# A Gentle Introduction to Predictive Filters

Siome Klein Goldenstein[1]

**Abstract:** Predictive filters are essential tools in modern science. They perform state prediction and parameter estimation in fields such as robotics, computer vision, and computer graphics. Sometimes also called Bayesian filters, they apply the Bayesian rule of conditional probability to combine a predicted behavior with some corrupted indirect observation.
When we study and solve a problem, we first need its proper mathematical formulation. Finding the essential parameters that best describe the system is hard; modeling their behaviors over time is even more challenging. Usually, we also have an inspection mechanism that provides us with indirect measurements, the observations, of the hidden underlying parameters. We also need to deal with the concept of uncertainty, and use random variables to represent both the state and the observations.
Predictive filters are a family of estimation techniques. They combine the uncertain prediction from the system's dynamics and the corrupted observation. There are many different predictive filters, each dealing with different types of mathematical representations for random variables and system dynamics.
Here, the reader will find a dense introduction to predictive filters. After a general introduction, we discuss briefly discussion about mathematical modeling of systems: state representation, dynamics, and observation. Then, we expose some basic issues related to random variables and uncertainty modeling, and discuss four implementations of predictive filters, in order of complexity: the Kalman filter, the extended Kalman filter, the particle filter, and the unscented Kalman filter.

Keywords: Predictive Filters, Density Estimators, Kalman Filter, Particle Filter, Unscented Kalman Filter.

## 1 Introduction

In applied sciences, we use mathematical models to describe, understand, and deal with real objects. Applications are varied, and may involve guiding a mobile exploratory vehicle, understanding a human face and its expressions, or controlling a nuclear power plant reactor. A good mathematical representation lets us infer properties of the real entity, and understand beforehand the possible implications of our interactions with it.

The *state vector* of a model is the set of parameters that uniquely describes the con-

[1]Instituto de Computação, UNICAMP
Caixa Postal 6176, Campinas - SP, 13084-971 Brazil
siome@ic.unicamp.br

figuration of the object. The set of parameters that model an object is usually not unique, and each situation leads to a different set of design choices. When the object's parameters are not static, we also need a mathematical model for their change over time. We call a *system* the mathematical representation of states and dynamics.

Sometimes, the parameters we choose to represent a system are intrinsic to the problem but impossible to measure, such as the temperature in the core of a nuclear power plant. Fortunately, we can usually measure other quantities that are related to the unknown, but essential parameter, such as the temperature of the cooling water leaving the core. These indirect measurements are called *observations*, and can be used to infer information about the hidden parameter.

Real life is hard, and full of errors. The observations we make are not precise, and they may not explain all that is happening in the system. Also, our mathematical representation of the model is not exact, giving slightly incorrect results. To cope with all these factors, we need to incorporate the notion of uncertainties. There are many mathematical descriptions for uncertainties, but in this paper we use probabilities.

Predictive filters estimate the optimal state of a system. First, they use the mathematical model of the system dynamics to propagate the state's values and uncertainties. Later, they combine this preliminary estimate and the best that can be used from the observation. There are several predictive filters, each appropriate for a different type of uncertainty representation and dynamic modeling.

The Kalman filter is the simplest example of a predictive filter. It represents uncertainties as Gaussian random variables, fully described by a mean and a covariance matrix, and models the system with linear dynamics and observations. Since Gaussians are preserved under linear transformations, the Kalman filter's implementation uses only linear algebra operations.

If we need more complex models to describe a system, we may have to use nonlinear functions. The extended Kalman filter linearizes the system around the current state to propagate and estimate covariance matrices.

We can represent the random variables as a collection of samples, and allow nonlinear operations. In this case, we can use a particle filter, which is powerful and flexible. Unfortunately, it needs a number of samples exponentially related to the state's dimension.

Here, the last predictive filter we talk about is the unscented Kalman filter. It uses a small number of specially selected particles to propagate random variables over nonlinear transformations.

In this paper, we follow the same type of structure as in Maybeck's classical book [20]. In Section 2, we deal with the problem of building a mathematical model that describes the

state of the system (Section 2.1), its dynamics (Section 2.2), and how to inspect it at every moment (Section 2.3). Then, in Section 3, we review uncertainties: probabilities and random variables. Finally, in Section 4, we describe the Bayesian concepts of predictive filters, and look at some of its concrete implementations: the Kalman filter (Section 4.1), the extended Kalman filter (Section 4.2), the particle filter (Section 4.3), and the unscented particle filter (Section 4.4). In Section 5, we conclude this paper with a series of words of caution, as well as a series of pointers to where the reader can find more information.

## 2 System Representation

Predictive filtering techniques rely on a mathematical representation of the system. In this section, we discuss how to model a system's parameters and their dynamical aspects, and how to describe our indirect ability to inspect them.

A system is just like a black box, controlled by a set of parameters. To manipulate and understand a system, we need a mathematical model that describes these parameters and their behavior over time.

There are three general guidelines to modeling: first, the mathematical model should be as precise as possible – the system's behavior model should be as close as possible to reality. Second, the model should be tractable – we need to handle the mathematical model in an efficient way. Finally, the model should be as simple as possible – there is no need to describe a detail of the system, if this detail is of no consequence for the desired application. Usually, these three guidelines are incompatible, and we have to strike a balance. In the next three sections we describe the ideas behind state description, the observation procedure, and the dynamic behavior.

### 2.1 State Description

We use several real-valued variables to describe a system state.[2] If we use $n$ parameters to describe the system, we group them in a vector

$$\vec{x}_t \in \mathbb{R}^n,$$

that uniquely identifies the system's situation at time $t$. We now look at two examples.

*Example 1: Simple Robot* Let's consider a mobile robot, that can turn around its axes and move in the forward direction. We know that the robot cannot leave the floor, so we only

[2] It is possible, and sometimes more adequate, to model systems with graphs and discrete variables, but that is beyond the scope of this paper.

need two coordinates to describe its position. We also need a third parameter to describe its heading. This is illustrated in Figure 1.

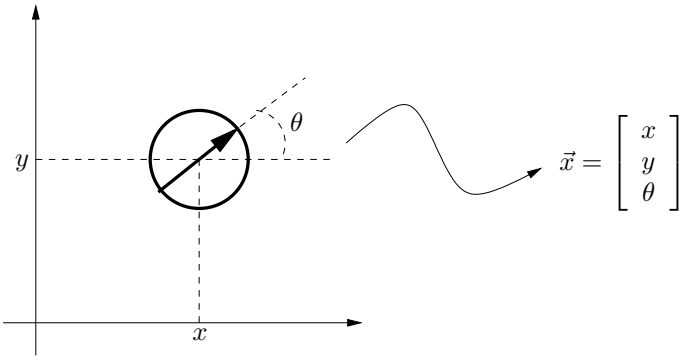


**Figure 1.** A simple state model of a robot, with $x$ and $y$ positions and forward direction $\theta$.

*Example 2: Human Face* When we view at a human face as a system, we can model deformation parameters. These parameters describe the effects of the muscles' actions on the surface of the face. In Figure 2 we see an illustration of how the state vector can affect the overall shape and position of a face.

### 2.2 Dynamic Behavior

In order to describe the dynamic behavior of a system, we take into account its state history and some external inputs, represented as $\vec{u_k} \in \mathbb{R}^l$. In control theory, we use the mathematical model, along with the initial configuration of the state, to plan the input $\vec{u}$ over time. This planned $\vec{u}$ guides the system's state according to our goals. In Bayesian state estimation, we do not know what the inputs really are (perhaps just some statistics), but we estimate the best state configuration using indirect observations.

There are two models for the passage of time, *continuous* and *discrete*. In the first one, time is a value $t \in \mathbb{R}$. In the second representation, time is a integer variable $k \in \mathbb{Z}$. Usually, the discrete time approach just represents a uniform sampling $t = k \cdot \Delta T$ of the continuous case. In this paper, we will only use the discrete representation.

A natural representation for the dynamics of discrete-time system is a difference equation. The next state depends on the current input and all the system's history

$$\vec{x}_{k+1} = f\left(\vec{u}_k, \vec{x}_k, \ldots, \vec{x}_0\right).$$

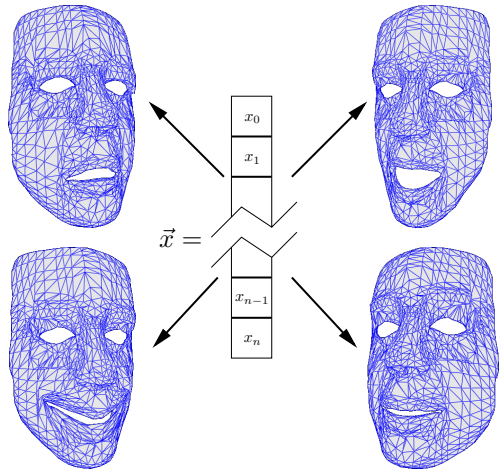


**Figure 2.** The state space of a deformable face describes the global position and orientation, as well as local changes such as eyebrow movement, jaw, and lip movement.

Fortunately, it is usually enough to consider that the next state depends only on the current state and the external input

$$\vec{x}_{k+1} = f\left(\vec{u}_k, \vec{x}_k\right), \tag{1}$$

where $f : \mathbb{R}^l \times \mathbb{R}^n \rightarrow \mathbb{R}^n$. If the system needs to take into account more past states, we use state enhancing techniques (Section 2.2.2) to be able to still use this simplification.

**2.2.1 Linear Modeling** In linear systems, the model of dynamics is a linear combination of the past state and the external inputs. Linear systems are mathematically easy to deal with, usually providing us with analytical, well-behaved, and efficient solutions. Sometimes it is possible to design the system's state variables to allow the use of linear dynamics.

*Example: One-dimensional uniform motion.* In this case, the state vector is

$$\vec{x} = \begin{bmatrix} x \\ v \end{bmatrix},$$

where x is the position, and $v$ the velocity. The dynamics can be represented as a simple matrix multiplication

$$\vec{x}_{k+i} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \vec{x}_k.$$

In general, the system depends on $m$ different inputs $\vec{u}_k \in \mathbb{R}^m$, and we represent the system dynamics through the difference equation

$$\vec{x}_{k+1} = A\vec{x}_k + B\vec{u}_k,$$

where $A \in \mathbb{R}^{n\times n}$, and $B \in \mathbb{R}^{n\times m}$.

**2.2.2 State Enhancing Techniques** During the mathematical modeling stage, there are many tricks we can play with the description and representation of a system. For example, suppose we need a more powerful model of dynamics than that of Equation 1, one that takes into account the two previous states,

$$\vec{x}_{k+1} = g\left(\vec{u}_k, \vec{x}_k, \vec{x}_{k-1}\right).$$

In order to use the one-state history formulation for systems, all we need is a change of variables

$$\vec{\mathcal{X}}_{k+1} = \begin{bmatrix} \vec{x}_k \\ \vec{x}_{k-1} \end{bmatrix},$$

where $\vec{\mathcal{X}}_i \in \mathbb{R}^{2m}$. The dynamics can then be written as

$$\vec{\mathcal{X}}_{k+1} = f\left(\vec{u}_k, \vec{\mathcal{X}}_k\right) = \begin{bmatrix} g\left(\vec{u}_k, \begin{bmatrix} I_n & 0_n \end{bmatrix}\vec{\mathcal{X}}_k, \begin{bmatrix} 0_n & I_n \end{bmatrix}\vec{\mathcal{X}}_k\right) \\ \begin{bmatrix} I_n & 0_n \end{bmatrix}\vec{\mathcal{X}}_k \end{bmatrix},$$

where $I_n$ is the $n$-dimension identity matrix, and $0_n$ is the $n$-dimension zero-valued matrix. It is worth noting that this technique is not limited to linear dynamics, it can take into account any finite number of previous states. In calculus, this idea is used to rewrite a high-degree differential equation as a system of first-degree differential equations.

### 2.3 Observation

To complete our mathematical formulation, we need to model the *observation*. An observation $\vec{y} \in \mathbb{R}^m$ is an indirect measurement of the state's value

$$\vec{y}_k = h\left(\vec{x}_k, \vec{u}_k\right). \tag{2}$$

In Section 4, we will see how to combine the information from the prediction from the system dynamics (Equation 1) and the observation, to obtain the best possible estimate.

## 3 Uncertainty Modeling

In prediction, estimation, and control, we have to deal with inexact quantities. There are different options to model and represent uncertainties, such as intervals and regions of

confidence [23, 24, 40], or random variables and probabilities [6, 27, 42]. In this paper, we only use random variables.

In this section, we briefly describe a few common terms and expressions from probability theory. For a short and good introduction, look for Chapter 2 in Wozencraft's classic communication book [42].

*Probability System* A probability system is a mathematical model composed of three entities: a *sample space*, a *set of events*, and a *probability measure* defined on these events. The sample space (generally referred to by the symbol $\Omega$) is a collection of possibilities, objects. An object in $\Omega$ is called a *sample point*, and is denoted by $\omega$. An event is a set of sample points, and a probability measure is the assignment of real numbers to the events defined on $\Omega$, such that this assignment satisfies the following conditions:

1. To every event $A_i$, there is a unique number $P[A_i]$ assigned such that $0 \leq P[A_i] \leq 1$.
2. $P[\Omega] = 1$.
3. If two events $A$ and $B$ are disjoint, $A \bigcap B = \emptyset$, $P[A \bigcup B] = P[A] + P[B]$.

*Conditional Probability* The knowledge about an event give us information regarding other events as well. If we know that an experiment of the sample space $\Omega$ belongs to event $B$, then the probability of it being also part of event $A$ is no longer $P[A]$. Effectively, in this case we narrow our sample space to the subspace $B$ of $\Omega$. In Figure 3 we can see a Venn diagram representing $\Omega$, $A$, and $B$.

If a sample belongs to event $B$, and $P[B] \neq 0$, then the probability that this sample also belongs to event $A$ is the conditional probability

$$P[A|B] \triangleq \frac{P[A \cap B]}{P[B]}. \tag{3}$$

When $P[A]$ is also nonzero, we can write

$$P[A \cap B] = P[A|B]\, P[B] = P[B|A]\, P[A].$$

This conditional probability relation is called *Bayes rule*, and it is the essence of the predictive, or Bayesian, filters.

*Random Variables* A random variable is a mapping $x(\omega) : \Omega \rightarrow \mathbb{R}$ of all samples from $\Omega$ to $\mathbb{R}$. We call the *probability distribution function*

$$F_x(\alpha) \triangleq P[\{\omega \in \Omega : x(\omega) \leq \alpha\}]. \tag{4}$$

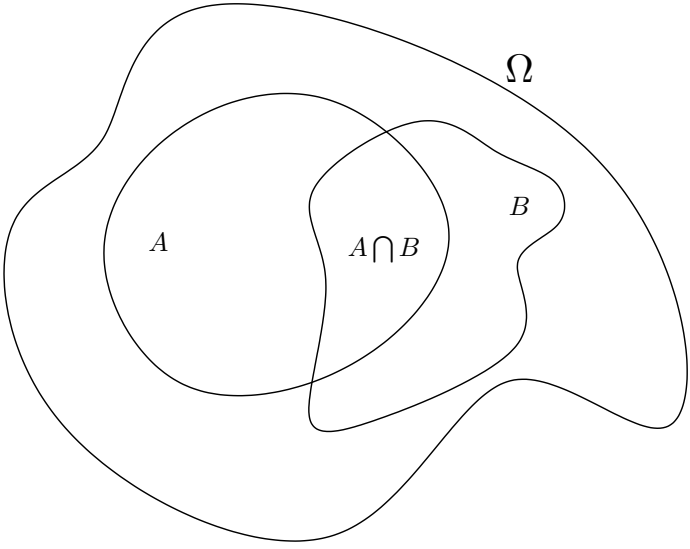


**Figure 3.** Two events, $A$ and $B$, in a sample space $\Omega$.

The probability distribution function provides us with the probability of the value of the random variable being smaller or equal to every number in $\mathbb{R}$, and it has a few useful properties:

1. $F_x(\alpha) \geq 0$; for $-\infty < \alpha < \infty$.
2. $F_x(-\infty) = 0$.
3. $F_x(\infty) = 1$.
4. If $a > b$, $\quad F_x(a) - F_x(b) = P[\{\omega \in \Omega : b < x(\omega) \leq \alpha\}]$.
5. If $a > b$, $\quad F_x(a) \geq F_x(b)$.

We can define the *probability density function*[3]

$$p_x(\alpha) \triangleq \frac{dF_x(\alpha)}{d\alpha}.$$

[3]We have to be careful when $F_x$ is not $\mathcal{C}^0$ or $\mathcal{C}^1$, $p_x$ may not be a classic function, and it may be necessary to use Dirac's impulse notation.

In the multivariate case the probability distribution function is

$$F_{x_1,\ldots,x_n}(\alpha_1,\ldots,\alpha_n) \triangleq P\left[\{\omega \in \Omega : x_1(\omega) \leq \alpha_1,\ldots,x_n(\omega) \leq \alpha_n\}\right]; \\ -\infty < \alpha_1,\ldots,\alpha_n < \infty,$$

and the probability density function is

$$p_{x_1,\ldots,x_n}(\alpha_1,\ldots,\alpha_n) \triangleq \frac{d^n}{d\alpha_1 \cdots d\alpha_n} F_{x_1,\ldots,x_n}(\alpha_1,\ldots,\alpha_n).$$

It is easy to see that the probability that an experiment lies in a volume $I \in \mathbb{R}^n$ is

$$P\left[\{\omega \in \Omega : \vec{x}(\omega) \text{ in } I\}\right] = \int_I p_{\vec{x}}(\vec{\alpha})\, d\vec{\alpha}.$$

We use a probability density function to describe a random variable.

*Statistical Indicators* Although a random variable is defined through its probability density function, sometimes we need simpler indicators or diagnostic tools to describe the random variable's characteristics. For example, the *mean*

$$\overline{\vec{x}} = E[\vec{x}] = \int \vec{\alpha} p_{\vec{x}}(\alpha)\, d\alpha,$$

is the expected value of a random variable.

Another common indicator is the variance, also known as the second central moment. The variance is an indicator of how concentrated the probability density function is around the mean. In one dimension the variance is calculated as $E\left[(x-\overline{x})^2\right]$. In the multivariate case it is called the covariance matrix, and obtained by

$$\Lambda = E\left[\left(\vec{x}-\overline{\vec{x}}\right)\left(\vec{x}-\overline{\vec{x}}\right)^\top\right].$$

As we will see in the next section, Gaussian distributions are fully specified by their mean and covariance matrix.

### 3.1 Gaussian Random Variables

Since a random variable is defined by its probability density function, to specify a random variable we have to describe a multivariate function. One method to do that is to choose a family of functions controlled by just a few parameters. When we do that, we are using a *parametric representation*.

A simple family of distributions is the Gaussian, whose distribution is

$$p_x(\alpha) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(\alpha-\bar{x})^2/\sigma^2},$$

which is uniquely identified by the mean $\overline{x}$ and variance $\sigma^2$. In Figure 4 we show a plot of the probability density function and of the probability distribution function of a one-dimensional Gaussian random variable with mean $\overline{x} = 0$ and variance $\sigma^2 = 1$.

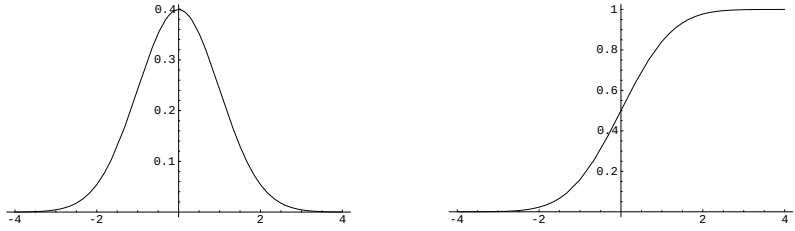


**Figure 4.** One-dimensional Gaussian random variable with $\overline{x} = 0$ and $\sigma^2 = 1$. On the left, the probability density function, on the right, the probability distribution function.

In the multivariate case, a Gaussian random variable $\vec{x}$ is described by a mean vector $\overline{\vec{x}} \in \mathbb{R}^n$ and a covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$, with probability density function

$$p_{\vec{x}}(\vec{\alpha}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(\vec{\alpha}-\overline{\vec{x}})^\top \Sigma^{-1} (\vec{\alpha}-\overline{\vec{x}})}. \tag{5}$$

Additionally, if we apply a linear transformation $T$

$$\vec{x}' = T.\vec{x},$$

the resulting random variable $\vec{x}'$ will still be a Gaussian, with

$$\overline{\vec{x}'} = T.\overline{\vec{x}}, \qquad \text{and} \quad \Sigma' = T.\Sigma.T^\top.$$

In Figure 5 we see the contour plot of two-dimensional Gaussian random variables (darker means larger value of the probability density function). On the left, we show the two original random variables. On the right, the result of subjecting both originals to the same linear transformation.

Gaussian random variables are convenient to use. They are simple to understand and computationally efficient, since we only need to store the mean vector and covariance matrix. They are easy to manipulate, we just use matrix multiplications to propagate covariances and means across linear transformations. These are the properties that make Kalman filters (Section 4.1) so powerful, simple, and popular in the literature.

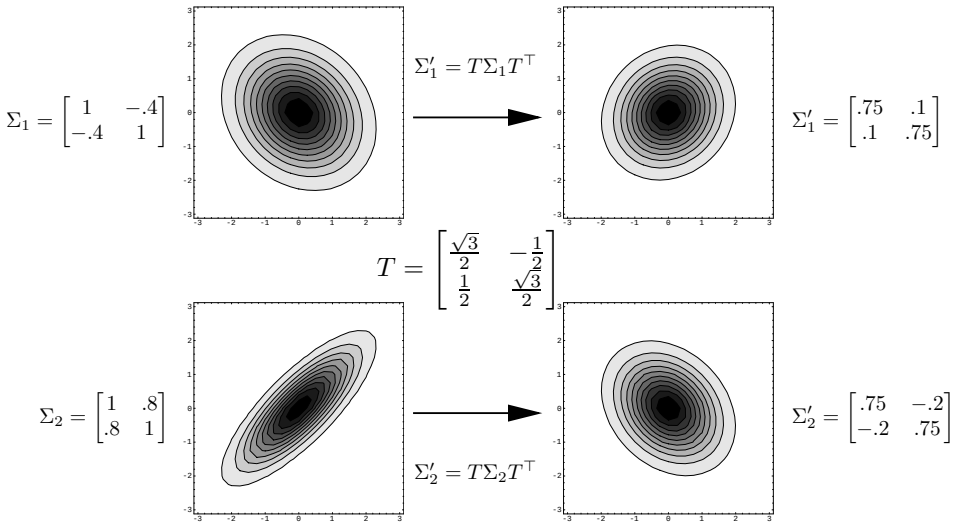


**Figure 5.** Contour plots of Gaussian random variables (dark means larger value of the probability density function). On the left, we show two different random variables. On the right, we show the result of subjecting both of them to the same linear transformation.

### 3.2 Samples as Distributions

Sometimes a Gaussian random variable is not able to describe, not even in an approximate way, the state of a system. For example, a Gaussian random variable is not able to represent simultaneously two strong groups of possibilities (as in the left of Figure 6). One possible approach to deal with this limitation is to look for a more powerful family of parametric models, such as a *mixture of Gaussians* [21]. Another approach is to use a particle-based, or sample-based, representation of the probability density function.

A particle-based representation of a distribution is not described by a few parameters, it is described by a large number of samples of the state space. Areas where the probability density function values are large will have a higher concentration of samples than areas where the probability density function values are smaller. In Figure 6, we illustrate the contour plot (dark means larger values of the probability density function) of a multimodal two-dimensional distribution, and a possible particle-based representation.

We can obtain information about the underlying random variable directly from the

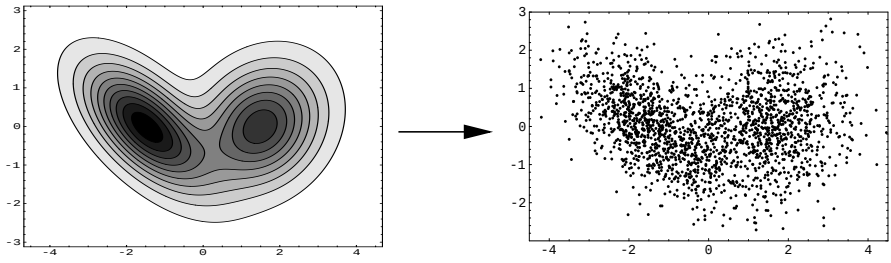

**Figure 6.** In the left, the contour plot of a multi-modal two-dimensional distribution (dark means larger values of the probability density function). In the right, its sampled representation.

samples. For example, to calculate the mean $\overline{\vec{x}}$ and covariance $\Sigma$,

$$\overline{\vec{x}} = \frac{1}{n}\sum_{i=1}^{n} \vec{x}^i \qquad \text{and} \quad \Sigma = \frac{1}{n}\sum_{i=1}^{n} \left(\vec{x}^i - \overline{\vec{x}}\right)\left(\vec{x}^i - \overline{\vec{x}}\right).^{\top}$$

## 4 Bayesian State Prediction

Predictive, or Bayesian, filters present a solid framework for combining a model of the system's evolution with an observation of the system's state.

We call $\vec{x}_k$, the multi-dimensional random variable that represents the system's state at the discrete-time $k$. The system's model provides us with a reasonable prediction $\vec{x}'_{k+1}$ of the state vector at time $k+1$:

$$\vec{x}'_{k+1} = f_k\left(\vec{x}_k, \vec{w}_k\right), \tag{6}$$

where $f_k : \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^n$, $n$ is the dimension of the state vector $\vec{x}$, and $l$ is the dimension of the multivariate noise $\vec{w}$.

At each time sample $k$, we make an observation $\vec{y}_k$, or measurement, of the system. These observations are related to the state vector as:

$$\vec{y}_k = h_k\left(\vec{x}_k, \vec{v}_k\right), \tag{7}$$

where $h_k : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^p$, $p$ is the dimension of the measurements $\vec{y}$, and $r$ is the dimension of the multivariate noise $\vec{v}$. All the information available at time sample $k$ is represented as $D_k = \{\vec{y}_1, \ldots, \vec{y}_k\}$. We also need to know the probability distribution function of the state vector at the starting time, also known as the *prior*, $p\left(\vec{x}_1 | D_0\right) = p\left(\vec{x}_1\right)$.

Predictive filters recursively find $p(\vec{x}_k|D_k)$ from $p(\vec{x}_{k-1}|D_{k-1})$ and $\vec{y}_k$. This computation is done in two steps. In the first step

$$p(\vec{x}_k|D_{k-1}) = \int p(\vec{x}_k|\vec{x}_{k-1})\, p(\vec{x}_{k-1}|D_{k-1})\, d\vec{x}_{k-1},$$

but using Equation 6 it becomes

$$p(\vec{x}_k|D_{k-1}) = \int f_k(\vec{x}_{k-1}, \vec{w}_{k-1})\, p(\vec{x}_{k-1}|D_{k-1})\, d\vec{x}_{k-1}. \tag{8}$$

The next step, also called correction, is a direct application of the Bayes rule:

$$p(\vec{x}_k|D_k) = \frac{p(\vec{y}_k|\vec{x}_k)\, p(\vec{x}_k|D_{k-1})}{p(\vec{y}_k|D_{k-1})},$$

and using Equation 7 we get

$$p(\vec{x}_k|D_k) = \frac{h_k(\vec{x}_k, \vec{v}_k)\, p(\vec{x}_k|D_{k-1})}{\int h_k(\vec{x}_k, \vec{v}_k)\, p(\vec{x}_k|D_{k-1})\, d\vec{x}_k}. \tag{9}$$

Each predictive filter solves the Bayesian recursive Equations 8 and 9 in a different way, using different assumptions. The Kalman filter, in Section 4.1, has a closed algebraic solution, but it assumes multivariate Gaussians distributions, and linear $f_k$ and $h_k$. Since Gaussian distributions are preserved under linear transformations, the Kalman filter only has to estimate the mean and covariance matrices. The extended Kalman filter, in Section 4.2, allows $f_k$ and $h_k$ to be nonlinear, but uses a linearization around the current state to propagate its covariances.

A particle filter, in Section 4.3, does not assume $f_k$ or $h_k$ to be linear, and neither does it use any particular parametric distribution. It applies $f$ to samples that represent the prior, and according to the Bayes rule, uses $h_k$ to resample from this new distribution.

The Unscented filter, in Section 4.4, uses a set of carefully chosen samples to represent the distributions, where the number of particles grows only linearly with the dimension of the state vector, and guarantees that at least the second moments are correct.

Predictive filters are ideal for tracking from noisy and corrupted data, as the measurements are smoothed out according to the system's evolution model. Unfortunately, they work as expected only if we know the real distribution of the measurement, $p(\vec{y}_k|\vec{x}_k)$ at every point in time. Later, in Figure 8, we can see that an incorrect estimate of the observation's distribution will lead to a poor final estimate.

### 4.1 Kalman Filter

When the system is linear, and Gaussian random variables are an appropriate representation of the system's state, the Kalman Filter is the optimal choice for parameter estimation.

We write the typical system as

$$\vec{x}_k = \Phi_{k-1}\vec{x}_{k-1} + \vec{w}_{k-1} \quad (10)$$

$$\vec{y}_k = H_k\vec{x}_k + \vec{v}_k, \quad (11)$$

where $\vec{x}_k$ is the $n$-dimension multivariate Gaussian distribution representing the system's state at time $k$, $\vec{y}_k$ is the $m$-dimension multivariate Gaussian distribution representing the observation of the system at time $k$, $\Phi$ is the $n \times n$ transition matrix, $\vec{w}$ is the prediction noise – an $n$-dimension multivariate Gaussian random variable with zero mean and covariance matrix $Q$, $H$ is the $m \times n$ observation matrix, and $\vec{v}$ is the observation noise – an $m$-dimension multivariate Gaussian random variable with zero mean and covariance matrix $R$.

For every iteration, the Kalman filter performs three tasks: prediction, weighting, and correction. Since it is a recursive process, the Kalman filter only keeps track of the current state's distribution $\vec{x}_k$, fully described by the mean vector $\overline{\vec{x}}_k$ and covariance matrix $\Sigma_k$.

*The prediction stage* The Kalman filter uses the system's dynamic, in Equation 10, to calculate the optimal state configuration $\vec{x}'_k$, described by mean $\overline{\vec{x}'_k}$ and covariance matrix $\Sigma'_k$,

$$\overline{\vec{x}'_k} = \Phi_{k-1}\overline{\vec{x}}_{k-1} \quad (12)$$

$$\Sigma'_k = \Phi_{k-1}\Sigma_{k-1}\Phi_{k-1}^{\top} + Q_{k-1}. \quad (13)$$

Since $\vec{w}_{k-1}$ has zero mean, it does not influence the mean. The noise's effect is the increase of the prediction's covariance matrix. In plain words, it decreases the prediction's precision – the larger the noise, the more unreliable the prediction. In the absence of an observation, this is the optimal estimation.

*The weighting stage* At every iteration, the Kalman filter receives an $n$-dimension observation vector $\vec{y}_k$ as input. This vector is an indirect measurement of the unknown state of the system, modeled through Equation 11. The observation has an uncertainty, represented by the additive noise $\vec{v}_k$, a Gaussian random variable. The covariance matrix $R_k$ represents the precision of the observation.

Intuitively, we should rely more on the observation when it is more precise than the prediction, and vice-versa. Unfortunately, this characterization is not clear to the naked eye – the observation is usually indirectly related to the state $\vec{x}_i$, through $H_k$, the observation and the prediction do not lie in the same dimensional space, and their principal axes (eigenvectors of the covariance matrices) are not aligned.

The weighting stage calculates a weighting matrix $K_k \in \mathbb{R}^{n \times m}$ at every iteration. To do so, this step only uses the covariance matrix $\Sigma'_k$ of the prediction, the covariance matrix $R_k$ of the observation, and the mathematical model $H_k$ of the observation:

$$K_k = \Sigma'_k H_k^\top \left( H_k \Sigma'_k H_k^\top + R_k \right)^{-1}. \tag{14}$$

The matrix $K_k$ has two roles: first, it projects values from the $m$-dimension observation space back into the $n$-dimension parameter space. Second, it decides how much should be used of the difference between the actual observation and the predicted observation $\vec{y}'_k = H_k \vec{x}'_k$. In the extreme case, when $\Sigma'_k = 0$ the prediction is perfect, so $K_k$ is zero. Similarly, when $R_k = 0$ the observation is perfect, so $K_k$ behaves as the pseudo-inverse of $H_k$.

The value of $K_k$ is temporary, and only used in the next stage to perform the optimal estimation. It is not carried over for the next iterations of the filter.

*The correction stage* In the last stage of the Kalman filter, we use the weighting matrix $K_k$ to optimally combine prediction and observation together. The resulting Gaussian random variable $\vec{x}_k$ has mean

$$\overline{\vec{x}}_k = \overline{\vec{x}}'_k + K_k \left( \vec{y}_k - H_k \overline{\vec{x}}'_k \right),$$

and covariance matrix $\Sigma_k = (I - K_k H_k) \Sigma'_k$. Unfortunately, this evaluation form for the covariance matrix $\Sigma_k$ is is not numerically stable, after several iterations it would eventually estimate $\Sigma_k$ as an invalid covariance matrix – it would not be symmetric positive definite[4]. We use the alternate form

$$\Sigma_k = (I - K_k) \Sigma'_k (I - K_k)^\top + K_k R_k K_k^\top$$

instead. It ensures that all intermediate values are symmetric positive definite.

In Algorithm 1, we display all stages and calculations of the Kalman filter.

### 4.2 Extended Kalman Filter

In many applications, the linearity requirement of the Kalman filter is not enough to model the system's dynamics, or the observation function. We describe the system in a more general way,

$$\begin{aligned} \vec{x}_k &= f(\vec{x}_{k-1}) + \vec{w}_{k-1}, \\ \vec{y}_k &= h(\vec{x}_k) + \vec{v}_k, \end{aligned} \tag{15}$$

[4] A symmetric matrix with all eigenvalues larger than zero.

**Algorithm 1** Kalman filter.

**Starts with:**

$\vec{x}_0$ is the prior, the Gaussian random variable of the state at time $k = 0$, with mean $\overline{\vec{x}_0}$ and covariance $\Sigma_0$.

**Prediction:** Use the system's dynamics equation to predict the new random variable,

$$\begin{aligned} \overline{\vec{x}'_k} &= \Phi_{k-1}\overline{\vec{x}}_{k-1}, \\ \Sigma'_k &= \Phi_{k-1}\Sigma_{k-1}\Phi_{k-1}^{\top} + Q_{k-1}. \end{aligned}$$

**Weighting:** Find out how to best combine prediction and observation at time $k$,

$$K_k = \Sigma'_k H_k^{\top} \left(H_k \Sigma'_k H_k^{\top} + R_k\right)^{-1}.$$

**Correction:** Combine prediction and observation,

$$\begin{aligned} \overline{\vec{x}}_k &= \overline{\vec{x}}'_k + K_k \left(\vec{y}_k - H_k \overline{\vec{x}}'_k\right), \\ \Sigma_k &= \left(I - K_k\right)\Sigma'_k \left(I - K_k\right)^{\top} + K_k R_k K_k^{\top}. \end{aligned}$$

where $f$ is a nonlinear function that will predict the state based on its previous value, $\vec{w}_{k-1}$ is the prediction noise, an $n$-dimensional Gaussian random variable with zero mean and covariance matrix $Q_{k-1}$, $h$ is the nonlinear mathematical model that calculates the observation based on the state's value, and $\vec{v}_k$ is the observation noise, an $m$-dimensional Gaussian random variable with zero mean and covariance matrix $R_k$.

The extended Kalman filter represents the states as Gaussian random variables. At every iteration, it applies the Taylor expansion to linearize both prediction and observation functions around the current state. It uses the linear approximation to propagate the covariance matrices. At step $k$ we construct a system

$$\begin{aligned} \vec{x}_k &= \Phi_{k-1}\vec{x}_{k-1} + \vec{w}_{k-1}, \\ \vec{y}_k &= H_k \vec{x}_k + \vec{v}_k, \end{aligned} \tag{16}$$

where

$$\Phi_{k-1} = \left.\frac{\partial f\left(\vec{x}\right)}{\partial \vec{x}}\right|_{\overline{\vec{x}}_{k-1}}, \qquad \text{and} \quad H_k = \left.\frac{\partial h\left(\vec{x}\right)}{\partial \vec{x}}\right|_{\overline{\vec{x}}_k}.$$

The extended Kalman filter is summarized in Algorithm 2. Note that the algorithm

uses Equation 15 to propagate the means, and Equation 16 to propagate the covariance matrices.

**Algorithm 2** Extended Kalman filter.

**Starts with:**

$\vec{x}_0$ is the prior, the Gaussian random variable of the state at time $k = 0$, with mean $\overline{\vec{x}_0}$ and covariance $\Sigma_0$.

**Prediction:** Linearize the system, and use dynamics equation to predict the new random variable,

$$\begin{aligned} \overline{\vec{x}'_k} &= h\left(\overline{\vec{x}}_{k-1}\right), \\ \Phi_{k-1} &= \left.\frac{\partial f\left(\vec{x}\right)}{\partial \vec{x}}\right|_{\overline{\vec{x}}_{k-1}}, \\ \Sigma'_k &= \Phi_{k-1}\Sigma_{k-1}\Phi_{k-1}^{\top} + Q_{k-1}. \end{aligned}$$

**Weighting:** Linearize observation relation, and find out how to best combine prediction and observation at time $k$,

$$\begin{aligned} H_k &= \left.\frac{\partial h\left(\vec{x}\right)}{\partial \vec{x}}\right|_{\overline{\vec{x}}_k}, \\ K_k &= \Sigma'_k H_k^{\top}\left(H_k\Sigma'_k H_k^{\top} + R_k\right)^{-1}. \end{aligned}$$

**Correction:** Combine prediction and observation,

$$\begin{aligned} \overline{\vec{x}}_k &= \overline{\vec{x}}'_k + K_k\left(\vec{y}_k - h\left(\overline{\vec{x}}'_k\right)\right), \\ \Sigma_k &= \left(I - K_k\right)\Sigma'_k\left(I - K_k\right)^{\top} + K_k R_k K_k^{\top}. \end{aligned}$$

If at the current state the linearization of the functions $f$ and $h$ are not good, i.e. higher order terms of the Taylor expansion are not negligible, then the extended Kalman filter will make gross mistakes in the estimations of the covariance matrices, which will in turn corrupt the correct fusion of the prediction with the observation.

With minor reordering, the algorithm can be rewritten to linearize $h$ around the prediction $\overline{\vec{x}'_k}$ instead of $\overline{\vec{x}}_{k-1}$, which can improve the convergence under borderline situations. Nevertheless, the need for this kind of adjustment is usually a strong indicator that the ex-

tended Kalman filter is not the most indicated technique for the problem. Perhaps a simplification of the mathematical model, a change in variables, a decrease the time $\Delta T$ between samples, or the use of a different predictive filter is in order.

### 4.3 Particle Filter

In this section, we use particles to represent the probability density function of non-parametric random variables, as initially introduced in Section 3.2. With a particle-based representation, the random variables can have multiple modes. This property enables particle filters to automatically keep parallel track of possible, and sometimes conflicting, solutions when confronted with temporarily insufficient data. As before, we can write the underlying modeling system as

$$\begin{aligned} \vec{x}_k &= f\left(\vec{x}_{k-1}\right) + \vec{w}_{k-1}, \\ \vec{y}_k &= h\left(\vec{x}_k\right) + \vec{v}_k, \end{aligned} \tag{17}$$

where $f$ is a nonlinear function that predicts the state based on its previous value, $\vec{w}_{k-1}$ is the prediction noise – an $n$-dimensional random variable with zero mean, $h$ is the nonlinear mathematical model that calculates the $m$-dimension observation based on the state's value, and $\vec{v}_k$ is the observation noise – an $m$-dimensional random variable with zero mean.

The algorithm starts with the prior of the state, a collection of samples

$$\vec{x}_0 \sim \left\{\vec{x}^1, \vec{x}^2, \ldots, \vec{x}^m\right\}_0, \tag{18}$$

where we use the symbol $\sim$ to indicate that the random variable $\vec{x}_0$ is being represented by the $m$ $n$-dimensional samples $\vec{x}^i$.

Like the previous techniques, particle filters have also three stages: prediction, weighting, and correction. Since the random variables' representation is particle-based, the particle filter implements each stage in its own unique way.

*The prediction stage* The prediction stage of a particle filter is simple. It does not require any linearization, such as in Section 4.2 with the extended Kalman filter, it just applies the function $f$ over every single particle,

$$\vec{x}'^i_k = f\left(\vec{x}^i_{k-1}\right) + \vec{w}_{k-1} \longrightarrow \vec{x}'_k \sim \left\{\vec{x}'^1_k, \vec{x}'^2_k, \ldots, \vec{x}'^m_k\right\}. \tag{19}$$

In Figure 7 we illustrate how particles can be used to propagate a random variable's probability density function over nonlinear transformations, when a parametric method would not be able to cope. It is essential to be able to efficiently sample from $\vec{w}_k$ for every particle.

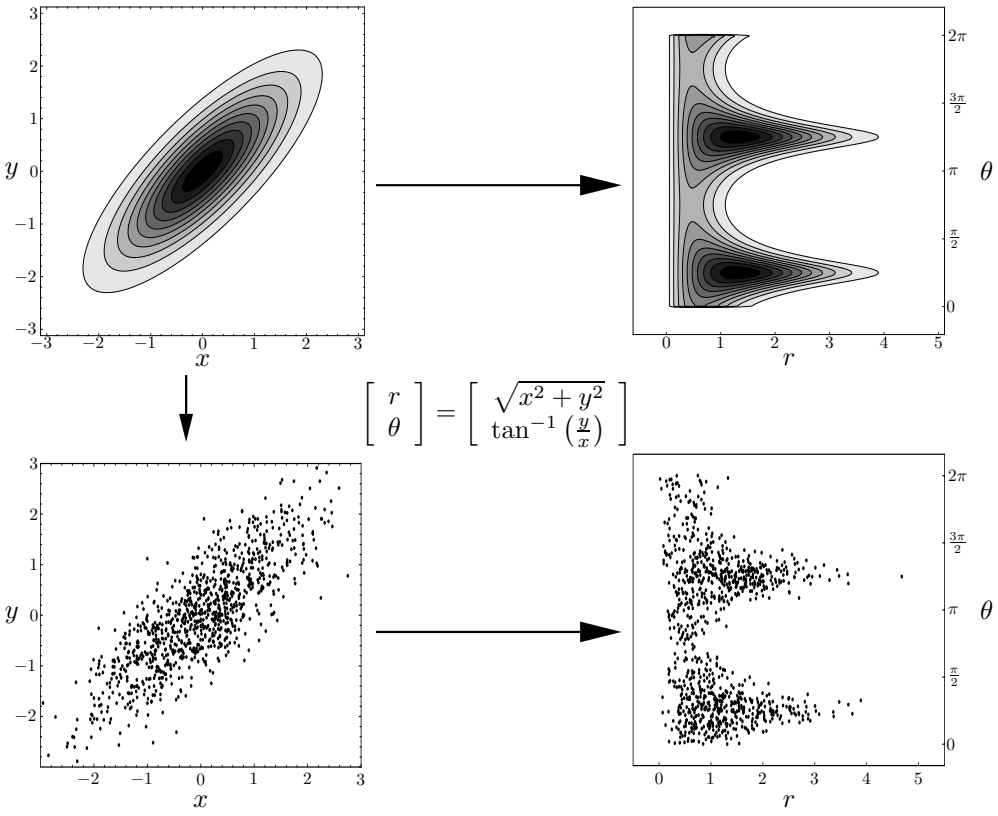


**Figure 7.** Nonlinear transformation over a sampled-based description of a random variable.

*The weighting stage* The weighting stage is the actual data fusion stage. For every particle $i$ of the prediction $\vec{x}'_k$, the filter calculates a weight

$$\omega_k^i = p\left(\vec{x}_k'^i | \vec{y}_k\right),$$

that is the conditional probability of the sample given the observation. For every particle $\vec{x}_k'^1$, the filter evaluates the probability distribution function of $\vec{v}_k$ at the point $\vec{x}_k'^i - \vec{y}_k$. The result of the weighting stage is a collection of weighted samples, represented as a collection of pairs $\left(\vec{x}_k'^i, \omega_k^i\right)$,

$$\left\{\left(\vec{x}_k'^1, \omega_k^1\right), \left(\vec{x}_k'^2, \omega_k^2\right), \ldots, \left(\vec{x}_k'^m, \omega_k^m\right)\right\}.$$

*The correction stage* The result of the previous state already represents the desired distribution, but it is not presented in a form that the filter can use recursively. This stage is also called *resampling*. This stage takes $m$ samples of the set, using their weights to give them different probabilities of being selected,

$$\left\{\left(\vec{x}'^1_k, \omega^1_k\right), \left(\vec{x}'^2_k, \omega^2_k\right), \ldots, \left(\vec{x}'^m_k, \omega^m_k\right)\right\} \xrightarrow{\text{re-sample}} \vec{x}_k \sim \left\{\vec{x}^1_k, \vec{x}^2_k, \ldots, \vec{x}^m_k\right\}.$$

The implementation of resampling is straightforward: each particle is mapped to an interval in $[0, 1]$, with length $\frac{\omega^i}{\sum_i \omega^i}$. We then sample a number $s$ from the uniform distribution in $[0, 1]$, and pick the particle that maps to the interval which contains $s$. Algorithm 3 summarizes the particle filter.

**Algorithm 3** Particle filter.

**Starts with:** The non-parametric prior distribution of the state is

$$\vec{x}_0 \quad \sim \quad \left\{\vec{x}^1_0, \vec{x}^2_0, \ldots, \vec{x}^m_0\right\},$$

represented by a set of $m$ unweighted samples.

**Prediction:** Use the dynamics equation to find a first estimate of the new random variable, as a new collection of samples,

$$\vec{x}'^i_k = f\left(\vec{x}^i_{k-1}\right) + \vec{w}_{k-1} \longrightarrow \vec{x}_k \sim \left\{\vec{x}'^1_k, \vec{x}'^2_k, \ldots, \vec{x}'^m_k\right\}.$$

Note that in order to avoid particle collapse, the noise $\vec{w}_{k-1}$ has to be re-sampled and added to each particle.

**Weighting:** Find out how to best combine prediction and observation at time $k$ by using the observation's density distribution to assign weights to every sample of the prediction,

$$\omega^i_k = p\left(\vec{x}'^i_k | \vec{y}_k\right) \longrightarrow \left\{\left(\vec{x}'^1_k, \omega^1_k\right), \left(\vec{x}'^2_k, \omega^2_k\right), \ldots, \left(\vec{x}'^m_k, \omega^m_k\right)\right\}.$$

**Correction:** Re-sample, to find new unweighted samples,

$$\left\{\left(\vec{x}'^1_k, \omega^1_k\right), \left(\vec{x}'^2_k, \omega^2_k\right), \ldots, \left(\vec{x}'^m_k, \omega^m_k\right)\right\} \xrightarrow{\text{re-sample}} \vec{x}_k \sim \left\{\vec{x}^1_k, \vec{x}^2_k, \ldots, \vec{x}^m_k\right\}.$$

Particle filters have an important role in state estimation; they deal in a unique way with non-parametric distributions and thrive with nonlinearities. They are particularly suitable when the observations may get extremely corrupted, as they will, from time to time,

and deal well with complicated and indirect types of observations. Particle filters are closely related to Monte Carlo [33] and genetic algorithms [22] methods.

Unfortunately, the particle filter technique has some serious drawbacks. The forerunner among them is the number of particles that it needs in order to correctly describe the state's distribution. In the literature, researchers have been using between 200 and 1000 particles, but in the vanilla version, as we describe here, the number of necessary particles grows exponentially with the dimension $n$ of the system's state space.

Due to the resampling stage, the final estimate will have multiple occurrences of several particles. This is the reason why the addition of the $\vec{w}_k$ noise in the prediction stage is of paramount importance. It keeps the system from suffering from a phenomenon called *particle collapse.* We need to be careful to correctly and efficiently sample from $\vec{w}_k$ [33] for each particle, and that is why so many people choose to use Gaussian or Uniform distributions for that. Finally, the weighting stage also requires the explicit evaluation of $\vec{v}_k$'s probability density function.

### 4.4 Unscented Kalman Filter

The last predictive filter we describe in this paper is the unscented Kalman filter. Like the extended Kalman filter, in Section 4.2, the unscented Kalman filter represents the system's state as Gaussian random variables, but unlike it, there is no linearization to propagate the covariance matrix. The system is represented as

$$\begin{aligned} \vec{x}_k &= f\left(\vec{x}_{k-1}, \vec{w}_{k-1}\right), \\ \vec{y}_k &= h\left(\vec{x}_k, \vec{v}_k\right), \end{aligned}$$

where $f$ is the nonlinear prediction function, $h$ is the nonlinear observation function, $\vec{w}_{k-1}$ is the prediction noise, and $\vec{v}_k$is the observation noise.

The unscented Kalman filter is based on the unscented transform [41, 15], which propagates the random variable's distribution information using only $2n + 1$ weighted particles, where $n$ is the dimension of the system's state. As before, there are three stages: prediction, weighting, and correction.

The recursive procedure begins with the knowledge of the prior distribution $\vec{x}_0$ of the system's state, with mean $\overline{\vec{x}}_0$ and covariance matrix $\Sigma_0$.

*The prediction stage* We proceed to calculate the $2n + 1$ particles[5] that appropriately represent the distribution,

$$\vec{x}_{k-1} \sim \left\{\vec{x}_{k-1}^0, \ldots, \vec{x}_{k-1}^{2n+1}\right\},$$

[5]These particles are called *sigma points* in the unscented transform formulation.

where

$$\begin{cases} \vec{x}^{\,0}_{k-1} = \overline{\vec{x}}_{k-1} \\ \vec{x}^{\,i}_{k-1} = \overline{\vec{x}}_{k-1} + \left(\sqrt{(n+\lambda)\,\Sigma_{k-1}}\right)_i & i = 1, \ldots, n \\ \vec{x}^{\,i}_{k-1} = \overline{\vec{x}}_{k-1} - \left(\sqrt{(n+\lambda)\,\Sigma_{k-1}}\right)_{i-n} & i = n+1, \ldots, 2n \end{cases}$$

where $\lambda = \alpha^2\,(n+\kappa) - n$ is a scaling parameter, $\alpha$ is set to a small values (in the order of $10^{-3}$) and is related to the spread of the sample points around the mean, $\kappa$ is usually 0, and $\left(\sqrt{(n+\lambda)\,\Sigma_{k-1}}\right)_i$ is the $i^{th}$ row of the square root matrix. These samples will be symmetrically spread around the mean.

Additionally, we also calculate two weights for every particle, one for mean and the other for covariance estimation

$$\begin{cases} {}^{m}\omega^{0}_{k-1} = \frac{\lambda}{n+\lambda} \\ {}^{c}\omega^{0}_{k-1} = \frac{\lambda}{n+\lambda} + \left(1-\alpha^2+\beta\right) \\ {}^{m}\omega^{i}_{k-1} = {}^{c}\omega^{i}_{k-1} = \frac{\lambda}{2(n+\lambda)} & i = 1, \ldots, 2n \end{cases}$$

where $\beta$ is a parameter that incorporates knowledge of the prior distributions (in the case of Gaussians distributions, the optimal value is $\beta = 2$).

The filter propagates the particles,

$$\vec{x}'^{\,i}_k = f\left(\vec{x}^{\,i}_{k-1}, \vec{w}_{k-1}\right) \longrightarrow \vec{x}'_k \sim \left\{\vec{x}'^{\,0}_k, \ldots, \vec{x}'^{\,2n+1}_k\right\},$$

and then uses them to obtain the mean and covariance of the prediction,

$$\overline{\vec{x}}'_k = \sum_{i=0}^{2n+1} {}^{m}\omega^{i}_{k-1}\vec{x}'^{\,i}_k, \qquad \text{and} \quad \Sigma'_k = \sum_{i=0}^{2n+1} {}^{c}\omega^{i}_{k-1}\left[\vec{x}'^{\,i}_k - \overline{\vec{x}}'_k\right]\left[\vec{x}'^{\,i}_k - \overline{\vec{x}}'_k\right]^{\top}.$$

In the unscented Kalman filter, we also explicitly calculate the predicted position of the observation, using $h$ to propagate of the state's particles

$$\vec{y}'^{\,i}_k = h\left(\vec{x}'^{\,i}_k, \vec{v}_k\right) \longrightarrow \vec{y}'_k \sim \left\{\vec{y}'^{\,0}_k, \ldots, \vec{y}'^{\,2n+1}_k\right\},$$

and then estimate its mean and covariance

$$\overline{\vec{y}}'_k = \sum_{i=0}^{2n+1} {}^{m}\omega^{i}_{k-1}\vec{y}'^{\,i}_k \qquad \text{and} \quad \Sigma_{yy_k} = \sum_{i=0}^{2n+1} {}^{c}\omega^{i}_{k-1}\left[\vec{y}'^{\,i}_k - \overline{\vec{y}}'_k\right]\left[\vec{y}'^{\,i}_k - \overline{\vec{y}}'_k\right]^{\top}.$$

*The weighting stage* In this stage, the filter computes the cross covariance of the predicted observation and predicted state,

$$\Sigma_{xy_k} = \sum_{i=0}^{2n+1} {}^c\omega_{k-1}^i \left[\vec{x}_k'^i - \overline{\vec{x}}_k'\right] \left[\vec{y}_k'^i - \overline{\vec{y}}_k'\right]^\top ,$$

and then calculates the matrix for final weighting

$$K_k = \Sigma_{xy_k} \Sigma_{yy_k}^{-1} .$$

*The correction stage* Finally, we combine everything together, to find the estimate of mean and covariance

$$\begin{aligned} \overline{\vec{x}}_k &= \overline{\vec{x}}_k' + K_k \left(\vec{y}_k - \overline{\vec{y}}_k'\right), \\ \Sigma_k &= \Sigma_k' + K_k \Sigma_{yy_k} K_k^\top . \end{aligned}$$

The complete algorithm of the unscented Kalman filter is summarized in Algorithm 4. Although the mathematical formulas may look intimidating, the underlying intuition is straightforward: it uses a few carefully placed particles to propagate the Gaussians. The unscented transform is actually more powerful than that, and can give guarantees about the correctness of higher moments even if the random variables were not originally Gaussians [41, 15].

The unscented Kalman filter is a compromise technique. It is not as flexible as a particle filter – it uses a mean and covariance to describe the random variable and can only properly represent unimodal distributions. On the other hand, because the unscented Kalman filter uses the unscented transform to propagate the samples, instead of linearizing the system, its distribution estimates are much more accurate than the extended Kalman filter. Finally, it is much faster than an equivalent particle filter, since the number of particles it requires grows only linearly with the dimension of the system's state vector. The unscented Kalman filter and extended Kalman filter aim at the same niche of applications and have similar computationally complexity. Nevertheless, the unscented Kalman filter is quickly replacing the extended Kalman filter, since it has such superior distribution estimates.

## 5 Final Remarks

Applications that need to estimate real-valued parameters over time usually use a predictive filter. *Tracking* problems are a typical example of these applications. In robotics and computer vision, tracking techniques estimate the unknown value of the system's underlying state. The intrinsic data-fusion properties of predictive filters plays a key role in filling in

**Algorithm 4** Unscented Kalman filter.

**Starts with:** The mean $\overline{\vec{x}}_0$ and covariance matrix $\Sigma_0$ of the prior distribution $\vec{x}_0$ of the system's state.

**Prediction:** Calculate $2n+1$ special particles,

$$\begin{cases} \vec{x}^0_{k-1} = \overline{\vec{x}}_{k-1} \\ \vec{x}^i_{k-1} = \overline{\vec{x}}_{k-1} + \left(\sqrt{(n+\lambda)\,\Sigma_{k-1}}\right)_i & i = 1,\ldots,n \\ \vec{x}^i_{k-1} = \overline{\vec{x}}_{k-1} - \left(\sqrt{(n+\lambda)\,\Sigma_{k-1}}\right)_{i-n} & i = n+1,\ldots,2n. \end{cases}$$

For every particle, calculate two weights (one for mean, and one for covariance estimations),

$$\begin{cases} {}^m\omega^0_{k-1} = \frac{\lambda}{n+\lambda} \\ {}^c\omega^0_{k-1} = \frac{\lambda}{n+\lambda} + \left(1-\alpha^2+\beta\right) \\ {}^m\omega^i_{k-1} = {}^c\omega^i_{k-1} = \frac{\lambda}{2(n+\lambda)} \quad i = 1,\ldots,2n. \end{cases}$$

Finds prediction of state

$$\vec{x}'^i_k = f\left(\vec{x}^i_{k-1}, \vec{w}_{k-1}\right) \longrightarrow \vec{x}'_k \sim \left\{\vec{x}'^0_k, \ldots, \vec{x}'^{2n+1}_k\right\},$$

$$\overline{\vec{x}}'_k = \sum_{i=0}^{2n+1} {}^m\omega^i_{k-1}\vec{x}'^i_k, \qquad \text{and} \quad \Sigma'_k = \sum_{i=0}^{2n+1} {}^c\omega^i_{k-1}\left[\vec{x}'^i_k - \overline{\vec{x}}'_k\right]\left[\vec{x}'^i_k - \overline{\vec{x}}'_k\right]^\top,$$

and then predicted observation

$$\vec{y}'^i_k = h\left(\vec{x}'^i_k, \vec{v}_k\right) \longrightarrow \vec{y}'_k \sim \left\{\vec{y}'^0_k, \ldots, \vec{y}'^{2n+1}_k\right\},$$

$$\overline{\vec{y}}'_k = \sum_{i=0}^{2n+1} {}^m\omega^i_{k-1}\vec{y}'^i_k \qquad \text{and} \quad \Sigma_{yy_k} = \sum_{i=0}^{2n+1} {}^c\omega^i_{k-1}\left[\vec{y}'^i_k - \overline{\vec{y}}'_k\right]\left[\vec{y}'^i_k - \overline{\vec{y}}'_k\right]^\top.$$

**Weighting:** Find out how to best combine prediction and observation at time $k$:

$$\Sigma_{xy_k} = \sum_{i=0}^{2n+1} {}^c\omega^i_{k-1}\left[\vec{x}'^i_k - \overline{\vec{x}}'_k\right]\left[\vec{y}'^i_k - \overline{\vec{y}}'_k\right]^\top, \qquad \text{and} \quad K_k = \Sigma_{xy_k}\Sigma^{-1}_{yy_k}.$$

**Correction:** Finally, estimate the new state's mean and variance

$$\begin{aligned} \overline{\vec{x}}_k &= \overline{\vec{x}}'_k + K_k\left(\vec{y}_k - \overline{\vec{y}}'_k\right), \\ \Sigma_k &= \Sigma'_k + K_k\Sigma_{yy_k}K_k^\top. \end{aligned}$$

the gaps left by incomplete observations, and sometimes even the prediction functions are themselves parameterized by instantaneous measured data [5].

The first few fundamental choices we make to deal with a particular problem are the first step in the direction for its solution: modeling. We need a set of real-valued parameters, the state vector, to describe the system. Additionally, we need a function to describe the change of the state over time, and a function to describe how our observations are related to the state vector. These three issues: state description, state dynamics, and state observation, are not independent among themselves. For example, we can make the state vector more complex, augmenting it with extra information, to simplify the dynamics, as we have seen in Section 2.2.2. The last essential decision about our system is the type of uncertainty we need to represent. We have to decide if our problem can be solved and understood using only unimodal distributions, if we can use a parametric representation of the distribution, etc.

Once we commit to a set of modeling decisions, the rest comes easy, since there is usually one type of predictive filter that is most appropriate for the choices we have already made. Deciding to use the fancy technique $A$ instead of the simple well proven technique $B$ before understanding and modeling the problem is just plain wrong, and unfortunately extremely common among researchers.

As a good rule of thumb, we should always first try to model our system as a linear Gaussian system, and use the Kalman filter. If the results are not satisfactory, and we believe that the problem lies in either the linear dynamics or the Gaussian description of the random variables, we can then proceed to use more complex distribution representation and predictive techniques. This way, we will be able to compare the results of fancier models and techniques against a well-built base case. Simple is good, efficient, and usually enough.

The value used for the prior's distribution is not crucial. As the predictive filter proceeds with its iterations, the state's distribution will eventually converge to its optimal configurations [1]. On the other hand, the correct characterization of the distribution of the observation's noise is essential for the quality and correctness of the final state estimation. In Figure 8, we show an example of how an incorrect observation's distribution can throw off the state estimation stage.

If the choice of the observation's distribution is made arbitrarily, then a predictive filter is nothing more than an *infinite impulse response* [26] low-pass filter. A commonly used approach is to measure this distribution beforehand, in off-line experiments, and then set them as constant in the actual application. A more robust approach is to measure, at every iteration, the observation's distribution properties [9].

Finally, there is a word of advice about particle filters. There is a class of applications, such as robot localization, where particle filters are very robust. Since it uses a particle-based representation of the state's distribution, the filter can consider multiple solutions in

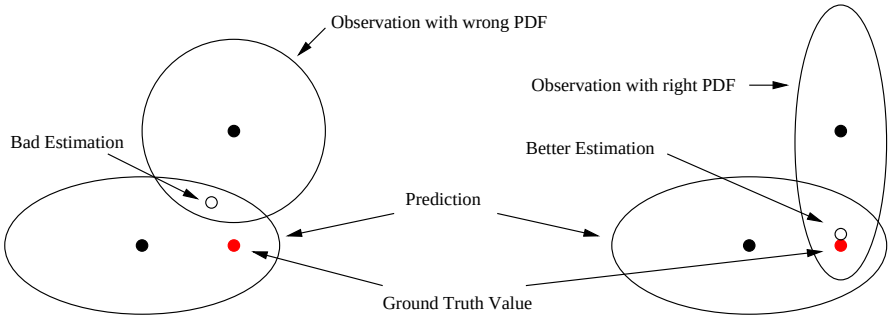


**Figure 8.** The effect of an incorrect observation's distribution in state estimation.

parallel (a multimodal distribution) until there is enough information in the observations to disambiguate them. The million-dollar question is how to extract a meaningful interpretation from the sampled distribution. For example, in a multimodal situation, the mean does not give a meaningful solution to the estimation problem.

**Where to Look for More**

In the past years, there has been a lot of active research in Predictive filters. In this section, we provide a guide to the reader that wants go beyond the vanilla concepts of this introduction.

The correct mathematical modeling of a system is the first step for a successful journey. Both the essential parametrization, the state, as well as its dynamics representation are essential in the areas of control and signal processing [2, 26], as well as machine learning and artificial intelligence [11, 35].

Probabilities [6, 27] and intervals [23, 24] are two common representations of uncertainty. A really good introduction to probability is in the classic Wozencraft's communication book [42]. Monte Carlo methods deal with sampled representations of probabilities and uncertainties_[33] and are close cousins of particle filters.

The reader should feel comfortable with the basic Bayesian concepts common to all predictive filters [38, 10, 7, 19], and Smith and Gelfand's paper "Bayesian statistics without tears: A sampling-resampling perspective" [38] should be a required reading in any engineering school.

Among particular implementations of predictive filters, the Kalman filter should be your starting point [20, 30, 1, 2], and for particle filters [14, 10], there is much to say beyond

the slight flavor we gave here, such as implementation details on how to make them fast [17, 4], and how to deal with the limited number of particles [16]. Unscented Kalman filters are relatively new [41, 15], but are already getting a lot of attention.

There are many applications of predictive filters in different fields and contexts. In in robotics [5, 8], data fusion [4, 29], tracking in computer vision [14, 9, 19], among others. Also, there are many other types of sequential estimation procedures, such as with neural networks [18, 12] and bounded errors [37, 39, 36, 25, 31].

In machine learning, the theory of graphical models unifies a large class of techniques, such as hidden Markov Models (HMM), Markov chains, Bayesian networks, and predictive filters [32, 3, 34, 28]. Finally, there has been a recent special issue about predictive filters on the *Proceedings of the IEEE* [13].

## Acknowledgment

The author has been funded by CNPq, FAPESP, and FAEP-Unicamp.

## References

[1] Gary Bishop and Greg Welch. An introduction to the kalman filter. SIGGRAPH 2001 Course Notes, 2001.

[2] R. Brown and P. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley and Sons, 1997.

[3] E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[4] Y. Chen and Y. Rui. Rea-time speaker tracking using particle filter sensor fusion. *Proceedings of the IEEE*, 92(3):485–494, 2004.

[5] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *CVPR*, 1999.

[6] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume II. John Willey & Sons, 1971.

[7] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[8] N. Freitas, R. Dearden, F. Hutter, R. Morales-Menendez, J. Mutch, and D. Poole. Diagnosis by a waiter and a mars explorer. *Proceedings of the IEEE*, 92(3):455–468, 2004.

[9] Siome Goldenstein, Christian Vogler, and Dimitris Metaxas. 3D facial tracking from corrupted movie sequences. In *CVPR*, pages 880 – 885, 2004.

[10] N. Gordon, D. Salmon, and A. Smith. A novel approach to nonlinear/nongaussian bayesian state estimation. *IEEE Proc. Radar Signal Processing*, (140):107–113, 1993.

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[12] S. Haykin. *Neural Networks: A Comprehensive Foundation*. IEEE Press, 1994.

[13] S. Haykin and N. Freitas. Special issue on sequential state estimation. *Proceedings of the IEEE*, 92(3), 2004.

[14] Michael Isard and Andrew Blake. CONDENSATION: conditional density propagation for visual tracking. *IJCV*, 29(1):5–28, 1998.

[15] SA. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[16] O. King and D. Forsyth. How does CONDENSATION behave with a finite number of samples? In *ECCV*, pages 695–709, 2000.

[17] C. Kwok, D. Fox, and M. Meila. Real-time particle filters. *Proceedings of the IEEE*, 92(3):469–481, 2004.

[18] J. Lo and L. Yu. Recursive neural filters and dynamical range transformers. *Proceedings of the IEEE*, 92(3):514–535, 2004.

[19] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Spriger, 2004.

[20] P. Maybeck. *Stochastic Models, Estimation, and Control*. Academic Press, 1979.

[21] G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley-Interscience, 2000.

[22] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, $2^{nd}$ edition, 1994.

[23] R. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[24] R. Moore. *Methods and Applications of Interval Analysis*. SIAM, 1979.

[25] D. Morrell and W. Stirling. Set-valued filtering and smoothing. *IEEE Transactions on Systems, Man. and Cybernetics*, 21(1):184–193, 1991.

[26] K. Ogata. *Modern Control Engineering*. Prentice Hall, 1990.

[27] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 1991.

[28] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[29] P. Perez, J. Vermaak, and A. Blake. Data fusion for visual tracking with particle. *Proceedings of the IEEE*, 92(3):495–513, 2004.

[30] R. Plessis. Poor man's explanation of Kalman filtering or how I stopped worrying and learned to love matrix inversion. Technical report, 1967.

[31] L. Pronzato and E. Walter. Minimal volume ellipsoids. *International Journal of Adaptive Control and Signal Processing*, 8:15–30, 1994.

[32] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[33] C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 1999.

[34] S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11(2):305–345, 1999.

[35] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2003.

[36] A. Sabater and F. Thomas. Set membership approach to the propagation of uncertain geometric information. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2718–2722, 1991.

[37] F. Schweppe. Recursive state estimation: Unknown but bounded errors and system inputs. *IEEE Transactions on Automatic Control*, 13(1):22–28, 1968.

[38] A.F.M Smith and A.E. Gelfand. Bayesian statistics without tears: A sampling-resampling perspective. *American Statistician*, 46(2):84–88, 1992.

[39] W. Stirling and D. Morrell. Convex bayes theory. *IEEE Transactions on Systems, Man. and Cybernetics*, 21(1):173–183, 1991.

[40] J. Stolfi and L. Figueiredo. *Self-Validated Numerical Methods and Applications*. 21º Colóquio Brasileiro de Matemática, IMPA, 1997.

[41] E. A. Wan and R. van der Merwe. *Kalman Filtering and Neural Networks*, chapter Chapter 7 : The Unscented Kalman Filter, (50 pages). Wiley Publishing, 2001.

[42] J. Wozencraft and I. Jacobs. *Principles of Communication Engineering*. Waveland Press, 1965.