Out-of-Order Sigma-Point Kalman Filtering for Target Localization using Cooperating Unmanned Aerial Vehicles

Gregory L. Plett*, Dimitri Zarzhitsky, and Daniel J. Pack

Department of Electrical and Computer Engineering
United States Air Force Academy, USAFA, CO 80840, USA

Abstract. This chapter outlines our research efforts toward developing a cooperative target localization method based on multiple autonomous unmanned aerial vehicles (UAVs) that are outfitted with heterogeneous sensors. The current focus of the research includes (1) optimizing the UAV trajectories to place them at desired locations at desired times to capture target locations, (2) cooperative sensor scheduling, and (3) intelligent fusing of multiple sensor measurements to accurately estimate the position and velocity of a target. The focus of this paper is the sensor-fusion task. One might consider addressing this problem using some form of Kalman filter. However, a complicating factor in the present application is that sensor readings arrive out-of-sequence to the sensor-fusion process. For example, there is non-deterministic latency in the inter- and intra-UAV communication channels. We address this problem by developing an out-of-order sigma-point Kalman filter (O^3SPKF).

1 Introduction

Detecting and localizing targets using multiple cooperative heterogeneous sensors is a challenging problem that can directly impact military and law-enforcement applications such as intelligence, surveillance, and reconnaissance as well as civilian applications such as search-and-rescue and forest-fire early detection. The particular solution of our interest must address a number of challenging requirements: (a) covert/passive sensing must be used; (b) the dynamic characteristics of the target are unknown; (c) the target is episodically mobile; and (d) the target is intermittently occluded from particular sensing mechanism(s). The cooperative method proposed in this paper plays an important role in our larger overall goal to develop a multiple cooperative UAV system that can autonomously search, detect, and localize multiple targets [1,2]. Our solution addresses these requirements using a flight of small autonomous UAVs with heterogeneous sensing capabilities. Multiple autonomous UAVs offer certain advantages over other conventional sensor platforms. They offer robustness in the presence of a loss of members; can quickly search a large area; can operate

* Visiting Research Professor (Associate Professor on sabbatical from the University of Colorado at Colorado Springs, CO 80918, USA)

using a decentralized but cooperative control algorithm, requiring minimal human intervention; and they are small and relatively inexpensive, allowing quick and easy deployment.

Each UAV carries a suite of one or more sensors including perhaps: radio frequency (RF) sensors to detect and determine direction-of-arrival (DOA), time-difference-of-arrival (TDOA) and/or frequency-difference-of-arrival (FDOA); infrared sensors to detect heat signatures; and optical image sensors. Due to cost considerations and payload constraints, it is not desirable for every UAV to have a full complement of sensing capability. In our implementation, when a target is initially detected by one UAV, a small formation of UAVs comprising complementary heterogeneous sensors is autonomously assembled to localize the target. The UAVs then cooperatively locate the target by combining the sensor information collected by heterogeneous sensors onboard the UAVs. The output of the localization process gives an estimate of the target's position and velocity and provides error bounds on the estimate.

We integrate dynamic sensor fusion and target localization using a modified sigma-point Kalman filter (SPKF). SPKFs are a generalization of the ubiquitous Kalman filter [3,4] to problems with nonlinear descriptions.[1] The problem specifically addressed in this chapter is that sensor readings may arrive out-of-sequence to the fusion process due, for example, to non-deterministic communication-channel latency between UAVs. A similar issue was solved in [11] where the latency was known and deterministic. However, in our case, the latency is not known a priori and is variable, so we take quite a different approach, which we have named the "out-of-order sigma-point Kalman filter" (O^3SPKF).

This chapter proceeds by first outlining several approaches that may be taken to handle out-of-order measurements. Sigma-point Kalman filters are then reviewed to provide background for the development of the O^3SPKF. To illustrate the results, we then give an example model of target dynamics, the overall simulation system used, and some results. Finally, we close with some concluding remarks.

2 Approaches to Handling Out-of-Sequence Measurements

There are a number of straightforward approaches to handling out-of-sequence measurements that might be considered in a target-localization application. These include the methods that we call the "simple approach" and the "buffered approach," which will be described below. The O^3SPKF method is perhaps not quite as straightforward, but has performance advantages, as will be shown. All of these methods are based on sigma-point Kalman filtering, which is itself explained in Section 3. We limit ourselves to this one technology to make a valid

[1] One variety of SPKF is the unscented Kalman filter (UKF) [5,6,7], which has been used to locate targets using TDOA measurements [8]. We use the central difference Kalman filter (CDKF) [9,10] here since it has slightly higher theoretic accuracy [10] and requires fewer algorithm parameters be tuned.

comparison of results.[2] The differences between the methods we describe in this chapter reside in either (1) how the data is made available to the SPKF, as in the "simple" approach and in the "buffered" approaches, or (2) how the SPKF is modified to be able to accommodate out-of-sequence measurements, as in the O^{3}SPKF approach.

Before we outline the methods themselves, it is instructive to consider the method for evaluating their performance. A number of important metrics might be considered: What is the computational complexity of the algorithm? What is the required amount of auxiliary memory/storage for the algorithm? What is the estimation accuracy of the algorithm? For the methods described in this chapter, the first two questions are quite straightforward to answer. The third requires more discussion. The issue lies in the question "How does one compute a localization error estimate at an arbitrary time, since the filter is only updated at random, asynchronous points in time?"

To clarify this explanation, we define some notation. Let t_m be the time a particular measurement is taken, t_a be the time that measurement arrives for processing ($t_a = t_m + d_t$, where d_t is the transport/processing delay), t_f be the time the measurement arrives at the filter ($t_f = t_a + d_q$, where d_q is the queueing delay), t_x be the time associated with the filter's most recent state estimate, and t be the present (real) time. Further, let $x(t)$ be the true state at time t and $\hat{x}(t)$ be an estimate of the state corresponding to time t.

Control decisions need to be made based on $\hat{x}(t)$; however, the filter only "knows" $\hat{x}(t_x)$ corresponding to the timestamp of the measurement most recently incorporated into the filter's estimate. Since (with probability one) $t \neq t_x$, we must define a means to propagate the filter's most recent estimate $\hat{x}(t_x)$ forward in time to predict $\hat{x}(t)$. This can be done using the target's motion-model state equation. Note that $\hat{x}(t_x)$ has incorporated all measurements with $t_f < t$, but has not necessarily incorporated all measurements with $t_a < t$ or $t_m < t$ due to the delays involved. We begin to suspect that there will be a definite cost to transmission latency: not all measurements taken prior to the present time will be included in the target position estimate, thus degrading accuracy of the state prediction. Furthermore, there will be a cost to queueing latency since the greater the difference $t - t_x$, the greater length of time we need to predict over, further degrading accuracy. The ad-hoc methods that we describe in this chapter to process out-of-sequence measurements take the approach of trying to manage queueing latency to improve the accuracy of the estimates. We will see that the O^{3}SPKF approach improves on these ad-hoc methods by adding no latency, and making better use of all available measurements.

The Simple Approach to Handling Out-of-Sequence Measurements. Since we suspect that latency will degrade our real-time estimate of target state, our first

[2] We have found the SPKF to give a very good tradeoff between computational complexity and accuracy of location estimates, but we recognize that if additional computational resources were available, a technology such as a particle filter might produce even more accurate estimates.

ad-hoc approach is to simply discard all measurements that arrive at the filter out-of-sequence. That is, if a particular measurement has $t_m < t_x$, that measurement is discarded. We call this the "simple" approach.

The Buffered Approach to Handling Out-of-Sequence Measurements. The simple approach has the beneficial property that it adds no latency ($d_q = 0$). However, many potentially useful measurements are discarded. We will see later, that if the simple approach is used with UAVs having multiple sensors, it is possible that only the measurements taken by one of these sensors are used if the processing time required by the sensors is quite different. Therefore, we seek means whereby d_q may be adjusted to provide improved target state estimates without introducing significant complexity.

The method we call the "buffered" approach is one way to do this. We form a buffer of N measurements. The oldest measurement in the buffer has timestamp denoted $t_{\min}$. When a new measurement arrives, we compare its timestamp t_m against $t_{\min}$. If $t_m < t_{\min}$, the measurement is discarded, as in the simple approach. However, if $t_m \geq t_{\min}$, the oldest buffer measurement is removed from the buffer and is used to update the SPKF, and the presently received measurement is added to the buffer. The filter time t_x will be updated to $t_{\min}$. If the buffer is very large, few measurements will be discarded—which we expect will result in near-optimal estimation of $\hat{x}(t_x)$—but since the difference between t and t_x will generally be large—we expect poorer estimation of $\hat{x}(t)$. If the buffer is very small, many measurements will be discarded—resulting in poor estimation performance of $\hat{x}(t_x)$ and by extension of $\hat{x}(t)$—but the difference between t and t_x will generally be small. In the limit as the buffer size goes to zero, the buffered approach becomes the simple approach, and there will be some size N that optimizes the estimation performance of $\hat{x}(t)$.

When determining the buffer size N, one might consider modeling the inter-UAV transmission latency (the dominant effect) as an exponential random variable, as is common in communication theory. Then, if μ is the expected latency, 68% of measurements are received with $d_t < \mu$, 86% of measurements are received with $d_t < 2\mu$, and 95% of measurements are received with $d_t < 3\mu$. We might then choose $N = \lceil k\mu/T_s \rceil \times$number of UAVs$\times$number of sensors per UAV, where $k \in \{1, 2, 3\}$ selects the average percentage of measurements queued and used in the filter, and T_s is the inter-sample interval for each sensor.

The O^3SPKF Approach to Handling Out-of-Sequence Measurements. The simple and buffered approaches both result in measurements arriving at the SPKF in order (either by discarding all out-of-sequence measurements, or using the buffering mechanism to sort the great majority of measurements in-order, while still discarding a few out-of-sequence measurements). The approach we propose in this chapter is fundamentally different from either of these in that it modifies the SPKF itself to be able to accommodate the out-of-sequence measurements. If a new measurement arrives with $t_m \geq t_x$, it is incorporated in the filter state using the standard SPKF steps and t_x is updated to t_m. However, if a new measurement arrives with $t_m < t_x$, an alternate sequence of steps is executed to

update the filter state estimate using the novel information regarding the system state at time t_x in this stale measurement, and the filter time remains at t_x.

The O^{3}SPKF is not a buffering method, so incurs no additional latency ($d_q = 0$). Furthermore, it never discards measurements; therefore, we expect that it will give better estimates than the other two approaches. Also, the O^{3}SPKF is of the same computational complexity as SPKF, so we incur no processing penalty for using it. However, because the O^{3}SPKF does not discard any measurements (as do the simple and buffered methods), it will execute more often. Finally, since the O^{3}SPKF does not buffer measurements it has no additional auxiliary storage requirements.

We note in passing that there are some other approaches to fusing out-of-order measurements that bear some similarity to O^{3}SPKF, which we did not consider while preparing this chapter. One method maintains a buffer of sensor data, but unlike the simple buffered methods proposed above, updates the SPKF immediately upon receiving a new measurement. The SPKF state and covariance estimates are stored along with the corresponding measurement in the buffer. When an out-of-order measurement arrives, the filter state is updated by first "rolling back" to the estimate immediatly prior to that measurement, and the SPKF algorithm is applied repeatedly to all following measurements in the buffer—potentially resulting in many SPKF update steps per new data point. This method gives the best achievable results, but we did not consider it due to its requirements of a potentially large buffer and challenges in a real-time implementation. A second method is similar—upon receiving an out-of-order measurement, a Kalman smoother is run backward in time from t_x to t_m (potentially requiring many iterations of the smoother steps as each data point is considered again) to make a smoothed estimate of the state at time t_m. The present measurement is then incorporated into $\hat{x}(t_m)$ via a measurement update equation, and the state estimate is re-propagated to time t_x [12]. The O^{3}SPKF has similar steps to this method: It also propagates the present state back in time, but it always does this in one step (not requiring a buffer of measurements). Additionally, it does not update the state estimate at time t_m; rather, covariance calculations are performed using the present and prior data that allow direct updating of the state estimate at time t_x using the old measurement.

We now present the SPKF and the O^{3}SPKF. Readers familiar with either the UKF or SPKF might skim the next section to discover the notation that we use, and to see how we partition the SPKF into six steps which are then mirrored in the O^{3}SPKF in Section 4.

3 Sigma-Point Kalman Filters (SPKF)

Kalman filters are an intelligent (and sometimes optimal) way to estimate the unmeasurable "state" $x(t)$ of some dynamic system given measurements of a signal $u(t)$ possibly affecting that state (the dynamic "input", sometimes called a forcing function), and measurements $y(t)$ (the dynamic "output") related to linear or nonlinear combinations of members of that state and $u(t)$. Here, we

assume that the state of the target to be estimated comprises its position and velocity: that is, $x(t) = [p_x(t), p_y(t), v_x(t), v_y(t)]^T$, where $p_x(t)$ is the "x" position coordinate of the target, $p_y(t)$ is the "y" position coordinate of the target, $v_x(t)$ is the "x" velocity of the target and $v_y(t)$ is the "y" velocity of the target. (We could easily extend this to three dimensions by simply adding extra state elements $p_z(t)$ and $v_z(t)$.) The state vector $x(t)$ is assumed to have dynamics that can be modeled in a "state-space" form. For example, a relationship that can be used with some kinds of Kalman filter is:

$$\dot{x}(t) = f(x(t), u(t), w(t), t) \tag{1}$$
$$y(t) = h(x(t), u(t), v(t), t), \tag{2}$$

where $w(t)$ is an unmeasurable "process noise" that is often modeled as a zero-mean white Gaussian random process, $v(t)$ is unmeasurable "sensor noise" that is also modeled as a zero-mean white Gaussian random process, $f(\cdot)$ is the "state equation" function that captures the dynamics of the state, and $h(\cdot)$ is the "measurement equation" or "output equation" function that describes how the sensor measurements relate to the state.

Creating an optimum estimate $\hat{x}(t)$ of the true state $x(t)$ is a very challenging problem in general. Very close approximations to the optimum estimate can be made using particle filters, but these are too computationally intensive for our application. Alternative suboptimal solutions can be derived by assuming that the state estimation error always retains a Gaussian probability density function—this assumption is the basis of the original Kalman filter, the extended Kalman filter, and the sigma-point Kalman filters to be discussed. Then, rather than having to propagate the entire density function through time, we need only to evaluate the conditional mean and covariance of the state vector once each sampling interval and make updates to the estimates using the following two relationships:

$$\hat{x}^+(t_x) = \hat{x}^-(t_x) + L(t_x, t_m)\big(y(t_m) - \hat{y}(t_m)\big) \tag{3}$$
$$\Sigma^+_{\tilde{x}(t_x)} = \Sigma^-_{\tilde{x}(t_x)} - L(t_x, t_m)\Sigma^-_{\tilde{y}(t_m)}L(t_x, t_m)^T, \tag{4}$$

where the superscript T is the matrix/vector transpose operator, and

$$\hat{x}^+(t_x) = \mathbb{E}\left[x(t_x) \mid \mathbb{Y}^+\right] \tag{5}$$
$$\hat{x}^-(t_x) = \mathbb{E}\left[x(t_x) \mid \mathbb{Y}^-\right] \tag{6}$$
$$\hat{y}(t_m) = \mathbb{E}\left[y(t_m) \mid \mathbb{Y}^-\right] \tag{7}$$
$$\Sigma^-_{\tilde{x}(t_x)} = \mathbb{E}\left[(x(t_x) - \hat{x}^-(t_x))(x(t_x) - \hat{x}^-(t_x))^T\right] = \mathbb{E}\left[\tilde{x}^-(t_x)\tilde{x}^-(t_x)^T\right] \tag{8}$$
$$\Sigma^+_{\tilde{x}(t_x)} = \mathbb{E}\left[(x(t_x) - \hat{x}^+(t_x))(x(t_x) - \hat{x}^+(t_x))^T\right] = \mathbb{E}\left[\tilde{x}^+(t_x)\tilde{x}^+(t_x)^T\right] \tag{9}$$
$$\begin{aligned}\Sigma^-_{\tilde{y}(t_m)} &= \mathbb{E}\left[(y(t_m) - \hat{y}(t_m))(y(t_m) - \hat{y}(t_m))^T\right] = \mathbb{E}\left[\tilde{y}(t_m)\tilde{y}(t_m)^T\right] \\ &= \mathbb{E}\left[\tilde{y}(t_m)\tilde{y}(t_m)^T\right]\end{aligned} \tag{10}$$

$$L(t_x, t_m) = \mathbb{E}\left[(x(t_x) - \hat{x}^-(t_x))(y(t_m) - \hat{y}(t_m))^T\right] \left(\Sigma^-_{\tilde{y}(t_m)}\right)^{-1}$$
$$= \Sigma^-_{\tilde{x}(t_x)\tilde{y}(t_m)} \left(\Sigma^-_{\tilde{y}(t_m)}\right)^{-1}. \quad (11)$$

While this is a linear recursion, we have not directly assumed that the system model is linear. In the notation we use, time t_x indicates the timestamp of the measurement closest in time to the present and t_m indicates the timestamp of the most recently received measurement. (In the literature, no distinction is usually made between t_x and t_m. However, they will not be equal if measurements arrive at the Kalman filter out-of-sequence, which is the topic of this chapter). The decoration "circumflex" indicates an estimated quantity (e.g., $\hat{x}$ indicates an estimate of the true quantity x). A superscript "$-$" indicates an a priori estimate (i.e., an estimate of a quantity's value at some point in time based on all sensor data except the most recently received measurement) and a superscript "$+$" indicates an a posteriori estimate (i.e., an estimate of a quantity's value at some point in time based on all sensor data including the most recently received measurement). The decoration "tilde" indicates the error of an estimated quantity (e.g., $\tilde{x}$ is the difference between x and $\hat{x}$). The symbol $\Sigma_{xy} = \mathbb{E}[xy^T]$ indicates the auto- or cross-correlation of the variables in its subscript. (Note that often these variables are zero-mean, so the correlations are identical to covariances). Also, for brevity of notation, we often use Σ_x to indicate the same quantity as Σ_{xx}. The symbol $\mathbb{Y}^+$ indicates the set of all sensor readings taken up to and including the most recently received measurement, while $\mathbb{Y}^-$ indicates the set of all sensor readings excluding the most recently received measurement.

Equations (3) through (11) (and approximations thereof) may be used to derive either the Kalman filter, the extended Kalman filter, or the sigma-point Kalman filter. All members of this family of filters comply with a structured sequence of six steps per iteration, as outlined here.

General step 1: State estimate time update. For each measurement received, first assign $t_p = t_x$ (the prior value of the filter time), then set $t_x = \max(t_x, t_m)$ (the updated value of the filter time). For in-order measurements this results in $t_x = t_m > t_p$; for out-of-order measurements, this results in $t_x = t_p > t_m$. An updated state prediction $\hat{x}^-(t_x)$ of the value of $x(t_x)$ is then made, based on a priori information and the system model using (1) and (6).

General step 2: Error covariance time update. The second step is to determine the predicted state-estimate error covariance matrix $\Sigma^-_{\tilde{x}(t_x)}$ based on a priori information and the system model using (8).

General step 3: Estimate system output $y(t_m)$. The third step is to estimate the system's output corresponding to the timestamp of the most recently received measurement using present a priori information and (2) and (7).

General step 4: Estimator gain matrix $L(t_x, t_m)$. The fourth step is to compute the estimator gain matrix by evaluating (11). We again emphasize that the literature generally makes no distinction between t_x and t_m. However, this distinction is key to the O^{3}SPKF developed herein, of which the most important aspect is the ability to compute the correct value for $L(t_x, t_m)$.

General step 5: State estimate measurement update. The fifth step is to compute the a posteriori state estimate by updating the a priori estimate using the estimator gain and the output prediction error using (3).

General step 6: Error covariance measurement update. The final step computes the a posteriori error covariance matrix using (4). The estimator output comprises $\hat{x}^+(t_x)$ and $\Sigma^+_{\tilde{x}(t_x)}$. The estimator then waits until the next measurement is received, and returns to Step 1.

The standard Kalman filter is obtained by replacing (1) and (2) with linear state-space equations using a fixed sampling interval, resulting in closed-form equations for Steps 1–6. When the system equations are nonlinear, however, we must make some approximations to evaluate the expectation operators, and might consider the extended Kalman filter. A better alternative is the sigma-point Kalman filter, which has the same computational complexity as the extended Kalman filter, but more accurately approximates these steps.

SPKF computes estimates of the mean and covariance of the output of a nonlinear function using a small fixed number of function evaluations. A set of points (*sigma points*) is chosen as input to the function so that the (possibly weighted) mean and covariance of the points exactly matches the a priori mean and covariance of the input random variable being modeled. These points are then passed through the nonlinear function, resulting in a transformed set of output points. The a posteriori mean and covariance that are sought are then approximated by the mean and covariance of these points. Note that the sigma points comprise a fixed small number of vectors that are calculated deterministically—unlike particle filter methods.

Specifically, if the input random vector x has mean $\bar{x}$ and covariance $\Sigma_{\tilde{x}}$, then $p + 1 = 2 \times \dim(x) + 1$ sigma points are generated as the set

$$\mathcal{X} = \{\bar{x}, \bar{x} + \gamma\sqrt{\Sigma_{\tilde{x}}}, \bar{x} - \gamma\sqrt{\Sigma_{\tilde{x}}}\},$$

with members of $\mathcal{X}$ indexed from 0 to p, where γ is a tuning parameter (see below for an example), and where the matrix square root $R = \sqrt{\Sigma}$ computes a result such that $\Sigma = RR^T$. Typically, the efficient *Cholesky decomposition* [13,14] is used, resulting in a lower-triangular R. The reader can verify that the weighted mean and covariance of $\mathcal{X}$ equal the original mean and covariance of random vector x for a specific set of $\{\gamma, \alpha^{(\mathrm{m})}, \alpha^{(\mathrm{c})}\}$ if we define the weighted mean as $\bar{x} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{X}_i$, the weighted covariance as $\Sigma_{\tilde{x}} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} (\mathcal{X}_i - \bar{x})(\mathcal{X}_i - \bar{x})^T$, $\mathcal{X}_i$ as the ith column of $\mathcal{X}$, and both $\alpha_i^{(m)}$ and $\alpha_i^{(\mathrm{c})}$ as real scalars with the necessary (but not sufficient) conditions that $\sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} = 1$ and $\sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} =$

1. The various sigma-point methods differ only in the choices taken for these weighting constants. The two most common methods are the *unscented Kalman filter* (UKF) [5,6,7,15] and the *central difference Kalman filter* (CDKF) [9,10]. The CDKF has only one "tuning parameter" h, which makes implementation simpler. It also has marginally higher theoretic accuracy than the UKF [10], so we focus on this method in the application sections later. Using the CDKF,

$$\begin{aligned} \gamma &= h, \quad (h = \sqrt{3} \text{ for Gaussian distributions}) \\ \alpha_0^{(\mathrm{m})} &= \alpha_0^{(\mathrm{c})} = \frac{h^2 - \dim(x)}{h^2} \\ \alpha_i^{(\mathrm{m})} &= \alpha_i^{(\mathrm{c})} = \frac{1}{2h^2}, \quad i \neq 0 \end{aligned}$$

To use SPKF in a general estimation problem, with nonlinear state and output equations, we first define an augmented random vector x^a that combines the randomness of the state, process noise, and sensor noise. This augmented vector is then used as the state in the estimation process. However, we will assume a linear state equation with zero-mean process noise and additive zero-mean sensor noise to the otherwise nonlinear output equation. This allows the SPKF steps to be somewhat simplified. The model we use is:

$$\begin{aligned} x(t) &= A(t_0)x(t - t_0) + w_{t_0}(t), \qquad t_0 > 0 \\ y(t) &= h(x(t), u(t)) + v(t), \end{aligned}$$

where $A(t_0)$ is a state-transition matrix that represents the homogeneous dynamics of the state over a generic time interval t_0. For a given t_0, $A(t_0)$ is a constant matrix; however, in this work we receive measurements at random times, so we must treat t_0 and therefore $A(t_0)$ as variable in the development of the algorithm.

In this section, we will assume that measurements arrive in-sequence, such that when the measurement arrives at the sensor-fusion process $t_m \geq t_x$. We will call this case the "In-order SPKF."

In-order SPKF step 1: State estimate time update. First, assign $t_p = t_x$ (the prior value of the filter time), then set $t_x = \max(t_x, t_m) = t_m$. We desire to estimate $\hat{x}^-(t_x)$ using prior information regarding $x(t_p)$ and the state equation. To do so, we compute sigma points $\mathcal{X}^+(t_p)$ corresponding to the prior state and covariance estimates. These $p + 1$ vectors are

$$\mathcal{X}^+(t_p) = \left\{ \hat{x}^+(t_p), \hat{x}^+(t_p) + \gamma\sqrt{\Sigma^+_{\tilde{x}(t_p)}}, \hat{x}^+(t_p) - \gamma\sqrt{\Sigma^+_{\tilde{x}(t_p)}} \right\}.$$

Sigma points corresponding to a prediction of the state at time t_x are generated by evaluating the process equation $f(\cdot)$ using all $\mathcal{X}_i^+(t_p)$ (where the subscript i denotes that the ith vector is being extracted from the original set), yielding the a priori sigma points $\mathcal{X}_i^-(t_x)$. The state prediction is a weighted average of the $\mathcal{X}_i^-(t_x)$. In general,

$$\begin{aligned}\hat{x}^-(t_x) &= \mathbb{E}\left[f(x(t_p), u(t_x), w(t_x)) \mid \mathbb{Y}^-\right] \\ &\approx \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{X}_i^-(t_x).\end{aligned}$$

However, this simplifies for our linear state equation. If $t_0 = t_m - t_p$,

$$\begin{aligned}\hat{x}^-(t_x) &= \mathbb{E}\left[A_{t_0} x(t_p) + w_{t_0}(t_x) \mid \mathbb{Y}^-\right] \\ &= \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{X}_i^-(t_x) = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} A(t_0) \mathcal{X}_i^+(t_p) \\ &= A(t_0)\hat{x}^+(t_p).\end{aligned}$$

In-order SPKF step 2: Error covariance time update. Using the a priori sigma points from Step 1, the a priori covariance estimate is computed as

$$\Sigma_{\tilde{x}(t_x)}^- = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \big(\mathcal{X}_i^-(t_x) - \hat{x}^-(t_x)\big)\big(\mathcal{X}_i^-(t_x) - \hat{x}^-(t_x)\big)^T + \Sigma_{w_{t_0}}.$$

For our linear state equation, this again simplifies:

$$\Sigma_{\tilde{x}(t_x)}^- = A(t_0)\Sigma_{\tilde{x}(t_p)}^+ A(t_0)^T + \Sigma_{w_{t_0}}.$$

In-order SPKF step 3: Predict system output $y(t_m) = y(t_x)$. The system output is predicted by evaluating the model output equation using the sigma points describing the state at time t_m. The in-order case has $t_m = t_x$, so we first compute the points $\mathcal{Y}_i^-(t_m) = h(\mathcal{X}_i^-(t_m), u(t_m)) = h(\mathcal{X}_i^-(t_x), u(t_m))$. The output estimate is then

$$\begin{aligned}\hat{y}(t_m) &= \mathbb{E}\left[h(x(t_x), u(t_m)) + v(t_m) \mid \mathbb{Y}^-\right] \\ &\approx \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} h(\mathcal{X}_i^-(t_x), u(t_m)) \\ &= \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{Y}_i^-(t_m).\end{aligned}$$

In-order SPKF step 4: Estimator gain matrix $L(t_x, t_m)$. To compute the estimator gain matrix, we must first compute the required covariance matrices.

$$\begin{aligned}\Sigma_{\tilde{y}(t_m)}^- &= \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \big(\mathcal{Y}_i^-(t_m) - \hat{y}(t_m)\big)\big(\mathcal{Y}_i^-(t_m) - \hat{y}(t_m)\big) + \Sigma_v \\ \Sigma_{\tilde{x}(t_x)\tilde{y}(t_m)}^- &= \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \big(\mathcal{X}_i^-(t_x) - \hat{x}^-(t_x)\big)\big(\mathcal{Y}_i^-(t_m) - \hat{y}(t_m)\big).\end{aligned}$$

Then, we simply compute $L(t_x, t_m) = \Sigma_{\tilde{x}(t_x)\tilde{y}(t_m)}^- \left(\Sigma_{\tilde{y}(t_m)}^-\right)^{-1}$.

In-order SPKF step 5: State estimate measurement update. The a posteriori state estimate is computed using (3).

In-order SPKF step 6: Error covariance measurement update. The state estimate error covariance matrix is updated directly from the optimal formulation: $\Sigma^{+}_{\tilde{x}(t_x)} = \Sigma^{-}_{\tilde{x}(t_x)} - L(t_x, t_m)\Sigma^{-}_{\tilde{y}(t_m)}L(t_x, t_m)^T$.

For reference, the in-order SPKF optimized for a linear state equation and additive sensor noise is summarized in Table 1.

4 Out-of-Order Sigma-Point Kalman Filters (O^{3}SPKF)

This chapter introduces a novel variant of the SPKF that allows the filter to be updated using out-of-sequence sensor data. That is, the filter state estimate may already be updated to time t_x using data sensed at t_x when a new piece of information arrives that is the result of a sensor reading taken at time $t_m < t_x$. The most common reasons for such out-of-sequence data include: inter- and intra-UAV communication latency, and processing latency. Ideally, this old sensor data should not be discarded, since it still contains information related to the target's present state, but its impact should be discounted appropriately, based on how stale the measurement is.

A similar problem was treated in [11], where a SPKF needed to be updated based on time-lagged sensor data from a global positioning system (GPS) unit. In their work, however, the time lag was constant and known a priori. In our case, the time lag is not constant, neither is it known a priori. However, we do assume that sensor data has a time-stamp on it so that we can calculate the time lag. Nevertheless, reference [11] gives us a clue as to how to modify the SPKF to our purposes.

In this section, we will assume that measurements arrive out-of-sequence; that is, $t_m < t_x$.

Out-of-order SPKF steps 1 and 2: State estimate time update: First, assign $t_p = t_x$ (the prior value of the filter time), then set $t_x = \max(t_x, t_m) = t_x$. We desire to estimate $\hat{x}^-(t_x)$ using prior information regarding $x(t_p)$ and the state equation. However, since t_x has not been changed by this measurement, we simply retain the prior values of $\hat{x}^-(t_x) = \hat{x}^+(t_x)$ and $\Sigma^{-}_{\tilde{x}(t_x)} = \Sigma^{+}_{\tilde{x}(t_x)}$.

Out-of-order SPKF step 3: Estimate system output $y(t_m) \neq y(t_x)$: When using out-of-sequence measured data to update the SPKF, the state update equation maintains the same linear form $\hat{x}^+(t_x) = \hat{x}^-(t_x) + L(t_x, t_m)(y(t_m) - \hat{y}(t_m))$. The key insight from [11] is that in such a case, $L(t_x, t_m)$ should be calculated via Eq. (11) instead of using the standard SPKF formulation where $t_x = t_m$. In order to compute this update, we require an estimate $\hat{y}(t_m)$ and the covariances required to compute $L(t_x, t_m)$. These in turn require sigma points representing $\hat{x}^-(t_x)$ and $\hat{y}(t_m)$. The first are easily computed:

$$\mathcal{X}^-(t_x) = \left\{\hat{x}^-(t_x), \hat{x}^-(t_x) + \gamma\sqrt{\Sigma^{-}_{\tilde{x}(t_x)}}, \hat{x}^-(t_x) - \gamma\sqrt{\Sigma^{-}_{\tilde{x}(t_x)}}\right\}.$$

Table 1. Summary of variable sample period in-order SPKF using linear state equation and additive noises.

Nonlinear state-space model:

$$x(t) = A(t_0)x(t-t_0) + w_{t_0}(t)$$
$$y(t) = h(x(t), u(t)) + v(t),$$

where $w_{t_0}(t)$ and $v(t)$ are independent, zero-mean Gaussian noise processes of covariance matrices $\Sigma_{w_{t_0}}$ and Σ_v, respectively.

Definition: Let $p = 2 \times \dim(x(t))$.
Initialization: At time zero, set $t_x = 0$ and

$$\hat{x}^+(0) = \mathbb{E}\left[x(0)\right]$$
$$\Sigma^+_{\tilde{x}(0)} = \mathbb{E}\left[(x(0) - \hat{x}^+(0))(x(0) - \hat{x}^+(0))^T\right]$$

Computation: For each sample occurring in-order, (i.e., $t_m \geq t_x$) compute:

Initialize time pointers: $t_p = t_x$, $t_x = t_m$, and $t_0 = t_x - t_p$.

State est. time update: $\hat{x}^-(t_x) = A(t_0)\hat{x}^+(t_p)$.

Error cov. time update: $\Sigma^-_{\tilde{x}(t_x)} = A(t_0)\Sigma^+_{\tilde{x}(t_p)}A(t_0)^T + \Sigma_{w_{t_0}}$.

Output estimate:

$$\mathcal{X}^-(t_x) = \Big\{\hat{x}^-(t_x), \hat{x}^-(t_x) + \gamma\sqrt{\Sigma^-_{\tilde{x}(t_x)}}, \hat{x}^-(t_x) - \gamma\sqrt{\Sigma^-_{\tilde{x}(t_x)}}\Big\}.$$
$$\mathcal{Y}_i(t_m) = h(\mathcal{X}_i^-(t_x), u(t_m)).$$
$$\hat{y}(t_m) = \textstyle\sum_{i=0}^{p} \alpha_i^{(\mathrm{m})}\mathcal{Y}_i(t_m).$$

Estimator gain matrix:

$$\Sigma^-_{\tilde{y}(t_m)} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})}\big(\mathcal{Y}_i(t_m) - \hat{y}(t_m)\big)\big(\mathcal{Y}_i(t_m) - \hat{y}(t_m)\big)^T + \Sigma_v.$$

$$\Sigma^-_{\tilde{x}(t_x)\tilde{y}(t_m)} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})}\big(\mathcal{X}_i^-(t_x) - \hat{x}^-(t_x)\big)\big(\mathcal{Y}_i(t_m) - \hat{y}(t_m)\big)^T.$$

$$L(t_x, t_m) = \Sigma^-_{\tilde{x}(t_x)\tilde{y}(t_m)}\left(\Sigma^-_{\tilde{y}(t_m)}\right)^{-1}.$$

State est. meas. update: $\hat{x}^+(t_x) = \hat{x}^-(t_x) + L(t_x, t_m)\big(y(t_m) - \hat{y}(t_m)\big)$.

Error cov. meas. update: $\Sigma^+_{\tilde{x}(t_x)} = \Sigma^-_{\tilde{x}(t_x)} - L(t_x, t_m)\Sigma^-_{\tilde{y}(t_m)}L^T(t_x, t_m)$.

It remains to calculate the sigma points to represent the distribution of $\hat{y}(t_m)$—we can do so using the output equation $h(\cdot)$ to find these output sigma points if we are able to calculate the sigma points representing $\hat{x}^-(t_m)$. To do so, consider the following specific form of a state equation where we define the time interval $t_0 = t_x - t_m$[3]

$$\begin{aligned} x(t_x) &= A(t_0)x(t_m) + w_{t_0}(t_x) \\ x(t_m) &= A(t_0)^{-1}x(t_x) - A(t_0)^{-1}w_{t_0}(t_x) \\ \hat{x}^-(t_m) &= \mathbb{E}\left[x(t_m)|\mathbb{Y}^-\right] = A(t_0)^{-1}\hat{x}^-(t_x), \end{aligned}$$

where $\mathbb{Y}^-$ is the history of all measurements, excluding the "new" measurement taken at time t_m. Therefore, we can "predict" a prior state estimate given the present state estimate. The prior covariance can also be computed, and is found to be

$$\Sigma^-_{\tilde{x}(t_m)} = A(t_0)^{-1}\left(\Sigma^-_{\tilde{x}(t_x)} + \Sigma_{w_{t_0}}\right)A(t_0)^{-T}.$$

Using these two quantities, we can compute the desired sigma points representing $\hat{x}^-(t_m)$ as

$$\mathcal{X}^-(t_m) = \left\{\hat{x}^-(t_m), \hat{x}^-(t_m) + \gamma\sqrt{\Sigma^-_{\tilde{x}(t_m)}}, \hat{x}^-(t_m) - \gamma\sqrt{\Sigma^-_{\tilde{x}(t_m)}}\right\}.$$

These sigma points are passed through the output equation $h(\cdot)$ to first form $\mathcal{Y}_i^-(t_m) = h(\mathcal{X}_i^-(t_m), u(t_m))$ and then $\hat{y}^-(t_m)$ as before.

Out-of-order SPKF steps 4–6: The remaining steps are straightforward now that we have a means for calculating the sigma points corresponding to $y(t_m)$. The entire O^{3}SPKF for a linear state equation and additive sensor noise is summarized in Table 2.

5 An Example Model of Motion

In order to use any Kalman filtering technique to localize a target, we require a model of the target's dynamics. Due to the non-cooperative nature of the target we wish to localize in our present research, this model cannot be known a priori; therefore, we must employ an approximate model. Here, we use a "nearly constant velocity" (NCV) model of dynamics. For a state vector $x(t) = [p_x(t), p_y(t), v_x(t), v_y(t)]^T$, where $p_x(t)$ is the "x" position coordinate of the target, $p_y(t)$ is the "y" position coordinate of the target, $v_x(t)$ is the "x"

[3] Note that the particular form of a linear state equation given above is not necessary for this general idea to work; however, if the equation is nonlinear, it must be locally Lipschitz. Sigma points representing $x(t_x)$ and $w_{t_0}(t_x)$ must be propagated backward in time to compute the mean and covariance estimates of $x(t_m)$.

Table 2. Summary of variable sample period out-of-order SPKF using linear state equation and additive noises.

Nonlinear state-space model:

$$x(t) = A(t_0)x(t-t_0) + w_{t_0}(t)$$
$$y(t) = h(x(t), u(t)) + v(t),$$

where $w_{t_0}(t)$ and $v(t)$ are independent, zero-mean Gaussian noise processes of covariance matrices $\Sigma_{w_{t_0}}$ and Σ_v, respectively.

Definition: Let $p = 2 \times \dim(x(t))$.
Initialization: At time zero, set $t_x = 0$ and

$$\hat{x}^+(0) = \mathbb{E}\left[x(0)\right]$$
$$\Sigma^+_{\tilde{x}(0)} = \mathbb{E}\left[(x(0) - \hat{x}^+(0))(x(0) - \hat{x}^+(0))^T\right]$$

Computation: For each sample occurring out-of-order, (i.e., $t_m < t_x$) compute:

Initialize time pointers:
$$t_0 = t_x - t_m.$$

Output estimate:
$$\hat{x}^-(t_m) = A(t_0)^{-1}\hat{x}^+(t_x).$$
$$\Sigma^-_{\tilde{x}(t_m)} = A(t_0)^{-1}\left(\Sigma^+_{\tilde{x}(t_x)} + \Sigma_{w_{t_0}}\right)A(t_0)^{-T}.$$
$$\mathcal{X}^-(t_m) = \Big\{\hat{x}^-(t_m), \hat{x}^-(t_m) + \gamma\sqrt{\Sigma^-_{\tilde{x}(t_m)}},$$
$$\hat{x}^-(t_m) - \gamma\sqrt{\Sigma^-_{\tilde{x}(t_m)}}\Big\}.$$
$$\mathcal{Y}_i(t_m) = h(\mathcal{X}^-_i(t_m), u(t_m)).$$
$$\hat{y}(t_m) = \textstyle\sum_{i=0}^{p} \alpha^{(\mathrm{m})}_i \mathcal{Y}_i(t_m).$$

Estimator gain matrix:
$$\mathcal{X}^+(t_x) = \Big\{\hat{x}^+(t_x), \hat{x}^+(t_x) + \gamma\sqrt{\Sigma^+_{\tilde{x}(t_x)}},$$
$$\hat{x}^+(t_x) - \gamma\sqrt{\Sigma^+_{\tilde{x}(t_x)}}\Big\}.$$
$$\Sigma^-_{\tilde{y}(t_m)} = \sum_{i=0}^{p} \alpha^{(\mathrm{c})}_i \big(\mathcal{Y}_i(t_m) - \hat{y}(t_m)\big)\big(\mathcal{Y}_i(t_m) - \hat{y}(t_m)\big)^T + \Sigma_v.$$
$$\Sigma^-_{\tilde{x}(t_x)\tilde{y}(t_m)} = \sum_{i=0}^{p} \alpha^{(\mathrm{c})}_i \big(\mathcal{X}^+_i(t_x) - \hat{x}^+(t_x)\big)\big(\mathcal{Y}_i(t_m) - \hat{y}(t_m)\big)^T.$$
$$L(t_x, t_m) = \Sigma^-_{\tilde{x}(t_x)\tilde{y}(t_m)}\left(\Sigma^-_{\tilde{y}(t_m)}\right)^{-1}.$$

State est. meas. update:
$$\hat{x}^+(t_x) = \hat{x}^-(t_x) + L(t_x, t_m)\big(y(t_m) - \hat{y}(t_m)\big).$$

Error cov. meas. update:
$$\Sigma^+_{\tilde{x}(t_x)} = \Sigma^-_{\tilde{x}(t_x)} - L(t_x, t_m)\Sigma^-_{\tilde{y}(t_m)}L^T(t_x, t_m).$$

velocity of the target and $v_y(t)$ is the "y" velocity of the target, we have:

$$\dot{x}(t) = \underbrace{\begin{bmatrix} 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 0\,0\,0\,0 \\ 0\,0\,0\,0 \end{bmatrix}}_{A} x(t) + w(t)$$
$$y(t) = h(x(t), u(t)) + v(t),$$

where the stochastic signals $w(t)$ and $v(t)$ are assumed to be Gaussian and white, and sensor noise $v(t)$ has covariance matrix Σ_v and process noise $w(t)$ has covariance matrix $\Sigma_w(t) = \mathrm{diag}(0, 0, \sigma^2, \sigma^2)$. The output equation depends on $h(\cdot)$, which itself depends on the sensor being used to produce a measurement and perhaps a measurable input signal $u(t)$. This model says, in effect, that the target velocity is generally constant except via perturbations to its acceleration through $w(t)$, and that measurements may be taken that somehow relate to the position and velocity of the target.

We will be updating the Kalman filter at non-deterministically separated discrete points in time. Therefore, we need to be able to integrate the effect of $w(t)$ on $x(t)$ over a variable period t_0 (the integral essentially performing a convolution) and result in a variable-sample-rate discrete-time model of the form:

$$x(t + t_0) = A(t_0)x(t) + w_{t_0}(t)$$
$$y(t) = h(x(t), u(t)) + v(t).$$

(We see that the output equation is unchanged). We compute $A(t_0) = e^{At_0}$, where $e^{(\cdot)}$ is the matrix-exponential function. We further compute [16]

$$\Sigma_{w_{t_0}} = \int_0^{t_0} e^{A(t_0-\tau)} \Sigma_w e^{A^T(t_0-\tau)}\, d\tau$$

to evaluate the equivalent discrete-time noise covariance based on the continuous-time noise covariance. For the NCV model, the state equation becomes

$$x(t + t_0) = \underbrace{\begin{bmatrix} 1\,0\,t_0\,0 \\ 0\,1\,0\,t_0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix}}_{A(t_0)} x(t) + w_{t_0}(t)$$

where $\Sigma_{w_{t_0}}$ can be evaluated analytically, and is found to be

$$\Sigma_{w_{t_0}} = \begin{bmatrix} \frac{t_0^3\sigma^2}{3} & 0 & \frac{t_0^2\sigma^2}{2} & 0 \\ 0 & \frac{t_0^3\sigma^2}{3} & 0 & \frac{t_0^2\sigma^2}{2} \\ \frac{t_0^2\sigma^2}{2} & 0 & t_0\sigma^2 & 0 \\ 0 & \frac{t_0^2\sigma^2}{2} & 0 & t_0\sigma^2 \end{bmatrix}.$$

This model is suitable for use with either the in-order SPKF or the out-of-order SPKF as developed in this chapter.

6 Performance Comparisons

Results indicative of the performance of the simple, buffered, and O^{3}SPKF approaches were generated via simulation of multiple UAVs locating a mobile target. Target and UAV trajectories were generated using the United States Air Force Academy (USAFA) multiple-UAV simulator, to be described next. The overall methodology for generating results will then be discussed.

6.1 The USAFA Multiple UAV Simulation System Control Architecture

In this section, we briefly present the distributed control architecture we developed to search, detect, and locate ground targets using multiple UAVs. The purpose is to provide readers the proper context in which the out-of-order Sigma Point Kalman Filter technique allows us to achieve our overall goal. As mentioned earlier, the overall goal is to develop a cooperative multiple heterogeneous UAVs system for military applications. In particular, we are interested in developing a distributed control architecture for each UAV that can optimize transmitted sensor information obtained by nearby UAVs. Collectively, the multiple UAVs cooperate to search, detect, and locate ground targets. To that end, we have developed the following control architecture.

The control architecture is made of a behavior-based state machine with four different states shown in Fig. 1: Global Search (GS), Approach Target (AT), Locate Target (LT), and Target Re-acquisition (TR). Each UAV operates in one of the four states at a time. The switch between two operating states is based on the current state of a UAV, sensor values obtained from the UAV and other neighboring UAVs, and state data transmitted from other UAVs in the mission area. For details of the switching conditions, see Table 3.

Table 3. List of events that trigger decisions to change states in the decision state machine of each UAV.

#	From	To	Event
1	GS	AT	New target (e.g., RF emitter) detected by UAV's sensor.
2	GS	AT	Decision to cooperate with an ongoing localization effort.
3	AT	LT	UAV arrives at orbit range from target's estimated position.
4	AT	GS	Target becomes occluded (e.g., emitter stops transmitting).
5	AT	GS	Decision to abandon an ongoing localization effort.
6	LT	GS	Target successfully located.
7	LT	TR	Target becomes occluded (e.g., emitter stops transmitting).
8	TR	AT	Target detected by UAV's sensor.
9	TR	GS	Maximum time for TR reached.

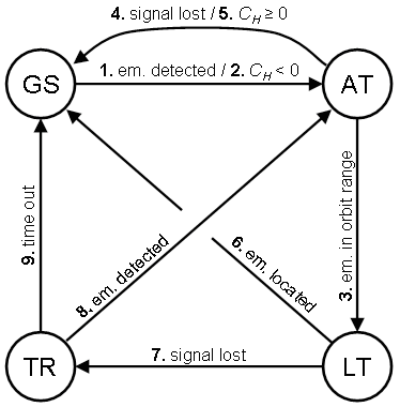


Fig. 1. Decision state machine for UAV state selection. The numbered events that trigger each particular directional connector are listed in Table 3.

Each UAV starts in the GS state when launched. During this state, a UAV uses a set of heuristic rules to guide its movement. The rules include visiting locations with little or no recent history, moving away from other nearby UAVs to maximize the search coverage, and moving straight, if possible, to reduce fuel use. Once a target is detected, the UAV that detected the target switches to the AT state and approaches the target, while other UAVs that receive the target detection information will independently decide whether to approach the detected target or continue to operate in the current operating state based on the estimated distance of the target, the number of UAVs that are approaching the target, and the estimated number of targets in the mission area that have not been detected. Once a UAV is within a region from which it can safely locate the target, it switches the operating state from AT to LT and flies around the target with a pre-determined orbit. Other UAVs that committed to help locate the target also switch to the LT state as they enter the orbit. As the UAVs fly around the target, they position themselves to maximize collective sensing capabilities while combining sensor data among the UAVs on the orbit. It is this state of our operation where the current work on O^3SPKF is used to combine multiple sensor information obtained by the UAVs. A target may disappear from the sensors before it is localized within a desired accuracy. For example, for a radio frequency signal emitting target, it may stop emitting before it can be located. For such situations, UAVs who are operating in the LT state switch to the TR state. During this state, a UAV engages in a search pattern similar to a global search except in a smaller scale to continue to look for the lost target. The UAV will continue in this state either until the target reappears at which time it switches back to the LT state or when a pre-determined time interval has elapsed at which time it switches to the GS state.

Figure 2 shows a screenshot of the simulator in action. The emitter location is indicated by a white cross in the center of the large circle—the emitter leaves

behind it a fading trail of crosses showing its path. The large circle indicates the desired orbiting radius of the UAVs—this is not possible in practice due to the random motion of the targets, but is approximated by the UAV control algorithm. The small squares denote the UAV positions (two in this case). Lines are drawn between the UAVs and their target-position estimates, with the SPKF state three-sigma uncertainty denoted by the ellipse centered on the target position estimate.

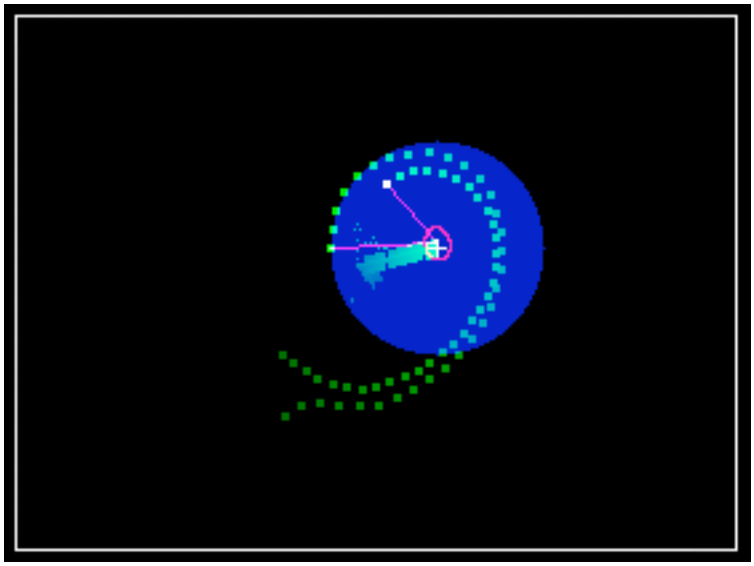


Fig. 2. Screenshot of simulation system in action.

6.2 The Simulation Process

The following methodology was employed

- First, the USAFA Multiple UAV Simulation System was used to generate the trajectories of a mobile target and the UAVs tracking it. UAV locations were initialized by randomly placing them within a 2 km radius of the target. The UAV flight paths were then controlled to converge to an orbit of 0.5 km distance from the target, with inter-UAV angles of 90° for a two-UAV simulation and ±120° for a three-UAV simulation. Locations of the target and UAVs were recorded once per second (of simulated time). Targets moved according to the NCV model with $\sigma = 2 \times 10^{-4}$, with maximum velocity limited to 20 km/h. Nominal UAV velocity was 100 km/h.
- Secondly, randomness was applied to the simulated data. Measurement requests were made to each sensor at a nominal sample rate of 1 Hz, corre-

sponding to the original data. Random clock timing jitter uniformly distributed between -5 ms and 5 ms was applied to each measurement time (locations of target and UAV were interpolated at these instants from the original data). Two sensors per UAV were simulated:

- One sensor providing target emission DOA information, with measurement timestamp of 0.05 s after the measurement was requested, and a sensor-data processing time uniformly distributed between $[0.06\,\text{s}, 0.061\,\text{s}]$ (the randomness accounts for timing uncertainty in a multi-threaded processing system), and Gaussian sensor noise with zero mean and standard deviation of 6 deg. These characteristics correspond roughly to a radio-frequency (RF) DOA sensor that we are currently building for a prototype UAV.
- A second sensor also providing DOA information, with measurement timestamp of 0.01 s after the measurement was requested, and a sensor-data processing time uniformly distributed between $[0.2\,\text{s}, 0.22\,\text{s}]$, and Gaussian sensor noise with zero mean and standard deviation of 3 deg. These characteristics correspond roughly to a camera-based DOA sensor that we are building for a prototype UAV. The measurement process is much faster than for the RF sensor, but requires a longer processing time.

We note that the "simple approach" will never make use of the data from the second sensor. We therefore expect that results using this approach will be quite poor, and that even a buffer of one sample could improve performance significantly.

Random latency was also added to those data traveling from one UAV to another. This was computed using an exponential random variable with differing means, as reported in the results sections below.

– The output sensor data and UAV position data from the second step were then used as input to the ideal method, the simple method, the buffered method with differing buffer lengths, and the O^{3}SPKF (the "ideal method" is a post-processing method that sorts all data in order by t_m, then applies an SPKF to it. This provides a performance bound that is not achievable in practice since the sorting process is non-causal). Target state estimates and uncertainties were output whenever they were updated.
– True position data from the first step and estimated position data from the third step were compared.

6.3 Results

Simulations using the above methodology were run for scenarios comprising two UAVs localizing a target and three UAVs localizing a target. In each case 500 simulations were run, with the data being processed by each of the methods, and the results averaged in a root-mean-square (RMS) sense. Figure 3 shows a representative plot of the average localization error versus time for an expected latency of 5 s. The ideal (non-achievable) case is best, as expected, followed by

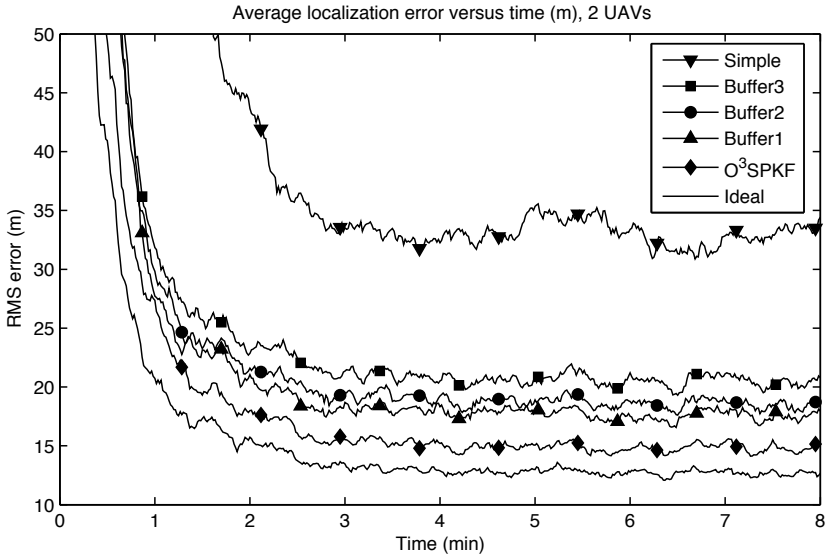
O^3SPKF, the buffered method with three different buffer lengths, and then the simple method. Note that the the "Buffer$k$" method denotes a buffer length $N = \lceil k\mu/T_s \rceil \times$ number of UAVs $\times$ number of sensors per UAV. The poor results of the simple method are not difficult to explain since the majority of the sensor readings are discarded. Perhaps surprisingly, the buffered method with the smallest buffer performed best—it appears that discarding data is not as costly as a stale state estimate (due to a larger prediction time) when the target is moving.

Various latencies were simulated, and summary results are presented in Fig. 4. Communication latencies of 0.2 s (the value we expect in our prototype UAV system), and 1 s through 5 s were simulated (again, the RMS average of the final localization error of 500 simulations per data point are plotted). We see that all methods degrade quite gently with increasing latency, and that the O^3SPKF performs best of all.

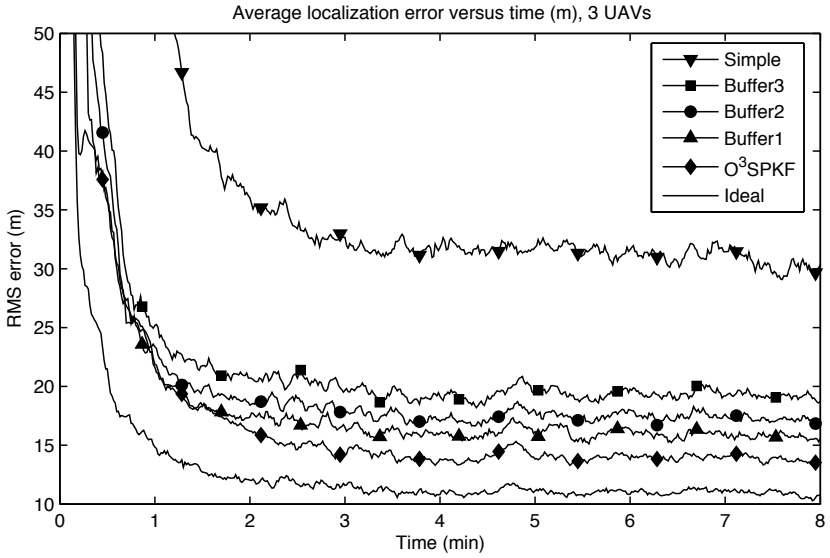
7 Conclusions

In this chapter we address the problem of sensor fusion to localize a target when some of the measurements arrive at the sensor-fusion process out-of-order. We propose that sigma-point Kalman filtering is a good approach to sensor fusion, but is not able to handle the out-of-order measurements directly. Several simple remedies are presented, which include discarding out-of-order measurements, or alternately buffering a number of measurements before presenting them to the sensor-fusion process so that the majority of them may be retained and sorted in order. The problem with simply discarding out-of-order measurements is that they do contain useful data regarding the target position, and it is not wise to throw away this information. The problem with buffering measurements is that the fusion process has added delay built into it, so that its state estimate is stale—this estimate must then be propagated forward in time to the present in order to make control decisions, and the propagation step adds error that increases with the amount of time required for the propagation.

We present an alternate approach to either of these ad-hoc methods. We rederive the sigma-point Kalman filter such that the modified filter, which we call the out-of-order sigma-point Kalman filter (O^3SPKF), is able to directly incorporate the out-of-order measurements without buffering and without discarding measurements. If a measurement arrives at the sensor-fusion process in-order, the standard SPKF steps are executed. If a measurement arrives at the sensor-fusion process out-of-order, the modified O^3SPKF steps are executed. Since no measurements are discarded by the O^3SPKF, it must execute its steps more frequently than the methods that do discard sensor data. (For example, the buffered methods for $k \in \{1, 2, 3\}$ retain on average 68%, 86%, and 95% of the measurements received (respectively), so the number of iterations of the SPKF required by the buffered methods are similarly that fraction of the number of iterations required by the O^3SPKF.) However, per iteration, the computational complexity of O^3SPKF is the same as SPKF, it does not require memory overhead for

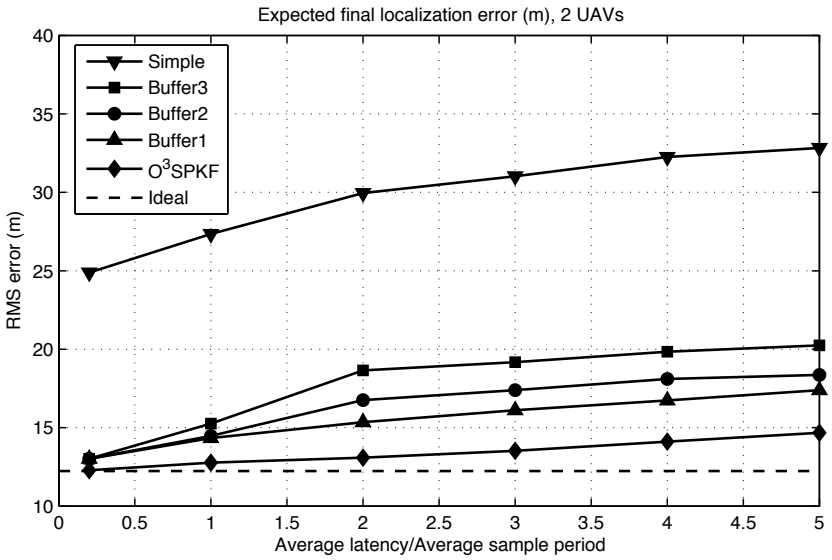


(a) Two UAVs

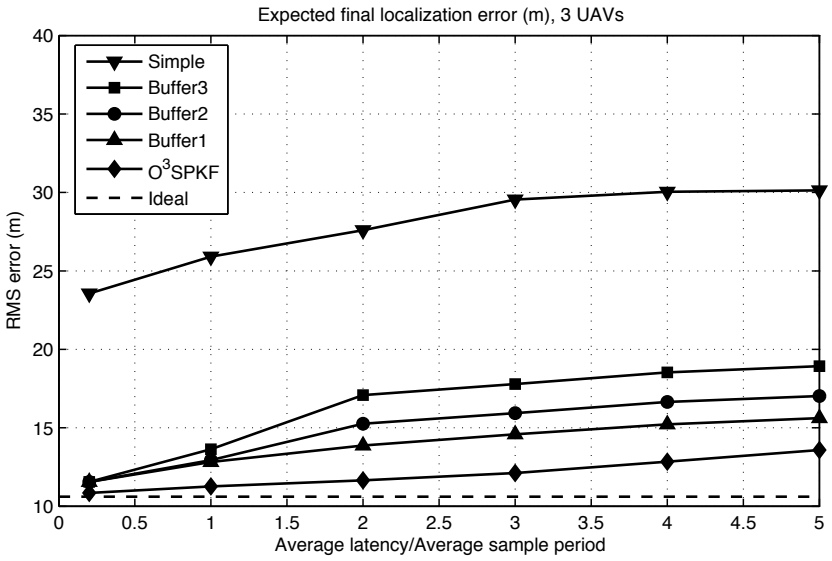


(b) Three UAVs

Fig. 3. Example of localization error improvement over time. "Ideal" = no-latency SPKF result.



(a) Two UAVs



(b) Three UAVs

Fig. 4. Simulation summary performance plots. "Ideal" = no-latency SPKF result.

buffering, and it gave the best simulation results of all the methods attempted. In conclusion, the O^3SPKF works very well, and is an excellent candidate for sensor fusion for the application of locating targets using multiple UAVs.

References

1. York, G., Pack, D.J.: Comparative study of time-varying target localization methods using multiple unmanned aerial vehicles: Kalman estimation and triangulation techniques. In: Proc. IEEE International Conference on Networking, Sensing and Control, (Tuscon, AZ) (March 2005) 305–10
2. Pack, D.J., York, G.: Developing a control architecture for multiple unmanned aerial vehicles to search and localize RF time-varying mobile targets: Part I. In: Proc. IEEE International Conference on Robotics and Automation, (Barcelona, Spain) (April 2005) 3965–70
3. Kalman, R.: A new approach to linear filtering and prediction problems. Transactions of the ASME—Journal of Basic Engineering **82, Series D** (1960) 35–45
4. Kalman, R.: The Seminal Kalman Filter Paper (1960). `http://www.cs.unc.edu/~welch/kalman/kalmanPaper.html` Accessed 20 May 2004.
5. Julier, S., Uhlmann, J., Durrant-Whyte, H.: A new approach for filtering nonlinear systems. In: Proceedings of the American Control Conference. (1995) 1628–32
6. Julier, S., Uhlmann, J.: A new extension of the Kalman filter to nonlinear systems. In: Proceedings of the 1997 SPIE AeroSense Symposium, SPIE (Orlando, FL, April 21–24, 1997)
7. Julier, S., Uhlmann, J.: Unscented filtering and nonlinear estimation. Proceedings of the IEEE **92**(3) (March 2004) 401–22
8. Savage, C., Cramer, R., Schmitt, H.: TDOA geolocation with the unscented Kalman filter. In: Proc. 2006 IEEE International Conference on Networking, Sensing and Control. (April 2006) 602–6
9. Nørgaard, M., Poulsen, N., Ravn, O.: Advances in derivative-free state estimation for nonlinear systems. Technical rep. IMM-REP-1998-15, Dept. of Mathematical Modeling, Tech. Univ. of Denmark, 28 Lyngby, Denmark (April 2000)
10. Nørgaard, M., Poulsen, N., Ravn, O.: New developments in state estimation for nonlinear systems. Automatica **36**(11) (November 2000) 1627–38
11. van der Merwe, R.: Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models. PhD thesis, OGI School of Science & Engineering at Oregon Health & Science University (April 2004)
12. Hirsch, M., Ortiz-Pena, H., Cooperman, R.: Smoothing techniques using an IMM and multi-sensor tracking with time-late data. In: Proceedings of the National Symposium on Sensor and Data Fusion. (Laurel, MD, June 2004)
13. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes in C: The Art of Scientific Computing. second edn. Cambridge University Press (1992)
14. Stewart, G.: Matrix Algorithms. Volume I: Basic Decompositions. SIAM (1998)
15. Julier, S., Uhlmann, J.: A general method for approximating nonlinear transformations of probability distributions. Technical report, RRG, Department of Engineering Science, Oxford University (November 1996)
16. Burl, J.B.: Linear Optimal Control: H_2 and H_∞ Methods. Addison-Wesley, Menlo Park, CA (1999)