Polynomial Chaos Theory for Performance Evaluation of ATR Systems

Michael D. DeVore and Alec J. Bateman

Barron Associates, Inc., 1410 Sachem Place Suite 202, Charlottesville, VA

ABSTRACT

The development of a more unified theory of automatic target recognition (ATR) has received considerable attention over the last several years from individual researchers, working groups, and workshops. One of the major benefits expected to accrue from such a theory is an ability to analytically derive performance metrics that accurately predict real-world behavior. Numerous sources of uncertainty affect the actual performance of an ATR system, so direct calculation has been limited in practice to a few special cases because of the practical difficulties of manipulating arbitrary probability distributions over high dimensional spaces. This paper introduces an alternative approach for evaluating ATR performance based on a generalization of Norbert Wiener's polynomial chaos theory. Through this theory, random quantities are expressed not in terms of joint distribution functions but as convergent orthogonal series over a shared random basis. This form can be used to represent any finite-variance distribution and can greatly simplify the propagation of uncertainties through complex systems and algorithms. The paper presents an overview of the relevant theory and, as an example application, a discussion of how it can be applied to model the distribution of position errors from target tracking algorithms.

Keywords: generalized polynomial chaos; performance prediction; ATR theory; tracking algorithms

1. INTRODUCTION

The development of a more unified theory of assisted/automatic target recognition (ATR) has received considerable attention over the last several years from individual researchers, working groups, and workshops. One of the major benefits expected to accrue from such a theory is an ability to analytically derive performance metrics that accurately predict real-world behavior. An understanding of ATR system performance is necessary long before the system is fielded. An ability to determine the expected performance of a system early in the design process can facilitate a number of important design activities, including:

- guiding the use of simplifying assumptions and approximations that make the algorithms tractable;
- determining an appropriate quantity of training data;
- selecting appropriate values of tunable parameters;
- understanding sensitivity to operating conditions; and
- developing a concept of operations for integrated systems.

When referring to ATR systems, the term "performance" is generally taken to be relative to a particular set of operating conditions. The operating conditions determine the distribution of data produced by the sensor platform, which are inputs to the ATR software. The performance of ATR systems are generally measured using testing data that are representative of what is expected in operation. However, this testing is generally limited in scope because of the expense associated with large-scale data collection and a fundamental inability to cover all possible operational scenarios.

To complement direct performance measurement from test data, many authors adopt a model-based approach to ATR evaluation to more completely characterize the range of system behaviors. Broadly, performance modeling for ATR systems involves two distinct activities:

1. characterizing distributions of sensor data that closely approximate a wide range of possible operational scenarios; and

E-mail: {devore,bateman}@barron-associates.com, Telephone: (434) 973-1215

2. characterizing the ATR system performance that arises from these distributions.

The focus of this paper is on the second of these, to seek methods by which ATR performance can be evaluated relative to arbitrary distributions of sensor data. Currently, this is most frequently performed in a Monte Carlo fashion, in which the ATR algorithm is presented with simulated sensor data. There are some potential drawbacks to this approach, notably that estimating the probabilities of rare (but potentially disastrous) misclassifications often requires a very large number of simulations.

In contrast to simulation-based evaluation, this paper presents some initial theory development in the area of analytical ATR performance evaluation. The goal is ultimately to produce a complete characterization of an ATR system's behavior with respect to any posited distribution of sensor data. Theoretically, a probability distribution characterizing the ATR system's outputs could be computed directly from the sensor data distribution. However, analytical derivations of performance metrics have historically been limited to a relatively few special cases because of the practical difficulties of manipulating arbitrary probability distributions over high dimensional spaces. These special cases typically involve either restricting the sensor data distributions to a family (such as the Gaussian distributions) that can be conveniently analyzed or employing performance bounds (such as the Chernoff bound) that require only a partial characterization of the system output.

This paper introduces an alternative approach for evaluating ATR performance based on a generalization of polynomial chaos theory, originally developed by Norbert Wiener in 1938.[1] Through this theory, random quantities are expressed not in terms of joint distribution functions but as convergent orthogonal series over a shared random basis. Provided that all variances are finite, arbitrary multivariate probability distributions can be represented in polynomial chaos form through a countable set of coefficients, which constitute a transform-space representation of the distributions. Many common operations are conveniently performed on polynomial chaos expressions, which has been shown to greatly simplify the propagation of uncertainties through many complex systems and algorithms of practical interest. For example, many authors have noted their their usefulness in finding solutions for stochastic differential equations, both ordinary and partial[2–8] in a wide variety of application areas. Simplifications achievable through polynomial chaos representations have led to their adoption in modeling highly complex systems within a wide variety of problem domains. For example, they have been applied to dynamic control systems that manage system and signal uncertainty[9] and to the design of data pathways customized to the dynamic range of nonlinear systems.[10] A significant body of literature documents their use in eliminating the need for time-consuming Monte Carlo simulation of highly complex systems, for example in analysis of nonlinear aeroelastic structures[11] and computational fluid dynamics.[12–14]

This paper presents an overview of the relevant theory and describes an example application for characterizing distribution of tracking error in Kalman Filter algorithms. Section 2 offers a brief review of common methods for characterizing random variables, contrasts them to the polynomial chaos approach, and presents a high-level discussion of the use of polynomial chaos in ATR performance modeling. Section 3 contains a more detailed look at polynomial chaos theory and demonstrates how common operations can be performed on random variables using this theory. Section 4 demonstrates the application of polynomial chaos methods for modeling the state estimation error for a basic Kalman filter algorithm for arbitrary state/measurement distributions. An example of how these results could be applied to a simple target tracking algorithm is contained in Section 5. Finally, a summary of the approach is contained in Section 6.

2. PROBABILISTIC PERFORMANCE CHARACTERIZATION

In general, ATR algorithms classify and/or estimate the state of objects in a scene by computing some function of the available observed data. (This may be a random function for non-deterministic algorithms, a distinction which is not relevant for this discussion.) The observed data, which we will denote by the random vector $\boldsymbol{S}$, may represent an arbitrary combination of sensor measurements and side information pertaining to the scene. The function, which we will denote by the random vector $\boldsymbol{T}(\boldsymbol{S})$, represents the computation involved in transforming the raw observed data into a quantity from which an inference directly follows. For example, in a classifier $\boldsymbol{T}(\boldsymbol{S})$ may represent a vector of test statistics (e.g. likelihood values or corresponding sufficient statistics), whereas in an estimation problem it typically represents the estimator itself. In the restricted context of this paper, performance characterization is the process of determining the distribution of $\boldsymbol{T}$ that arises from an arbitrary distribution posited for $\boldsymbol{S}$. Finding useful distributions to posit for $\boldsymbol{S}$ is a matter of scenario and sensor modeling, which are beyond the scope of this paper.

More formally, we seek methods that can be used to determine a *complete characterization* of $\boldsymbol{T}$ given a complete characterization of $\boldsymbol{S}$. Any information that is sufficient to determine the probability $\Pr[\boldsymbol{S} \in \mathcal{S}]$ for any measurable set $\mathcal{S}$ is said to be a complete characterization of $\boldsymbol{S}$. Complete characterizations of arbitrary random vectors are typically expressed in terms of the joint probability density function (PDF) $f_{\boldsymbol{S}}(\cdot)$ of its components. Common alternatives include the joint cumulative distribution function (CDF) of its components and the characteristic function (CF), the multi-dimensional Fourier transform of its joint PDF. Theoretically, any of these could be used along with the functional form of $\boldsymbol{T}(\cdot)$ to determine a complete characterization of $\boldsymbol{T}$, such as its joint PDF, etc. In practice, however, this is can be quite difficult because the computations involved are often unwieldy. Moreover, it is a challenging problem to simply represent arbitrary PDFs in such a way that they are guaranteed to remain valid (i.e. are everywhere non-negative and integrate to unity). It is similarly difficult to represent arbitrary CDFs and CFs.

Polynomial chaos theory takes an alternative approach to characterizing random quantities, representing more closely how the quantity arises from a random experiment, in this case the production of sensor data $\boldsymbol{S}$. In probability theory, all random quantities are ultimately defined in terms of an underlying *probability space* that summarizes the random experiment that gives rise to all observed variability. Formally, a probability space is a triple $(\Omega, \mathcal{F}_\Omega, \mathcal{P}_\Omega)$, where

- Ω is the set of possible outcomes of the random experiment;
- $\mathcal{F}_\Omega$ is a set of events (subsets of Ω) to which meaningful probabilities can be assigned; and
- $\mathcal{P}_\Omega$ is a function that assigns a probability to each element of $\mathcal{F}_\Omega$.

The random n-vector $\boldsymbol{S}$ is formally defined to be a vector-valued function (technically, a $\mathcal{P}_\Omega$-measurable function) from Ω to the set of real vectors $\Re^n$. That is, $\boldsymbol{S}(\omega)$ is a vector whose value is random because the random experiment resulted in outcome $\omega \in \Omega$. Similarly, the ATR output $\boldsymbol{T}(\boldsymbol{S}(\omega))$ is a function defined over Ω.

It is always possible to define a random d-vector $\boldsymbol{Z}(\omega)$ with independent components and a corresponding vector function $\tilde{\boldsymbol{S}}(\boldsymbol{Z})$ such that $\tilde{\boldsymbol{S}}(\boldsymbol{Z}(\omega)) = \boldsymbol{S}(\omega)$. Because of this equivalence, it is common to abuse notation slightly and simply write $\boldsymbol{S}(\boldsymbol{Z})$, a practice adopted through the remainder of this paper. The individual components of $\boldsymbol{Z}$ can be thought of as abstract (or modeled) sources of variability governing the random experiment. They are a numerical representation of whatever physical processes govern the outcome ω and are often referred to as the random basis for $\boldsymbol{S}$. The function $\boldsymbol{Z}$ induces a secondary probability space $(\Re^d, \mathcal{F}_Z, \mathcal{P}_Z)$, where $\mathcal{F}_Z$ contains the Borel sets of $\Re^\nu$, and $\mathcal{P}_Z$ is the probability measure for $\boldsymbol{Z}$.

For a given model of the sources of randomness $\boldsymbol{Z}$, the function $\boldsymbol{S}(\cdot)$ is a complete characterization of $\boldsymbol{S}$. In polynomial chaos theory, this function is represented as a generalized Fourier series in which the basis functions are an orthonormal family of polynomials. The components of $\boldsymbol{Z}$ have a PDF matched to the weighting function for these polynomials, and these are generally chosen to have a convenient form (e.g. Gaussian, etc.). This approach has a number of practical advantages. First, there are a countable number of coefficients in this Fourier series, and they constitute a complete characterization of $\boldsymbol{S}$. Second, there are no constraints on the values of these coefficients, so the series always represents a valid random variable, even when the coefficients are subject to numerical roundoff, etc. In particular, the set of coefficients can be truncated to any desired finite number, resulting in an approximation of the actual distribution that meets available computational capabilities. Finally, as will be demonstrated in the next section, many common operations on random variables (e.g. arithmetic and other functions) can be expressed as corresponding operations on these coefficients.

These properties make the polynomial chaos representation very well suited to performance modeling. The approach illustrated in this paper can be summarized as follows:

1. Identify the sources of random variation that affect the production of sensor data $\boldsymbol{S}$;
2. Define a random basis $\boldsymbol{Z}$ such that each component is associated with one of the identified sources of variation;
3. Express the sensor data $\boldsymbol{S}$ as a function of $\boldsymbol{Z}$ and compute its polynomial chaos representation;
4. Calculate the polynomial chaos representation of $\boldsymbol{T}$ by applying the ATR algorithm to $\boldsymbol{S}(\boldsymbol{Z})$, using the rules for propagating polynomial chaos coefficients through the various operations of $\boldsymbol{T}(\cdot)$;
5. Express the performance metric of interest as a function of $\boldsymbol{T}$, and compute its polynomial chaos representation.

The result of Step 5 above will be a complete characterization of the ATR algorithm performance relative to any distribution posited for $\boldsymbol{S}$. This characterization can be used to compute any statistic or probability of interest.

3. GENERALIZED POLYNOMIAL CHAOS

In polynomial chaos theory, a random vector $\boldsymbol{Y}$ is expressed as a transformation of a random basis $\boldsymbol{Z}$, a d-vector with statistically independent components. To produce $\boldsymbol{Y}$, the random basis is operated on by a series of polynomial functions $\psi_{n_1,n_2,\ldots,n_\nu}(\cdot)$ which are linearly combined as

$$\boldsymbol{Y}(Z) = \sum_{k_1=0}^{\infty}\sum_{k_2=0}^{\infty}\cdots\sum_{k_d=0}^{\infty} \boldsymbol{y}_{k_1,k_2,\ldots,k_d}\psi_{k_1,k_2,\ldots,k_d}(\boldsymbol{Z}). \tag{1}$$

The coefficients $\boldsymbol{y}_{n_1,n_2,\ldots,n_d}$ are deterministic vectors the same size as $\boldsymbol{Y}$, and they provide the degrees of freedom that allow $\boldsymbol{Y}$ to assume any distribution with finite variance. The coefficients can be found by projecting $\boldsymbol{Y}(Z)$ onto the individual basis functions, as

$$\boldsymbol{y}_{k_1,k_2,\ldots,k_d} = \mathrm{E}\left[\boldsymbol{Y}(Z)\psi_{k_1,k_2,\ldots,k_d}(\boldsymbol{Z})\right], \tag{2}$$

where the expectation is taken over all components of $\boldsymbol{Z}$. In these expressions, the polynomial $\psi_{k_1,k_2,\ldots,k_d}(\cdot)$ is of order k_1 in Z_1 (i.e. has terms involving $Z_1^{k_1}$ but not any larger exponents), of order k_2 in Z_2, etc. To simplify notation, the terms of the series are frequently reindexed by replacing the multiple index $k_1,\ldots,k_d$ with the single index k. (That is, k is a function of $k_1,\ldots,k_d$, and given k one can recover the $k_1,\ldots,k_d$.) In this simplified notation, the function $\psi_{k_1,k_2,\ldots,k_d}(\boldsymbol{Z})$ is written as $\psi_k^{(d)}(\boldsymbol{Z})$, and the series becomes $\boldsymbol{Y} = \sum_{k=0}^{\infty}\boldsymbol{y}_k\psi_k^{(d)}(\boldsymbol{Z})$. As a practical matter, the series are truncated to some finite number of terms

$$\begin{aligned}\boldsymbol{Y} \approx \hat{Y} &= \sum_{k_1=0}^{K-1}\sum_{k_2=0}^{K-1}\cdots\sum_{k_d=0}^{K-1} \boldsymbol{y}_{k_1,k_2,\ldots,k_d}\psi_{k_1,k_2,\ldots,k_d}(\boldsymbol{Z}) \\ &= \sum_{k=0}^{K^d}\boldsymbol{y}_k\psi_k^{(d)}(\boldsymbol{Z}).\end{aligned}$$

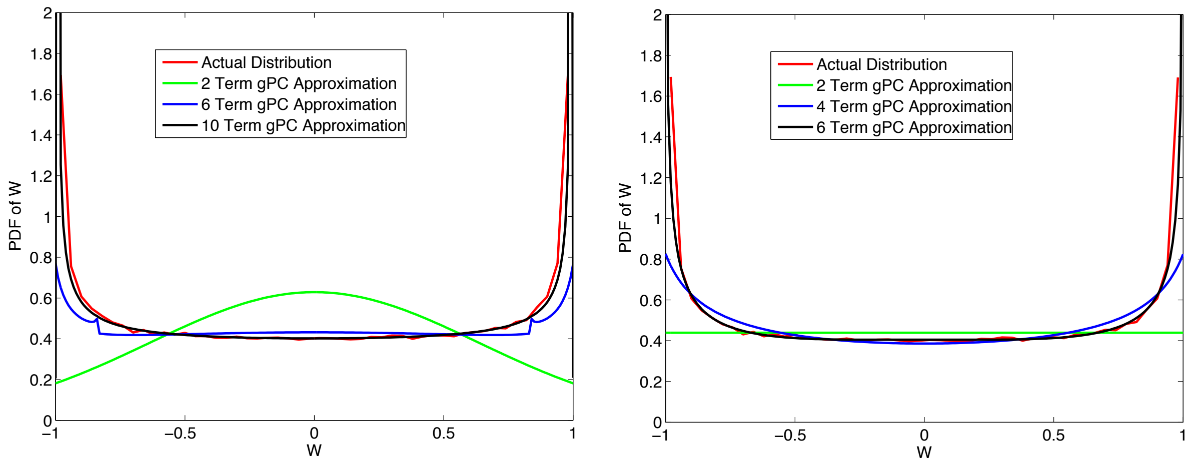
The collection of functions $\{\psi_k\}_{k=0}^{\infty}$ form an orthonormal set relative to a weighting function which is the probability density of $\boldsymbol{Z}$. This choice leads to the orthonormality relation

$$\mathrm{E}[\psi_j^{(d)}(\boldsymbol{Z})\psi_k^{(d)}(\boldsymbol{Z})] = \begin{cases} 0 & \text{if } j \neq k \\ 1 & \text{if } j = k \end{cases}, \tag{3}$$

which in turn leads to great simplification in many operations involving $\boldsymbol{Y}$.

The representation can be used for random vectors, matrices, and processes in an analogous way. For example, an arbitrary scalar random process $W(t)$ with finite autocovariance can be represented as $W(t) = \sum_{k=1}^{\infty} w_k(t)\psi_k^{(d)}(\boldsymbol{Z})$, where the deterministic functions $w_k(t) = \mathrm{E}[W(t)\psi_k^{(d)}]$. This allows a consistent representation for all random quantities in the problem domain. In Wiener's original formulation, the components of $\boldsymbol{Z}$ are zero-mean Gaussian random variables, and the functions $\psi_k^{(d)}(\cdot)$ are the Hermite polynomials over $\Re^d$. The technique has since been generalized, and many alternative pairings have been investigated in detail.[3] For example, gamma distributed $\boldsymbol{Z}$ with the Laguerre polynomial family exhibits the desired orthogonality property, as does a uniform distributed $\boldsymbol{Z}$ with the Legendre polynomial family, etc. The theory extended to arbitrary polynomial families is referred to as *generalized polynomial chaos* (gPC).

As an example, consider a gPC representation of the random quantity $W = \sin(Q)$, where Q is a standard Gaussian random variable. We can define a one-dimensional random basis Z and express Q as a function of Z such that $Q(Z)$ is a standard Gaussian random variable. For example, we could define $Q(Z) = F_Q^{-1}(F_Z(Z))$, where F_Q^{-1} is the inverse CDF of Q (i.e. of a standard Gaussian random variable), and F_Z is the CDF of Z. (The random quantity $F_Z(Z)$ is known as the *probability integral transform* of Z and yields a uniformly distributed quantity, which is transformed to the distribution of Q by the subsequent *inverse probability integral transform*, F_Q^{-1}.) An infinite variety of alternative mappings are possible. Also, the distribution of Z is quite arbitrary, as long as it is continuous. While it may appear most natural to let Z follow a Gaussian distribution, other distribution choices may be preferable because of their ability to more efficiently represent related variables (such as W), as demonstrated below. The choice of distribution for Z is thus an engineering decision to balance representation efficiency among the random variables within a system.



Gaussian Basis Uniform Basis

Figure 1. Convergence of gPC series with increasing numbers of coefficients for a Gaussian basis (left) and a uniform basis (right).

The gPC expansion of Q is $Q(Z) = \sum_{k=1}^{\infty} q_k \psi_k^{(1)}(Z)$, where $q_k = \mathrm{E}[Q(Z)\psi_k^{(1)}(Z)]$, and the $\psi_k^{(1)}$ form a complete orthonormal family of polynomials relative to a weighting function equal to the PDF of Z. For example, if Z is chosen to follow a Gaussian distribution, then $\psi_k^{(1)}$ is the kth normalized Hermite polynomial. If Z is chosen to follow a uniform distribution, then $\psi_k^{(1)}$ is the kth normalized Legendre polynomial. The random variable W is, indirectly, a function of Z, and it has a gPC expansion whose coefficients can be computed from the gPC coefficients of Q. The series coefficients can be truncated to any finite number to facilitate practical numerical approximations. For example, the left panel of Figure 1 shows the PDFs of W that result from a sequence of approximations with 2, 6, and and 10 coefficients when the basis Z is chosen to follow a Gaussian distribution. The actual PDF of W is shown in red, and the plots suggest that the gPC expansion of W does indeed converge as the number of coefficients increases. The right panel of that figure shows a similar result when the basis is chosen to follow a uniform distribution. Note that convergence is much faster with this choice of basis distribution. In general, though any continuous basis distribution will result in convergence, the corresponding rates of convergence will vary.

Manipulation of random quantities (variables, vectors, processes) in gPC form reduces to processing of the deterministic coefficients. For example, suppose that X and Y are random variables with polynomial chaos expansions

$$X = \sum_{k=0}^{K^d} x_k \psi_k^{(d)}(\boldsymbol{Z}) \text{ and } Y = \sum_{k=0}^{K^d} y_k \psi_k^{(d)}(\boldsymbol{Z}).$$

Their sum is a random variable $V = X + Y$ with gPC representation

$$\begin{aligned} V &= \sum_{k=0}^{K^d} x_k \psi_k^{(d)}(\boldsymbol{Z}) + \sum_{k=0}^{K^d} y_k \psi_k^{(d)}(\boldsymbol{Z}) \\ &= \sum_{k=0}^{K^d} (x_k + y_k) \psi_k^{(d)}(\boldsymbol{Z}) \\ &= \sum_{k=0}^{K^d} v_k \psi_k^{(d)}(\boldsymbol{Z}). \end{aligned}$$

Thus, the gPC coefficients of a sum equal the sum of corresponding gPC coefficients. Similarly, the product $W = XY$ has a simple gPC representation, which can be found as

$$W = \sum_{k=0}^{K^d} w_k \psi_k^{(d)}(\boldsymbol{Z}) = \sum_{m=0}^{K^d} \sum_{n=0}^{K^d} x_m y_n \psi_m^{(d)}(\boldsymbol{Z}) \psi_n^{(d)}(\boldsymbol{Z}).$$

To find the coefficients w_k, multiply both sides by $\psi_j^{(d)}(\boldsymbol{Z})$ and take expectations, noting that $\mathrm{E}[\psi_j^{(d)}(\boldsymbol{Z})\psi_k^{(d)}(\boldsymbol{Z})] = 0$ unless $j = k$. Because of this, the sum on the left yields a single nonzero term,

$$w_k = \sum_{m=0}^{K^d} \sum_{n=0}^{K^d} x_m y_n \mathrm{E}[\psi_m^{(d)}(\boldsymbol{Z})\psi_n^{(d)}(\boldsymbol{Z})\psi_k^{(d)}(\boldsymbol{Z})],$$

where the expectation on the right is a constant that can be computed offline. These results can be used to compute gPC representations for arbitrary polynomial functions of random variables in gPC form. Transcendental functions (e.g. trigonometric, logarithm, exponential, etc.) can be approximated by polynomials through their Taylor series expansion. More complicated approaches are also available, for a comparison see Ref. [15].

In the above examples, the random variables W and V are completely defined by their deterministic coefficients w_k and v_k, respectively. In particular, note that in the solutions above there is no requirement that X and Y be independent – the solution is valid regardless of their dependence structure. Their dependence (or lack of it) is represented implicitly in the deterministic quantities x_k and y_k, and is propagated naturally through to W and V. The gPC characterization is complete, for example the probability that V takes a value in some set is equal to the probability that $\boldsymbol{Z}$ takes a value in the pre-image of that set under the function $V(\boldsymbol{Z})$. That is, $\Pr[V \in A] = \Pr[\boldsymbol{Z} \in V^{-1}(A)]$. A limiting form as A shrinks to a point can be used to evaluate the PDF of V.

A number of common statistical procedures are readily performed using gPC expressions. Joint simulation of multiple random variables is especially simple. For example, pairs (V, W) from above can be simulated by generating random variates from the distribution of $\boldsymbol{Z}$ (which is simple because the components of $\boldsymbol{Z}$ are independent) and plugging these into the gPC expansions for V and W. Moments of random variables are also very simple to compute, for example if r is a positive integer, then

$$\begin{aligned}
\mathrm{E}[W^r] &= \mathrm{E}\left[\left(\sum_{k=0}^{K^d} w_k \psi_k^{(d)}(\boldsymbol{Z})\right)^r\right] \\
&= \sum_{k_1=0}^{K^d} \sum_{k_2=0}^{K^d} \cdots \sum_{k_r=0}^{K^d} w_{k_1} w_{k_2} \cdots w_{k_r} \mathrm{E}\left[\psi_{k_1}^{(d)}(\boldsymbol{Z})\psi_{k_2}^{(d)}(\boldsymbol{Z}) \cdots \psi_{k_r}^{(d)}(\boldsymbol{Z})\right].
\end{aligned}$$

Expected values in the expression above are often easily evaluated and can be computed offline if necessary. With $r = 1$ it can easily be shown that $\mathrm{E}[W] = w_0$, and with $r = 2$ it can be shown that $\mathrm{var}(W) = \sum_{k=1}^{K^d} w_k^2$, regardless of the distribution of $\boldsymbol{Z}$.

4. TRACKING ALGORITHM ANALYSIS

Real-time tracking algorithms provide continual estimates of the state (position and motion) of target objects based on an input stream of sensor data. Because it is a function of the sensor data, the estimated state at each point in time is a random quantity, and tracking system performance can be characterized as the distribution of the difference between the actual and estimated states. Calculation of this distribution is very difficult except in special cases. This section contains an aplication of the use of gPC theory to determine a closed-form representation of tracking error when there is a mismatch between the modeled and actual system. This analysis is applied jointly to the target motion and to the tracking algorithm that responds to that motion.

4.1 Kalman Filter

Perhaps the most common tracking algorithms are variants of the basic Kalman filter, including the extended Kalman filter and the unscented Kalman filter. The Kalman filter is based on an assumption that target state $\boldsymbol{X}[i]$ evolves linearly between sensor measurements $\boldsymbol{Y}[i-1]$ and $\boldsymbol{Y}[i]$, and that it is further subjected to additive independent Gaussian noise. It is further assumed that the measurement process yields a value of $\boldsymbol{Y}[i]$ that is a linear function of the target's current state, again with additive independent Gaussian noise. This is expressed through the state-output equations

$$\boldsymbol{X}[i] = A\boldsymbol{X}[i-1] + \boldsymbol{W}[i] \tag{4}$$

$$\boldsymbol{Y}[i] = C\boldsymbol{X}[i] + \boldsymbol{V}[i], \tag{5}$$

where $\boldsymbol{W}[i]$ and $\boldsymbol{V}[i]$ are additive noise terms effecting the state transition and output, respectively, at stage i. Let their mean vectors be denoted $\boldsymbol{\mu}_W$ and $\boldsymbol{\mu}_V$, respectively, and let their covariance matrices be denoted Σ_W and Σ_V. A basic Kalman filter-based tracking algorithm begins with a mean vector $\hat{\boldsymbol{\mu}}_X[0]$ and covariance matrix $\hat{\Sigma}_X[0]$ describing the initial target state, then iteratively performs the following three-step process:

1. Compute the prior mean and covariance estimates of the current output $\boldsymbol{Y}[i]$ based on the step-ahead prediction of state produced at the prior iteration:

$$\hat{\boldsymbol{\mu}}_Y[i] = \tilde{C}\hat{\boldsymbol{\mu}}_X[i] + \tilde{\boldsymbol{\mu}}_V \quad (6)$$
$$\hat{\Sigma}_Y[i] = \tilde{C}\hat{\Sigma}_X[i]\tilde{C}^T + \tilde{\Sigma}_V \quad (7)$$

   In these equations, a hat above a quantity indexed by i (as in $\hat{\boldsymbol{\mu}}_X[i]$) denotes an estimate based on measurements obtained prior to stage i (i.e. on measurements $\boldsymbol{y}[1], \boldsymbol{y}[2], \ldots, \boldsymbol{y}[i-1]$). The same quantity without a hat (as in $\boldsymbol{\mu}_X[i]$) denotes an estimate based on all measurements up to and including stage i. A tilde above a quantity (e.g. $\tilde{C}$) is used to emphasize that it is a value *assumed* by the tracking algorithm and does not necessarily match the actual state evolution and measurement process.

2. Compute the posterior mean and covariance estimates of the current state $\boldsymbol{X}[i]$ based on the most recent measurement, $\boldsymbol{Y}[i]$:

$$\boldsymbol{\mu}_X[i] = \hat{\boldsymbol{\mu}}_X[i] + G[i]\left(\boldsymbol{Y}[i] - \hat{\boldsymbol{\mu}}_Y[i]\right) \quad (8)$$
$$\Sigma_X[i] = \left(I - G[i]\tilde{C}\right)\hat{\Sigma}_X[i] \quad (9)$$

   where $G[i]$ is the Kalman gain at stage k. For an optimal Kalman filter,[16]

$$G[i] = \hat{\Sigma}_X[i]\tilde{C}^T\left(\tilde{\Sigma}_V + \tilde{C}\hat{\Sigma}_X[i]\tilde{C}^T\right)^{-1}. \quad (10)$$

3. Compute the prior mean and covariance estimates of the next state $\boldsymbol{X}[i+1]$:

$$\hat{\boldsymbol{\mu}}_X[i+1] = \tilde{A}\boldsymbol{\mu}_X[i] + \tilde{\boldsymbol{\mu}}_W \quad (11)$$
$$\hat{\Sigma}_X[i+1] = \tilde{A}\Sigma_X[i]\tilde{A}^T + \tilde{\Sigma}_W \quad (12)$$

The value $\boldsymbol{\mu}_X[i]$ represents the best estimate of target state after considering all sensor measurements up to and including $\boldsymbol{Y}[i]$. Note that the tracking algorithm requires knowledge of the matrices A and C as well as the mean vector and covariance matrix of the additive noise terms. If these are known exactly and the assumptions regarding linear state evolution, linear measurements, and Gaussian noise are all accurate, then the tracking error $\boldsymbol{e}[i] = \boldsymbol{X}[i] - \boldsymbol{\mu}_X[i]$ will be a Gaussian random vector with mean zero and covariance matrix $\Sigma_X[i]$. In practice, however, these assumptions will be inaccurate and the matrices A, etc. will at best be rough approximations. As a result, the tracking error will follow some other distribution. Note that, depending on the nature of the difference between the actual and assumed target behavior, the actual tracking error may be greater than or less than the idealized error.

4.2 Kalman Filter Performance

To develop a gPC analysis of the actual error, we begin with Eqns. (4) and (5). The matrix A describes the motion dynamics of a target, in particular how the position, velocity, and other state variables evolve from one sensor measurement to another. In general this may exhibit variation between vehicles and may vary for a single vehicle depending on the environment. As a result, it may be useful to think of this matrix as random across the set of possible targets and to express it as a gPC quantity $A = \sum_{k=0}^{K_d} A_k \psi_k^{(d)}(\boldsymbol{Z})$. The matrix C describes the combined behavior of the sensor and target detection algorithm. This behavior too may vary from one target to another, for example because of variations in appearance, and may vary for a single target over time, for example because of changes in sensor-target geometry. Thus, it may be useful to think of this matrix as random and to model it as a gPC quantity $C = \sum_{k=0}^{K_d} C_k \psi_k^{(d)}(\boldsymbol{Z})$. Finally, the initial target state

$\boldsymbol{X}[0]$ and the noise terms $\boldsymbol{W}[i]$ and $\boldsymbol{V}[i]$ are random and can be represented in gPC form, with gPC coefficients $\boldsymbol{x}_k[0]$, $\boldsymbol{w}_k[i]$, and $\boldsymbol{v}_k[i]$, respectively. Substituting these gPC representations in (4) yields

$$\sum_{k=0}^{K_d} \boldsymbol{x}_k[i]\psi_k^{(d)}(\boldsymbol{Z}) = \left(\sum_{m=0}^{K_d} A_m \psi_m^{(d)}(\boldsymbol{Z})\right)\left(\sum_{n=0}^{K_d} \boldsymbol{x}_n[i-1]\psi_n^{(d)}(\boldsymbol{Z})\right) + \sum_{k=0}^{K_d} \boldsymbol{w}_k[i]\psi_k^{(d)}(\boldsymbol{Z}).$$

Applying the gPC addition and multiplication results of Section 3, we have that

$$\begin{aligned} \boldsymbol{X}[i] &= \sum_{k=0}^{K_d} \boldsymbol{x}_k[i]\psi_k^{(d)}(\boldsymbol{Z}), \quad \text{where} \\ \boldsymbol{x}_k[i] &= \sum_{m=0}^{K_d}\sum_{n=0}^{K_d} A_m \boldsymbol{x}_n[i-1]\mathrm{E}\left[\psi_m^{(d)}(\boldsymbol{Z})\psi_n^{(d)}(\boldsymbol{Z})\psi_k^{(d)}(\boldsymbol{Z})\right] + \boldsymbol{w}_k[i]. \end{aligned} \tag{13}$$

Similarly, substituting gPC expressions in (5) yields

$$\begin{aligned} \boldsymbol{Y}[i] &= \sum_{k=0}^{K_d} \boldsymbol{y}_k[i]\psi_k^{(d)}(\boldsymbol{Z}), \quad \text{where} \\ \boldsymbol{y}_k[i] &= \sum_{m=0}^{K_d}\sum_{n=0}^{K_d} C_m \boldsymbol{x}_n[i]\mathrm{E}\left[\psi_m^{(d)}(\boldsymbol{Z})\psi_n^{(d)}(\boldsymbol{Z})\psi_k^{(d)}(\boldsymbol{Z})\right] + \boldsymbol{v}_k[i]. \end{aligned} \tag{14}$$

Note that the gPC representation can also address cases in which the matrices A and C could be made time varying (i.e. are functions of i). It can also address cases in which the evolution of state and subsequent mapping to output observation are non-linear, as discussed in Section 3.

We next consider the response of the tracking algorithm to the random sensor sequence characterized by (14). This sequence enters the Kalman filter algorithm in Step 2, Eqn. (8). As a result, the estimated state $\boldsymbol{\mu}_X[i]$ is a random quantity. The current estimated state is used in the state prediction of Step 3, Eqn. (11), so the terms $\hat{\boldsymbol{\mu}}_X[i]$ are random. Finally, the prior estimate of sensor output in Step 1, Eqn. (6) is a function of the state prediction, so the terms $\hat{\boldsymbol{\mu}}_Y[i]$ are random. Each of these quantities can be represented in gPC form over the random basis $\boldsymbol{Z}$. The remaining quantities are not random, either because they have fixed values selected by the algorithm designer (such as $\tilde{C}$) or because they follow directly from the assumed initial state covariance matrix $\hat{\Sigma}_X[0]$. Finally, note that the error sequence $\boldsymbol{e}[i]$ is the difference between $\boldsymbol{X}[i]$ in (13) and $\boldsymbol{\mu}_X[i]$. The distribution of this difference can be found in gPC form for all i through the following iterative algorithm. Note that in the equations given, quantities with a gPC expansion are underlined for clarity.

Algorithm. To compute the distribution of tracking error at each stage i in response to any posited state sequence distribution $\boldsymbol{X}[i]$ from (13), iterate the following steps for each $i = 0, 1, \ldots$:

1. Determine a gPC representation characterizing the prior mean vector for sensor output at stage i as it will be calculated by the Kalman filter:
$$\underline{\hat{\boldsymbol{\mu}}_Y[i]} = \tilde{C}\underline{\hat{\boldsymbol{\mu}}_X[i]} + \tilde{\boldsymbol{\mu}}_V.$$

2. Calculate the value of $\hat{\Sigma}_Y[i]$ produced by the Kalman filter:
$$\hat{\Sigma}_Y[i] = \tilde{C}\hat{\Sigma}_X[i]\tilde{C}^T + \tilde{\Sigma}_V.$$

3. Calculate the gain $G[i]$ produced by the Kalman filter:
$$G[i] = \hat{\Sigma}_X[i]\tilde{C}^T\left(\tilde{\Sigma}_V + \tilde{C}\hat{\Sigma}_X[i]\tilde{C}^T\right)^{-1}.$$

4. Determine a gPC representation for the state estimate at stage i as it will be calculated by the Kalman filter, using (14) for $\boldsymbol{Y}[i]$:

$$\underline{\boldsymbol{\mu}_X[i]} = \underline{\hat{\boldsymbol{\mu}}_X[i]} + G[i]\left(\underline{\boldsymbol{Y}[i]} - \underline{\hat{\boldsymbol{\mu}}_Y[i]}\right).$$

5. Calculate the value of $\Sigma_X[i]$ produced by the Kalman filter:

$$\Sigma_X[i] = \left(I - G[i]\tilde{C}\right)\hat{\Sigma}_X[i].$$

6. Determine a gPC representation for the step-ahead state at stage i as it will be calculated by the Kalman filter:

$$\underline{\hat{\boldsymbol{\mu}}_X[i+1]} = \tilde{A}\underline{\boldsymbol{\mu}_X[i]} + \tilde{\boldsymbol{\mu}}_W.$$

7. Calculate the value of $\hat{\Sigma}_X[i+1]$ produced by the Kalman filter:

$$\hat{\Sigma}_X[i+1] = \tilde{A}\Sigma_X[i]\tilde{A}^T + \tilde{\Sigma}_W.$$

8. Determine the gPC representation for tracker error at stage i:

$$\underline{\boldsymbol{e}[i]} = \underline{\boldsymbol{X}[i]} - \underline{\boldsymbol{\mu}_X[i]}$$

The results of Step 8 of this algorithm give a complete characterization of the tracking error in gPC form.

5. EXAMPLE

As an example of the analysis approach outlined in this section, consider a simplified problem of tracking an object along a line, such as a vehicle traveling along a path. We assume that the object approximately follows a normative acceleration profile that is determined by its environment. This is the case, for example, with most vehicles in traffic. The state vector $\boldsymbol{X}[i]$ consists of the object's position and velocity when the ith sensor measurement becomes available, and the sensor platform produces a noisy measurement $Y[i]$ of the object's position. These are random quantities, and for the purposes of this example we model their random variation as arising from of several underlying sources. We collect abstract representations of these sources into a vector $\boldsymbol{Z}$, whose components are:

- Z_1 - variation in the timing of sensor data generation, for example between sensor sweeps;
- Z_2 - variation in the object's acceleration magnitude;
- Z_3 - variation in the object's acceleration time lag;
- Z_4 - effect of non-zero velocity on the sensor's position measurement;
- Z_5 - sensor measurement errors stemming from track association and/or feature extraction variation.

The quantity d from Section 3 is thus equal to 5.

We represent the state vector with two components $\boldsymbol{X}[i] = [U[i], U'[i]]^T$, where $U[i]$ denotes the position of the object at stage i and $U'[i]$ denotes its velocity. A typical state equation (cf. Eqn. (4)) has the form

$$\begin{bmatrix} \underline{U[i]} \\ \underline{U'[i]} \end{bmatrix} = \begin{bmatrix} 1 & \delta_t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{U[i-1]} \\ \underline{U'[i-1]} \end{bmatrix} + \begin{bmatrix} \underline{W_1[i]} \\ \underline{W_2[i]} \end{bmatrix},$$

where the underlined quantities have gPC series representations in terms of $\boldsymbol{Z}$. In this model, the position of the object at stage i equals the sum of three terms: (1) its position at stage $i-1$; (2) a time interval δ_t times its velocity at stage $i-1$; and (3) a random quantity $W_1[i]$ representing higher order effects, such as acceleration. Since the timing of sensor measurements is not guaranteed, we model δ_t as a random quantity expressed as a gPC series involving Z_1, which abstractly represents the underlying source of timing variation for the sensor. We assume that tracked objects follow a typical acceleration profile $ag(t-\tau)$, where the amplitude a and time lag τ vary randomly from one object to another. These can be represented as gPC series in terms of Z_2 and Z_3, respectively. The dominant contribution to $W_1[i]$ is the change in object position due to acceleration during the sensing period, which is approximately

$$\underline{W_1[i]} = \frac{1}{2}\underline{a}g(i\underline{\delta_t} - \underline{\tau})\underline{\delta_t}^2,$$

where, again, the underlined quantities have gPC series representations in terms of $\boldsymbol{Z}$. From the analysis equation (2), we conclude that $W_1[i]$ can be expressed in gPC form as

$$W_1[i] = \sum_{k=0}^{K^5} w_{1,k}\psi_k^{(5)}(\boldsymbol{Z}),$$

where

$$w_{1,k} = \frac{1}{2}\mathrm{E}\left[a(Z_2)g\left(i\delta_t(Z_1) + \tau(Z_3)\right)\delta_t^2(Z_1)\psi_k^{(5)}(\boldsymbol{Z})\right].$$

The coefficients in the gPC representation of $U[i]$ can thus be determined from those of $U[i-1]$, δ_t, $U'[i-1]$, and $W_1[i]$ using the properties of addition and multiplication in Section 3.

In a similar way, the gPC coefficients of object velocity $U'[i]$ can be determined from those of $U'[i-1]$ and $W_2[i]$. Since the change in velocity between sensor measurements equals acceleration multiplied by a change in time,

$$\underline{W_2[i]} = \underline{a}g(i\underline{\delta_t} - \underline{\tau})\underline{\delta_t}.$$

The change in velocity thus has a gPC representation

$$W_2[i] = \sum_{k=0}^{K^5} w_{2,k}\psi_k^{(5)}(\boldsymbol{Z}),$$

where

$$w_{2,k} = \mathrm{E}\left[a(Z_2)g\left(i\delta_t(Z_1) + \tau(Z_3)\right)\delta_t(Z_1)\psi_k^{(5)}(\boldsymbol{Z})\right].$$

Next, consider an example of the output equation (5). A typical video-based system will report an approximate object position at each stage by detecting candidate positions in each image and attempting to associate each object track with one of these positions. The reported object position will not be exact, and may be a function of the object's velocity, for example due to motion-induced blur. An example of the measurement model is

$$\underline{Y[i]} = \begin{bmatrix} 1 & \underline{\rho} \end{bmatrix} \begin{bmatrix} \underline{U[i]} \\ \underline{U'[i]} \end{bmatrix} + \underline{V[i]}. \tag{15}$$

Here ρ captures the first-order dependence of reported position on object velocity. There will be some variability in this quantity, for example due to the visual appearance of differing objects, and Z_4 is an abstract representation this variability. The quantity ρ has a gPC expansion in terms of Z_4. The noise term $V[i]$ captures variability in reported position due to all other effects, such as track association errors, feature extraction errors, etc. The sources of these errors are abstractly represented by Z_5. Thus, just as with $U[i]$ and $U'[i]$, we can determine the gPC coefficients of $Y[i]$ in terms of the gPC coefficients of ρ, $U[i]$, $U'[i]$, and $V[i]$.

Finally, we consider the behavior of a Kalman filter tracking algorithm for this example. As discussed in Section 4.1, a Kalman filter will be incorporate a set of (non-random) design parameters consisting of $\tilde{A}$, $\tilde{C}$, $\tilde{\boldsymbol{\mu}}_U$, $\tilde{\mu}_V$, $\tilde{\Sigma}_W$, and $\tilde{\Sigma}_V$. Substituting these and the gPC expression for $Y[i]$ from (15) into the analysis algorithm of Section 4.2, we can determine a complete characterization of Kalman filter tracking performance for this example system. The performance characterization will be expressed through the error sequence $\boldsymbol{e}[i]$ in Step 8 as a polynomial function of the random vector $\boldsymbol{Z}$. This characterization will have the form

$$\boldsymbol{e}[i] = \sum_{k=0}^{K^5} \boldsymbol{e}_k[i]\psi_k^{(5)}(\boldsymbol{Z}).$$

With this expression computed, it is possible to determine arbitrary statistics on this sequence, as described in Section 3. For example, the mean error in estimated state at stage i will be $\boldsymbol{e}_0[i]$, and the error covariance matrix at state i will be $\sum_{k=1}^{K^d} \boldsymbol{e}_k[i]\boldsymbol{e}_k[i]^T$. Similar expressions can be used to compute serial correlation (i.e. the correlation of errors between stages i and $i+j$) and higher order moments. Finally, the probabilities of particular events involving tracking error (i.e. the probability that tracking error will exceed a given amount) can be calculated as the probability that $\boldsymbol{Z}$ takes a value in the pre-image of that event.

6. CONCLUSIONS

In this paper we have outlined a theory by which a closed-form expression for the performance of ATR algorithms can be derived. The approach results in a complete characterization of algorithm performance, which is sufficient to calculate any performance statistic of interest. The approach is based on polynomial chaos theory, in which random quantities are expressed as generalized Fourier transforms of the functions that define them relative to an underlying probability space. The probability space is constructed as an abstract representation of all physical sources of random variability that affect the system. All random variables manipulated by the ATR algorithms are then expressed in terms of this probability space, an operation that is often relatively straightforward. The approach was applied to a class of Kalman Filter-based tracking algorithms to determine an exact expression for their state estimation error in non-white, non-Gaussian environments.

REFERENCES

1. N. Wiener, "The homogeneous chaos," *American Journal of Mathematics* **60**, pp. 897–936, Oct. 1938.
2. M. S. Deb, I. M. Babuška, and J. T. Oden, "Solution of stochastic partial differential equations using Galerkin finite element techniques," *Computer Methods in Applied Mechanics and Engineering* **190**, pp. 6359–6372, 2001.
3. D. Xiu and G. E. Karniadakis, "The Wiener-Askey polynomial chaos for stochastic differential equations," *SIAM Journal of Scientific Computation* **24**(2), pp. 619–644, 2002.
4. I. Babuška, R. Tempone, and G. E. Zouraris, "Galerkin finite element approximations of stochastic elliptic partial differential equations," *SIAM Journal of Numerical Analysis* **42**(2), pp. 800–825, 2004.
5. Y. Yan, "Galerkin finite element methods for stochastic parabolic partial differential equations," *SIAM Journal of Numerical Analysis* **43**(4), pp. 1363–1384, 2005.
6. H. G. Matthies and A. Keese, "Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations," *Computer Methods in Applied Mechanics and Engineering* **194**, pp. 1295–1331, 2005.
7. M. Grigoriu, "Galerkin solution for linear stochastic algebraic equations," *Journal of Engineering Mechanics* **132**(12), pp. 1277–1289, 2006.
8. D. Xiu and D. M. Tartakovsky, "Numerical methods for differential equations in random domains," *SIAM Journal of Scientific Computation* **28**(3), pp. 1167–1185, 2006.
9. F. S. Hoover and M. S. Triantafyllou, "Application of polynomial chaos in stability and control," *Automatica* **42**, pp. 789–795, 2006.
10. B. Wu, J. Zhu, and F. N. Najm, "Dynamic range estimation for nonlinear systems," in *IEEE/ACM International Conference on Computer Aided Design*, pp. 660–667, Nov. 2004.
11. D. R. Millman, P. I. King, R. C. Maple, P. S. Beran, and L. K. Chilton, "Estimating the probability of failure of a nonlinear aeroelastic system," *Journal of Aircraft* **43**, pp. 504–516, Mar. 2006.
12. D. Lucor, D. Xiu, C.-H. Su, and G. E. Karniadakis, "Predictability and uncertainty in CFD," *International Journal for Numerical Methods in Fluids* **43**, pp. 483–505, 2003.
13. Y. Yu, M. Zhao, T. Lee, N. Pestieau, W. Bo, J. Glimm, and J. Grove, "Uncertainty quantification for chaotic computational fluid dynamics," *Journal of Computational Physics* **217**, pp. 200–216, 2006.
14. O. M. Knio and O. P. Le Maître, "Uncertainty propagation in CFD using polynomial chaos decomposition," *Fluid Dynamics Research* **38**, pp. 616–640, 2006.
15. B. J. Debusschere, H. N. Najm, P. P. Pébay, O. M. Knio, R. G. Ghanem, and O. P. Le Maître, "Numerical challenges in the use of polynomial chaos representations for stochastic processes," *SIAM Journal of Scientific Computation* **26**(2), pp. 698–719, 2004.
16. R. J. Meinhold and N. D. Singpurwalla, "Understanding the Kalman filter," *The American Statistician* **32**, pp. 123–127, May 1983.