10th time step. The latter was computed using one of the possible true orientations, which is denoted by $\tilde{x}_t$. The other possible true orientations were also considered due to the definition of the distance function (16).

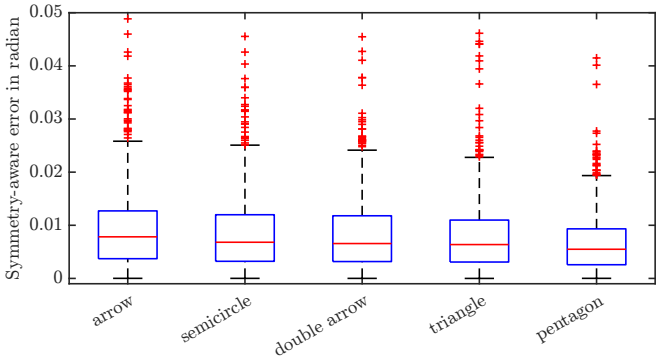


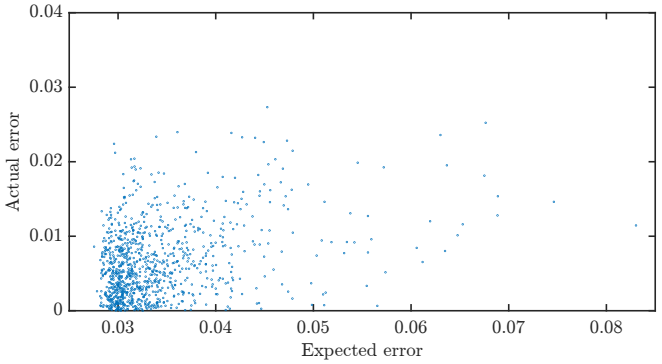
Fig. 4. Box plots showing the errors. At most three outliers are omitted for each of the shapes.

The individual expected and actual errors of all 1000 runs for the pentagon shape are depicted as tiny circles in Fig. 5a. A few circles were omitted to improve the visualization of those that are shown. Due to the uncertainty from the prediction step and the uncertainty in the likelihood, the expected deviation was never below 0.0288. Much higher expected errors were only observed rarely.

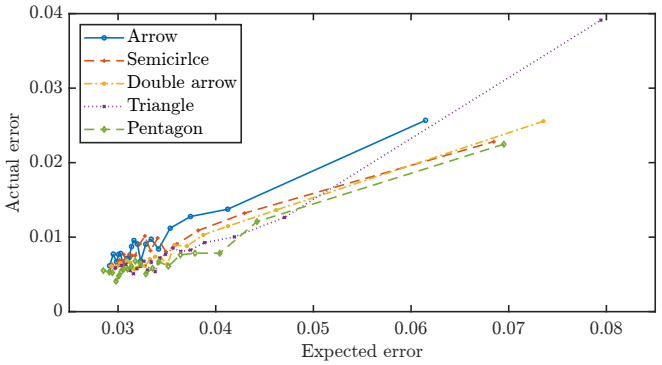
For a better visual assessment, we subdivided the data points along the axis for the expected deviation into 20 bins with 50 points each. We calculated the averages of the expected and actual errors in each individual bin. We show these for all shapes in Fig. 5b. The expected and actual errors are clearly correlated. The average expected errors were higher than the average actual errors. One of the possible causes for the excessive expected errors is that the trigonometric polynomial, even with 81 coefficients, has some likelihood outside the direct vicinity of the mode, as can be seen in Fig. 2. The overestimation of the uncertainties indicates that it may be feasible to use even more concentrated likelihoods.

There are some related observations in the literature on estimation of uncertainties using neural networks. In Euclidean spaces, the variance is commonly used to quantify the expected and actual uncertainty. [17] stated that uncertainty estimates by neuronal networks are often inaccurate. The authors proposed a way to amend this. However, since the proposed mapping can map each confidence level to almost any other value, this approach should be considered with caution as it can invalidate what the network has learned about the uncertainties. A better approach was proposed in [18]. While this approach still allows for too high and too low uncertainties to cancel out, it can be seen as the current state of the art.

Note that even if we found a way to derive the true average error from the expected error, we cannot directly use it in the filter. Unlike in the linear case, in which the variance may be used in the filter, we only use the expected error for quality assessment, and our filter relies directly on the Fourier coefficients. For us, it would be more valuable to improve the coefficients instead.



(a) The individual expected and actual errors are shown for the triangle data set. Some circles have been omitted to improve the visualization of the majority of the circles.



(b) Line going through the averages of 20 bins.

Fig. 5. Comparison of the expected and actual errors.

V. Conclusion and Outlook

In this paper, we presented two loss terms for training neural networks to convert image data into likelihood functions for directional estimation. The output of the network were Fourier coefficients that described a trigonometric polynomial that was used as the square root of the (potentially scaled) likelihood. Possible ambiguities due to rotational symmetries of the shapes were explicitly considered. The first loss term is based on the likelihood values at the possible true orientations. The second was inspired by the Hellinger distance. While the latter loss term does not yield comparable results for different numbers of coefficients, it leads to fast convergence and inherently allows for modeling uniform distributions.

A simple tracking scenario was designed, and a filter that uses the generated likelihoods was employed. The estimates derived using the filter were highly accurate. The uncertainties in the form of expected errors were clearly correlated with the actual errors, even though the precise values did not match.

A limitation of our paper is that we used synthetic data and had many images per shape in the training data. A potential area for future work is to use real-world data. Currently, the trained network can only be employed when the shape was included in the training data. It would be interesting to train a network on a larger variety of shapes and evaluate how it performs on unknown shapes. For this, one should think of a coherent way to define the orientation (or, in the case of ambiguities, possible orientations) of arbitrary shapes.

In future work, it may also be of interest to investigate how one can adapt the network to provide coefficients that lead to posterior densities for which the average estimated errors match the average actual errors more closely. This may lead to better likelihoods overall and thus also to better estimates. Another very interesting piece of future work would be to develop a similar approach for 3-D orientation estimation. For this, it would be an option to provide likelihoods for the hyperhemispherical grid filter [19].

Acknowledgments

This work is partially funded by the Helmholtz AI Cooperation Unit within the scope of the project *Ubiquitous Spatio-Temporal Learning for Future Mobility (ULearn4Mobility)*.

References

[1] J. Traa and P. Smaragdis, "A Wrapped Kalman Filter for Azimuthal Speaker Tracking," *IEEE Signal Processing Letters*, vol. 20, no. 12, pp. 1257–1260, 2013.

[2] R. S. Bucy and A. J. Mallinckrodt, "An Optimal Phase Demodulator," *Stochastics*, vol. 1, no. 1-4, pp. 3–23, 1975.

[3] Q. Q. Liu and J. B. Li, "Orientation Robust Object Detection in Aerial Images Based on R-NMS," *Procedia Computer Science*, vol. 154, pp. 650–656, Jan. 2019.

[4] F. Pfaff, G. Kurz, and U. D. Hanebeck, "Multimodal Circular Filtering Using Fourier Series," in *Proceedings of the 18th International Conference on Information Fusion (Fusion 2015)*, Washington D.C., USA, Jul. 2015.

[5] M. Azmani, S. Reboul, J.-B. Choquel, and M. Benjelloun, "A Recursive Fusion Filter for Angular Data," in *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics (ROBIO 2009)*, Dec. 2009.

[6] S. R. Jammalamadaka and A. Sengupta, *Topics in Circular Statistics*. World Scientific, 2001.

[7] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[8] I. Gilitschenski, R. Sahoo, W. Schwarting, A. Amini, S. Karaman, and D. Rus, "Deep Orientation Uncertainty Learning Based on a Bingham Loss," in *International Conference on Learning Representations*, 2019.

[9] H. Deng, M. Bui, N. Navab, L. Guibas, S. Ilic, and T. Birdal, "Deep Bingham Networks: Dealing with Uncertainty and Ambiguity in Pose Estimation," *arXiv:2012.11002 [cs]*, Dec. 2020, arXiv: 2012.11002.

[10] T. Avant and K. A. Morgansen, "Rigid Body Dynamics Estimation by Unscented Filtering Pose Estimation Neural Networks," in *2020 American Control Conference (ACC)*, Jul. 2020, pp. 2580–2586, iSSN: 2378-5861.

[11] A. Zygmund, *Trigonometric Series*, 3rd ed. Cambridge University Press, 2003, vol. 1 and 2.

[12] F. Pfaff, G. Kurz, and U. D. Hanebeck, "Nonlinear Prediction for Circular Filtering Using Fourier Series," in *Proceedings of the 19th International Conference on Information Fusion (Fusion 2016)*, Heidelberg, Germany, Jul. 2016.

[13] R. A. Kennedy and P. Sadeghi, *Hilbert Space Methods in Signal Processing*. Cambridge University Press, 2013.

[14] D. Masters and C. Luschi, "Revisiting Small Batch Training for Deep Neural Networks," *arXiv:1804.07612 [cs, stat]*, Apr. 2018, arXiv: 1804.07612.

[15] Shun'ichi Amari and Hiroshi Nagaoka, *Methods of Information Geometry*, ser. Translations of Mathematical Monographs, 2000, vol. 191.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha, "Gradient-Based Learning Applied to Document Recognition," p. 46, 1998.

[17] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate Uncertainties for Deep Learning Using Calibrated Regression," in *International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 2796–2804, iSSN: 2640-3498. [Online]. Available: http://proceedings.mlr.press/v80/kuleshov18a.html

[18] D. Levi, L. Gispan, N. Giladi, and E. Fetaya, "Evaluating and Calibrating Uncertainty Prediction in Regression Tasks," *arXiv:1905.11659 [cs, stat]*, Feb. 2020, arXiv: 1905.11659.

[19] F. Pfaff, K. Li, and U. D. Hanebeck, "A Hyperhemispherical Grid Filter for Orientation Estimation," in *Proceedings of the 23rd International Conference on Information Fusion (Fusion 2020)*, Virtual, Jul. 2020.