June 2015 Volume 10 Number 1 ISSN 1557-6418

JAIF 15

Journal of Advances in Information Fusion

A semi-annual archival publication of the International Society of Information Fusion

Regular Papers Page

INTERNATIONAL SOCIETY OF INFORMATION FUSION

The International Society of Information Fusion (ISIF) is the premier professional society and global information resource for multidisciplinary approaches for theoretical and applied INFORMATION FUSION technologies. Technical areas of interest include target tracking, detection theory, applications for information fusion methods, image fusion, fusion systems architectures and management issues, classification, learning, data mining, Bayesian and reasoning methods.

Journal of Advances in Information Fusion (ISSN 1557-6418) is published semi-annually by the International Society of Information Fusion. The responsibility for the contents rests upon the authors and not upon ISIF, the Society, or its members. ISIF is a California Nonprofit Public Benefit Corporation at P.O. Box 4631, Mountain View, California 94040.

Standard Bayesian Approach to Quantized Measurements and Imprecise Likelihoods

LAWRENCE D. STONE
STEPHEN L. ANDERSON

In this paper we show that the standard definition of likelihood function used in Bayesian inference simply and correctly handles quantized measurements and imprecise likelihood functions. Some recent papers have stated or implied that methods involving random sets, fuzzy membership functions, generalized likelihood functions, or Dempster-Shafer concepts are required to handle imprecise likelihood functions and quantized measurements. While it is true that one can use these methods, in the spirit of Occam's razor, we feel the simplest correct solution is the best.

Manuscript received on December 19, 2013; revised April 10, 2014 and April 22, 2014; released for publication May 11, 2014.

Refereeing of this contribution was handled by Jean Dezert.

This is an extended version of the paper "Standard Bayesian approach to quantized measurements and imprecise likelihoods" published at the 16th International Conference on Information Fusion (Fusion 2013).

Authors' address: Metron Inc., 1818 Library Street, Reston Virginia, 20190 (e-mail: stone@metsci.com and Anderson@metsci.com).

1 INTRODUCTION

Some recent papers have stated or implied that methods involving random sets, fuzzy membership functions, generalized likelihood functions, or Dempster-Shafer concepts are required to handle quantized measurements and imprecise likelihood functions. In particular, reference [3] considers the problem of constructing likelihood functions for quantized measurements and proposes that these types of measurements require a generalization of the standard notion of likelihood function that involves the use of random sets, concepts from fuzzy logic and Dempster-Shafer theory, as well as generalized or imprecise likelihood functions. Similarly, reference [4] presents examples of problems where the author claims that imprecise likelihood functions (a generalization of standard likelihood functions) are required.

The purpose of this paper is to show that the standard concept of likelihood function as defined in [1] or [6] is sufficient to solve the problems presented in [3] and [4] in an easy and straightforward manner. This is an important point because in the spirit of Occam's razor, we believe the simplest correct solution to a problem is the best one. Simplicity allows readers to clearly understand the nature of the problem and its solution. It facilitates the use of a concept in applications and makes it easier to extend it to more challenging problems. It enables progress.

This suggests the following question which we pose but do not presume to answer here: What situations involving quantized measurements or imprecise likelihood functions *require* the use of alternate, non-Bayesian models of uncertainty?

Section 2 of this paper presents the standard Bayesian inference formulation. Section 3 shows how this formulation can be used to handle the quantized measurement examples presented in [3]. Section 4 shows that the examples given in [4] can be readily handled using standard Bayesian likelihood functions and that a generalization to imprecise likelihood functions is not required for these examples.

Reference [2] investigates the problem of tracking a target with quantized measurements. The authors assume a Gaussian motion model for the target and develop an approximate Minimum Mean Squared Error (MMSE) solution. They provide a numerical algorithm for obtaining this solution. This is very impressive work, and one must admire the authors for the cleverness of their solution. However, the solution is complex and does require many special assumptions. In contrast to this we present in Section 5 a particle filter approach to solving this problem with standard Bayesian likelihood functions that is very simple and general. One is not constrained to Gaussian motion models or measurement errors. It is straight-forward to incorporate a wide variety of types of measurements and motion models. We present an example to illustrate this approach.

2 BAYESIAN INFERENCE FORMULATION

In order to clarify what we mean by a standard Bayesian approach, we give the formulation of the basic Bayesian inference problem that is presented in [6] and is consistent with that in [1].

There is an unknown parameter Θ that we wish to estimate. There is a prior distribution p_0 on Θ such that

$$p_0(\theta) = \Pr\{\Theta = \theta\} \tag{1}$$

where Pr indicates either probability or probability density as appropriate. We obtain a measurement Z from a sensor. The measurement is viewed as a random variable whose distribution depends on θ. We define the likelihood function

$$l(z \mid \theta) = \Pr\{Z = z \mid \Theta = \theta\}. \tag{2}$$

If we receive a measurement $Z = z$, we compute the posterior distribution

$$p_1(\theta \mid Z = z) = \frac{l(z \mid \theta)p_0(\theta)}{\int l(z \mid \theta')p_0(\theta')d\theta'} \tag{3}$$

where integration is replaced by summation if the distribution on Θ is discrete.

Note, when we use the term likelihood function to describe (2), we mean the function $l(z \mid \cdot)$ obtained by holding the measurement z fixed and letting the parameter θ vary. $l(z \mid \cdot)$ need not be a probability (density) function. It may integrate to a number different than 1. We use the notation l in place of the more usual p to emphasize this point.

3 QUANTIZED MEASUREMENTS

Reference [3] sought to illustrated the necessity of its approach by presenting examples of performing inference using quantized measurements. In this section, we use the same examples to show that standard likelihood functions and the Bayesian inference process as given in (1)–(3) provide a straight-forward and correct way of incorporating quantized measurements into Bayesian inference. No generalization is required, and no extensions of the standard Bayesian probability concepts are needed.

In the digital voltmeter example given in [3], measurements are taken by a digital voltmeter that provides voltage readings to two decimal places. From the digital voltmeter measurement, we wish to estimate the actual voltage Θ. Let p_0 be the prior on Θ. We consider three cases, measurements without noise, measurements with noise, and measurements where the quantization is unknown.

3.1 Quantized Measurements without noise—known quantization

If there is no noise added to the actual voltage, then any voltage in the interval $(199.975, 199.985]$ will produce a measurement of 199.98. The measurement space is a discrete set of points on the real line of the form $j \times 0.0$ where j is an integer such that $-\infty < j < \infty$. Any voltage in the set

$$S_j = (j \times 0.01 - 0.005, j \times 0.01 + 0.005] \tag{4}$$

will produce a measurement $Z = j \times 0.01$. From the definition of likelihood function in (2), we have

$$\begin{aligned} l(j \times 0.01 \mid \theta) &= \Pr\{Z = j \times 0.01 \mid \theta\} \\ &= \begin{cases} 1 & \text{if } \theta \in S_j \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

For notational convenience, we shall use $Z = j$ for the measurement and $l(j \mid \theta)$ for the likelihood function in (5).

The posterior on the actual voltage Θ is computed by,

$$\begin{aligned} p_1(\theta \mid Z = j) &= \frac{l(j \mid \theta)p_0(\theta)}{\int l(j \mid \theta')p_0(\theta')d\theta'} \\ &= \begin{cases} \dfrac{p_0(\theta)}{\int_{S_j} p_0(\theta')d\theta'} & \text{if } \theta \in S_j \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

3.2 Quantized Measurements with Noise—Known Quantization

In this example we suppose the received voltage r at the digital voltmeter is the true voltage θ plus noise ε. Specifically, the true voltage is $\theta + \varepsilon$ where ε has the density function

$$f(y) = \Pr\{\varepsilon = y\} \quad \text{for } -\infty < y < \infty.$$

The digital voltmeter produces measurements to two decimal places as above. In this case the likelihood function becomes

$$\begin{aligned} l(j \mid \theta) &= Pr\{Z = j \mid \theta\} \\ &= \Pr\{j \times 0.01 - 0.005 < \theta + \varepsilon \\ &\qquad \le j \times 0.01 + 0.005\} \\ &= \Pr\{j \times 0.01 - \theta - 0.005 < \varepsilon \\ &\qquad \le j \times 0.01 - \theta + 0.005\} \\ &= \int_{j \times 0.01 - \theta - 0.005}^{j \times 0.01 - \theta + 0.005} f(y)dy. \end{aligned} \tag{7}$$

and the posterior distribution on Θ given the measurement $Z = j$ is

$$\begin{aligned} p_1(\theta \mid Z = j) &= \frac{p_0(\theta)l(j \mid \theta)}{\int p_0(\theta')l(j \mid \theta')d\theta'} \\ &= \frac{p_0(\theta)\int_{j \times 0.01 - \theta + 0.005}^{j \times 0.01 - \theta + 0.005} f(y)dy}{\int p_0(\theta')\left(\int_{j \times 0.01 - \theta' - 0.005}^{j \times 0.01 - \theta' + 0.005} f(y)dy\right)d\theta'} \end{aligned}$$

As an example, let us consider the situation where

$$f(y) = \eta(y, 0, \sigma^2)$$

where $\eta(\cdot,0,\sigma^2)$ is the probability density function for the normal distribution with mean 0 and variance σ^2. The notation $\eta(\cdot,0,\sigma^2)$ is used to indicate the function of one variable obtained by fixing the values of the 2nd and 3rd variables at 0 and σ^2. We use a similar notation for a function of two variables when we wish to fix the value of one of the variables. Let

$$\Phi(z,\sigma^2) = \int_{-\infty}^{z} \eta(y,0,\sigma^2)dy \quad \text{for } -\infty < z < \infty. \quad (8)$$

Then the likelihood function $l(j \mid \theta)$ in (7) becomes

$$l(j \mid \theta) = \Phi(j \times 0.01 - \theta + 0.005, \sigma^2) - \Phi(j \times 0.01 - \theta - 0.005, \sigma^2). \quad (9)$$

Figure 1 shows plots of the likelihood function in (9) for $Z = 10$ and $\sigma^2 = 0.0001$, 0.0004, and 0.0016.

Reference [4] also presents a quantized measurement example that the author claims requires the use of imprecise likelihood functions. In fact one can produce the example in [4] and obtain Figure 1 in [4] if he considers the case where the quantization has bin-size 15 and the measurement $Z = j$ indicates the interval $(15j, 15(j+1)]$. In this case the sets S_j in (4) become

$$S_j = (15j, 15(j+1)],$$

and the likelihood function in (9) becomes

$$l(j \mid \theta) = \Phi(j \times (15+1) - \theta, \sigma^2) - \Phi(j \times 15 - \theta, \sigma^2)$$

where θ plays the role of the variable z in [4].

3.3 Quantized Measurements when Quantization is Unknown

In the case where the quantization is unknown, we expand the state space on which we perform inference to simultaneously estimate the voltage and the quantization. To illustrate how this is done within the conventional Bayesian inference formalism, we consider the case where there is no noise added to the voltage and the quantized bins have a known and equal size. However, we do not know the anchor point for the bins.

Following Example 1, we take the bin size to be 0.01. However we do not know the anchor point of the bins. Specifically there is a unknown parameter Δ such that $-0.005 \le \Delta \le 0.005$ and

$$S_j(\Delta) = (j \times 0.01 - 0.005 + \Delta, j \times 0.01 + 0.005 + \Delta].$$

The inference problem is to estimate both θ and Δ.

In classic Bayesian fashion, we impose a prior distribution on (θ,Δ) which represents our prior knowledge (or uncertainty) about (θ,Δ). As an example we suppose that the priors on the two parameters are independent and the joint density on (θ,Δ) is given by

$$g_0(\theta,\delta) = p_0(\theta)q_0(\delta) \quad \text{for } -\infty < \theta < \infty \ -0.005 < \delta \le 0.005.$$

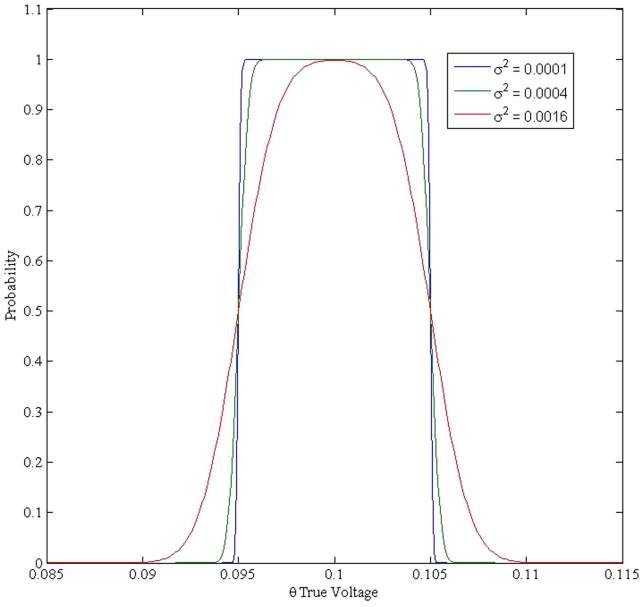


Fig. 1. Likelihood functions for $j = 10$ (0.1 volt reading on voltmeter) when $\sigma^2 = 0.0001$, 0.0004, and 0.0016.

The likelihood function for the observation $Z = j \times 0.01$ is

$$l(j \mid (\theta,\delta)) = \Pr\{Z = j \times 0.01 \mid (\Theta,\Delta) = (\theta,\delta)\} = \begin{cases} 1 & \text{if } \theta \in S_j(\delta) \\ 0 & \text{otherwise.} \end{cases}$$

Let g_1 be the posterior joint density on (θ,Δ) given $Z = j$. Then

$$g_1((\theta,\delta) \mid Z = j) = \frac{l(j \mid (\theta,\delta))p_0(\theta)q_0(\delta)}{\int_{-0.005}^{0.005}\int l(j \mid (\theta',\delta'))p_0(\theta')q_0(\delta')d\theta'd\delta'} = \begin{cases} \dfrac{p_0(\theta)q_0(\delta)}{\int_{-0.005}^{0.005}\int_{S_j(\delta')} p_0(\theta')q_0(\delta')d\theta'd\delta'} & \text{for } \theta \in S_j(\delta) \\ 0 & \text{otherwise.} \end{cases}$$

3.4 Alternate Quantization Models

Digital signal processing involves quantized measurements. The effect of this quantization is sometimes modeled as adding random and independent noise to the measurements. This is discussed in [7] which notes "It has been shown to be a valid model in cases of high resolution quantization (small Δrelative to the signal strength) with smooth probability density functions. However, additive noise behaviour is not always a valid assumption, and care should be taken to avoid assuming that this model always applies. In actuality, the quantization error…is deterministically related to the signal rather than being independent of it."

4 IMPRECISE LIKELIHOOD FUNCTIONS

In this section we show that the localization example given in IV of [4] can be computed in a straight-

forward way by standard Bayesian likelihood functions without resorting to imprecise likelihood functions or other generalizations of Bayesian inference.

For the convenience of the reader we reproduce the description of the example given in [4].

4.1 Localization using RSS

Received signal strength (RSS) is often used for localizing an emitting energy source although the source level of the emitter is unknown. As an example, consider the two-dimensional situation shown in Figure 2 where the unknown source position $X = (X_1, X_2)$ is located inside the square defined by

$$5 \le X_1 \le 95; \quad 5 \le X_2 \le 95, \tag{10}$$

and the prior on distribution on (X_1, X_2) is uniform over this square. The unknown source level A (in dB) of the emitter has a uniform distribution on $[25, 65]$. There are 12 receivers uniformly spaced on a circle of radius 50 centered at $(50, 50)$ as shown by the squares in Figure 2. The receivers are numbered in counter-clockwise fashion starting with receiver 1 at the 3 o'clock position. Let (x_1^i, x_2^i) be the location of the ith receiver.

For $i = 1, \ldots, 12$, the measurement Z_i of RSS at the ith receiver satisfies the following equation

$$Z_i = A - 10\theta_i \log(d_i(x)/d_0) + v_i \tag{11}$$

where

$$d_i(x) = \sqrt{(x_1^i - x_1)^2 + (x_2^i - x_2)^2}$$

is the distance from the location of the ith receiver to the source given the source is located at $x = (x_1, x_2)$,

θ_i is an unknown propagation loss factor where $2 \le \theta_i \le 4$,

$d_0 = 10$ is a reference distance,

v_i has a Gaussian distribution with mean 0 and variance 4,

v_i is independent of v_j for $i \ne j$.

4.1.1 Likelihood Function

Reference [4] does not provide an explicit formula for the likelihood function employed in this example. For our computations, we assume that θ_i is uniformly distributed over $[2.0, 4.0]$ for each receiver and that θ_i is independent of θ_j for $i \ne j$ and obtain an explicit likelihood function as follows.

Let η_{04} be the density function for a Gaussian distribution with mean 0 and variance 4. Then

$$\begin{aligned} &\Pr\{Z_i = z \mid X = x, A = a, \text{ and } \theta_i = \theta\} \\ &\quad = \eta_{04}(z - a + 10\theta \log(d_i(x)/d_0)) \end{aligned}$$

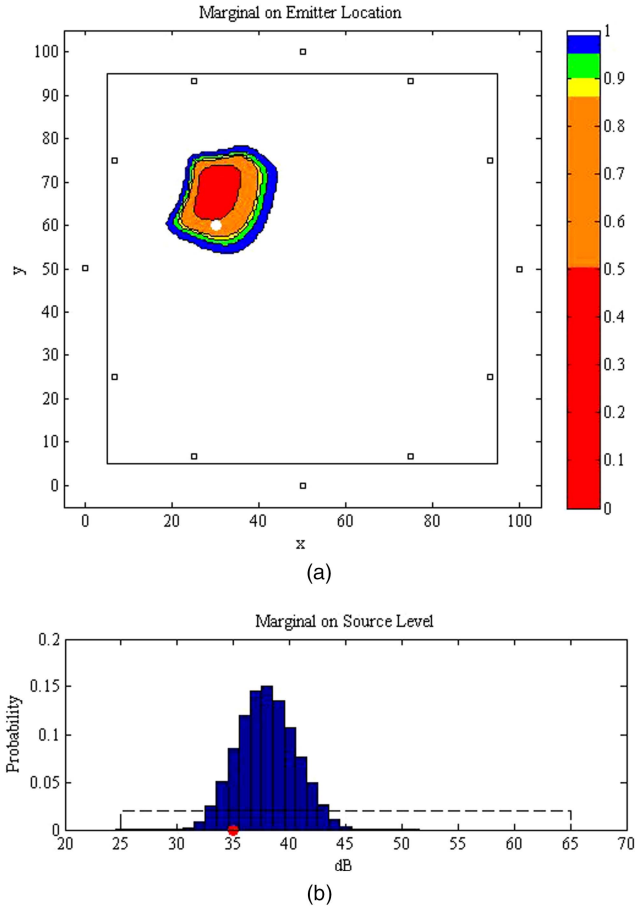


Fig. 2. Marginal distributions on position and source level.

and

$$\begin{aligned} l(z \mid x, a) &= \Pr\{Z_i = z \mid X = x, A = a\} \\ &= \frac{1}{2}\int_2^4 \eta_{04}(z - a + 10\theta \log(d_i(x)/d_0))d\theta \\ &= \frac{\int_{20\log(d_i(x)/d_0)}^{40\log(d_i(x)/d_0)} \eta_{04}(z - a + y)dy}{20\log(d_i(x)/d_0)} \\ &= \frac{\Phi(z - a + 40\log(d_i(x)/d_0), 4)}{20\log(d_i(x)/d_0)} \\ &\quad - \frac{\Phi(z - a + 20\log(d_i(x)/d_0), 4)}{20\log(d_i(x)/d_0)} \end{aligned} \tag{12}$$

is the likelihood for the measurement $Z_i = z$ given $X = x$ and $A = a$.

4.1.2 Results

Following [4] we simulated measurements at the 12 receivers using the model in (11) for a source located at $X = (30, 60)$ with source level $A = 35$. We set $\theta_i = 2.3$ for $i = 1, \ldots, 6$ and $\theta_i = 3.5$ for $i = 7, \ldots, 12$. Using the resulting measurements and the likelihood function in (12), we computed the joint posterior distribution on emitter position X and source level A. Figure 2 shows the resulting marginals on X and A. For numerical convenience we computed the distributions on a grid that

has 200 by 200 cells in position and 41 cells in source level. We calculated the posterior probability in each of these cells given the measurements from the 12 sensors. The color bar next to the position marginal indicates containment. The red region is the 50% containment region (i.e., the region with the smallest number of cells that contains 50% probability). The red plus orange region is the 86% containment region, and so on for the yellow, green, and blue regions. Observe that the emitter's location (shown as a white dot in Figure 2) is close to edge of the 50% containment region. The marginal on source level is represented by a bar graph with the height of the bar being equal to the posterior probability in the cell containing the bar. The cells are 1 dB in width. The actual source level, 35 dB, is in a reasonable location in this distribution.

In order to test whether the above method produces a good representation of the uncertainty in the posterior estimate of the position X of the emitter, we followed the approach of Section V in [4] and simulated 1000 sets of measurements at the 12 receivers. For each replication of the simulation, we made an independent draw for the value of θ_i from a uniform distribution over $[2.0, 4.0]$ and computed the resulting measurement Z_i from (11) for $i = 1, \ldots, 12$. The location of the emitter and source level remained fixed at $X = (30, 60)$ and $A = 35$ for all replications. For each replication, we computed the marginal on emitter position as in Figure 2. To test whether the resulting distributions correctly represent the uncertainty in the location of the emitter, we computed the following Kolmogorov-Smirnov (KS) graph.

For the nth replication, we started with the cell containing the emitter (the one containing the point $(30, 60)$) and summed the probability in all cells having probability greater than or equal to the probability in the emitter cell. This produced a containment region and containment probability c_n for $n = 1, \ldots, 1000$. Next we ordered the containment probabilities into a set $\{\hat{c}_n; n = 1, \ldots, 1000\}$ such that $0 \le \hat{c}_1 \le \hat{c}_2 \ldots \le \hat{c}_{1000} \le 1$. We then plotted the points $(n/1000, \hat{c}_n)$ for $n = 1, \ldots, 1000$. This plot is shown in blue in Figure 3 which is the empirical distribution for the containment probability produced by the marginal distribution on position. If the marginal distributions accurately represent the uncertainty in emitter location, this distribution should converge to the red straight line in Figure 3 as the number of replication increases to infinity. That is, the percentage of replications in which the target is inside the p percent containment region should be p percent. One can see that the empirical distribution is indeed a close fit to the straight line indicating an accurate representation of the uncertainty in the marginal. One could perform a KS test to test the hypothesis that the empirical distribution is the same except for sampling noise as the straight line. However, it is clear from Figure 3 that the fit is very good.

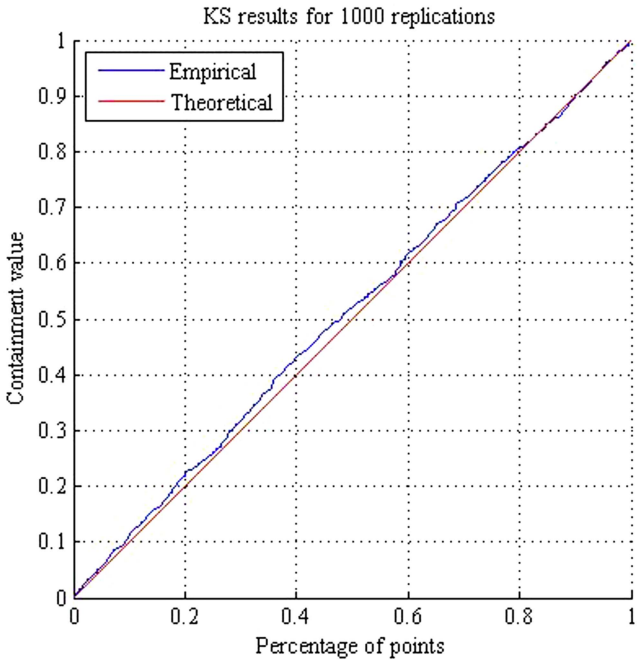


Fig. 3. KS graph for marginal distribution on emitter location.

To further test the accuracy of the marginal posterior on location, we performed another set of 1000 replications similar to the ones above. For each of these replications, we made an independent draw from a uniform distribution over the square defined by (10) for the location of the emitter and an independent draw from a uniform distribution over $[25, 65]$ for the source level. As above, we made independent draws for the values of $\theta_i$for $i = 1, \ldots 12$. We then computed the KS graph for the posterior marginal distribution on emitter location. The result looked very similar to Figure 3 further confirming the accuracy of these posterior distributions.

5 TRACKING MOVING TARGETS WITH QUANTIZED MEASUREMENTS

The target considered in Section 4 is stationary. One can also track moving targets using sensors with quantized measurements. Since the measurements do not satisfy the linear-Gaussian assumptions required for a Kalman filter, we perform the tracking using a particle filter as described in [5] or Chapter 3 of [6].

The general procedure is straightforward. The particles are motion updated to the time of a measurement. The likelihood function for a quantized measurement is applied to the weight of each particle to produce the posterior distribution on target state. The particles are resampled and then motion updated to the time of the next measurement.

In particular, suppose that the target state distribution at time t is represented by the set of particles

$$\{(x_n(t), w_n(t)\} \quad \text{for } n = 1, \ldots, N$$

where $x_n(t)$ is the state of the nth particle at time t and w_n is its probability. This set of particles represents a discrete probability approximation to the distribution on target state at the time t. Suppose we obtain a quantized measurement $Z = z$. Let $l(z \mid x)$ be the likelihood function for this measurement. The posterior distribution on target state at time t is given by

$$\{(x_n(t), \tilde{w}_n(t)\} \quad \text{for } n = 1, \ldots, N \tag{13}$$

where

$$\tilde{w}_n(t) = \frac{l(z \mid x_n(t))w_n(t)}{\sum_{n'=1}^{N} l(z \mid x_{n'}(t))w_{n'}(t)}.$$

If the next measurement is received at time $t' > t$, the posterior particle filter representation in (13) can be motion-updated to the time t' to act as a proposal distribution for the incorporation of the measurement at time t'. The posterior in (13) is typically resampled before the motion update is performed. If desired, other proposal distributions can used to improve particle filter performance as discussed in [5].

In the following example we consider an underwater acoustic detection situation that goes beyond the quantized measurement examples considered in [3]. In particular, the bins are unions of disjoint intervals. In this example we employ a particle filter to track a moving target.

5.1 Likelihood Function for Acoustic Detection

When a passive acoustic sensor is located in a deep water region of the ocean, the sound propagation conditions often produce detection areas that are disjoint. For example, there may be good detection conditions from the sensor's location out to range 5 nm. This is typically called the direct path region. In addition there are often convergence zone regions at ranges of roughly 30 nm, 60 nm, and even farther out. A convergence zone is a region where the acoustic rays converge and produce low propagation loss and increased detection probability for the sensor. Suppose the convergence zones are 5 nm wide. It is often the case that the uncertainty about the source level of a potential target means that although one cannot calculate the detection probability as a function of range, one does know that if a target has been detected, it is in one of these zones. In this case, a detection means that the target is in one of the above range intervals, i.e., its range is in the union of disjoint intervals

$$S = [0,5] \cup [27.5, 32.5] \cup [57.5, 62.5]. \tag{14}$$

Generally one does not know the edges of the intervals in S exactly. Depending on the source level of the target and the ambient noise in the ocean, these areas can be a bit larger or smaller than the nominal numbers in (14). We will model this uncertainty with a likelihood function that is similar to the one given in the example in Section 3.2 with the exception that there is only one bin corresponding to a detection, which we denote by $Z = 1$. Specifically we let r denote the range of the target and ε_i be mutually independent normally distributed random variables with mean 0 and variance σ_i^2 for $i = 1,2,3$. Then the likelihood function l_d for a detection becomes

$$\begin{aligned} l_d(1 \mid r) &= \Pr\{Z = 1 \mid \text{target at range } r\} \\ &= \Pr\left\{ \begin{array}{c} r \le 5 + \varepsilon_1 \text{ or } 27.5 - \varepsilon_2 \le r \le 32.5 + \varepsilon_2 \\ \text{or } 57.5 - \varepsilon_3 \le r \le 62.5 + \varepsilon_3 \end{array} \right\} \\ &\approx \begin{cases} \Pr\{r \le 5 + \varepsilon_1\} & \text{for } 0 \le r \le 20 \\ \Pr\{27.5 - \varepsilon_2 \le r \le 32.5 + \varepsilon_2\} & \text{for } 20 < r \le 45 \\ \Pr\{57.5 - \varepsilon_3 \le r \le 62.5 + \varepsilon_3\} & \text{for } 45 < r < \infty. \end{cases} \end{aligned} \tag{15}$$

The approximation in the last line is essentially an equality if $\sigma_i^2 < 4$ for $i = 1,2,3$. In terms of Φ defined in (8), the likelihood function in (15) becomes

$$l_d(1 \mid r) = \begin{cases} 1 - \Phi(r - 5, \sigma_1^2) & \text{for } 0 \le r \le 20 \\ \min\{1 - \Phi(27.5 - r, \sigma_2^2), 1 - \Phi(r - 32.5, \sigma_2^2)\} & \\ & \text{for } 20 < r \le 45 \\ \min\{1 - \Phi(57.5 - r, \sigma_3^2), 1 - \Phi(r - 62.5, \sigma_3^2)\} & \\ & \text{for } 45 < r < \infty. \end{cases} \tag{16}$$

Figure 4 shows the likelihood function $l_d(1 \mid \cdot)$ when $\sigma_i^2 = 0.5$ for $i = 1,2,3$.

5.2 Acoustic Tracking Example

For this example we consider a target moving at 14 kn in a 70 nm by 70 nm square as shown in Figure 5. There are six sensors located at $(0,0)$, $(0,35)$, $(0,70)$, $(70,0)$, $(70,35)$, and $(70,70)$. These are shown as white dots in Figure 5. Each of these sensors has the detection characteristics described in Section 5.1 and in particular has the detection likelihood function given by and plotted in Figure 4. There is an independent detection opportunity for each of these sensors every 5 minutes. The detection opportunities occur simultaneously for all sensors. Note that the detections produce range information but no bearing information.

In order to show the coverage of the sensor field over the 70 nm $\times$ 70 nm square, we computed the sum of the likelihood functions for the six sensors evaluated at each point in the square. This sum yields the expected number of sensors that can detect a target at each point in the square. The results are color-coded and plotted on the square in Figure 5. The color code is given by the color bar on the right. Note, this is not a calculation of the likelihood function resulting from the detections at an opportunity time. That likelihood function is obtained by pointwise *multiplying* the likelihood functions for the sensors obtaining a detection at that time.

For the example, the target follows the track shown in white in Figure 5 moving at a constant 14 kn from the

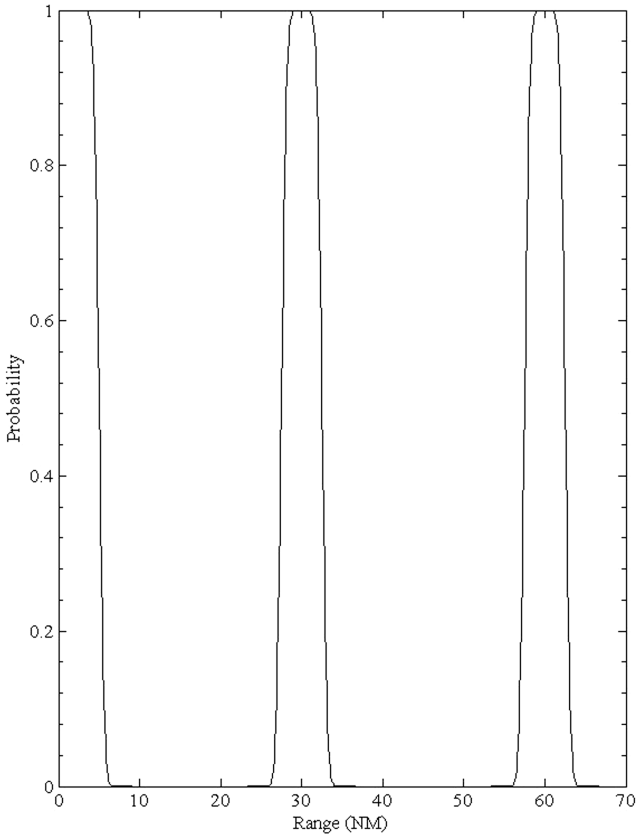


Fig. 4. Likelihood function $l_d(1 \mid \cdot)$ for accoustic detection in a convergence zone environment.

bottom to the top of the figure. The maneuver occurs at 2.5 hours.

Using the likelihood function in (16) and a particle filter that is a minor modification of the one described in Section 1.3 of [6], we estimated the track of this moving target. We simulated detections as follows. At each opportunity time, the simulation calculated the range of the target from each sensor. A detection was called with probability equal to the likelihood function value at that range. Detections are independent from sensor to sensor.

We used 25,000 particles. The particle paths were initialized from the first sensor detection as follows. For each particle, the range was randomly drawn from a uniform distribution on the convergence zone intervals with a Gaussian component having mean 0 and variance 0.5 nm added. The bearing from the sensor was chosen uniformly over the interval 0 to 360 degrees. By doing this we obtained 25,000 equally weighted independent points from the posterior distribution on target position given the first detection.

The initial speed for each particle was drawn from a uniform distribution on the interval 2 to 30 kn. The initial course was drawn from a uniform distribution over the interval 0 to 360 degrees. The particles change velocity according to an exponential distribution with mean 0.5 hours. When a velocity change takes place, a new velocity is chosen from a distribution that produces

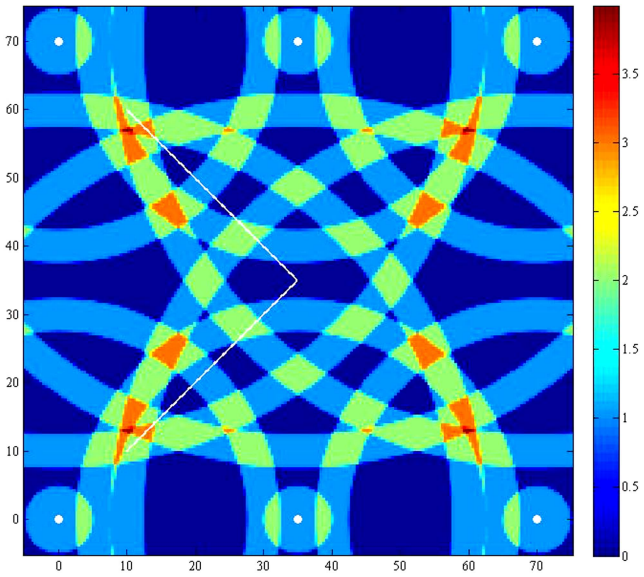


Fig. 5. Sensor field and target track.

a mean change of 30 degrees in course and 2 kn in speed. See Section 1.3.3 of [6] for the details of this motion model.

Figures 6 and 7 show the filter output at 0.5 and 2.0 hours. In the figures, we show only a 500 point sample of the particles in the figure. At 0.5 hours, the distribution has two modes. By 2.0 hours the second mode has disappeared, and the remaining particle cloud is centered at the target's location.

Figures 8 and 9 show the filter output at 3 hours and 5 hours. At 3 hours, just after the maneuver, the particle distribution has spread out. By 5 hours it has condensed around the position of the target.

This example shows two things. First that even complicated quantized measurements can be represented by standard likelihood functions, and second these likelihood functions can be easily used in a particle filter to track moving targets.

6 CONCLUSIONS

The examples have shown how to construct likelihood functions for quantized measurements using the standard Bayesian approach with standard likelihood functions.

In Section 3, we showed that the quantized measurement examples presented in [3] can be treated without employing the notions of generalized likelihood functions.

In Section 4 we have shown that the examples presented in [4] can be handled with standard Bayesian likelihood functions without the extra complexity required for imprecise likelihood functions. If anything, the results presented in Section 4.1.2 are somewhat better than the ones obtained by the use of imprecise likelihood functions in [4].

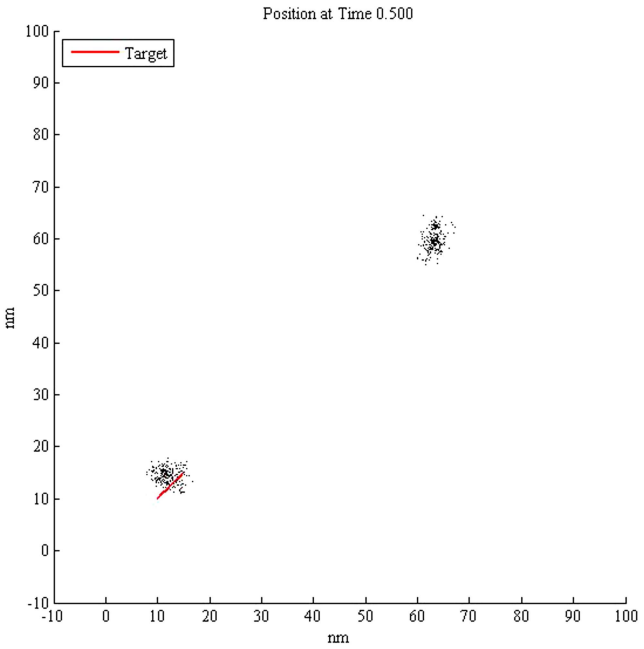


Fig. 6. Position marginal at 0.5 hr.

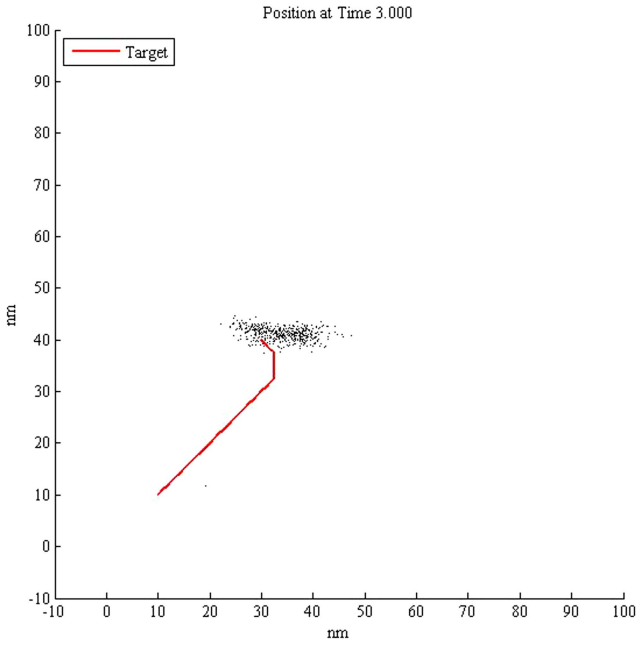


Fig. 8. Postion marginal at 3 hr.

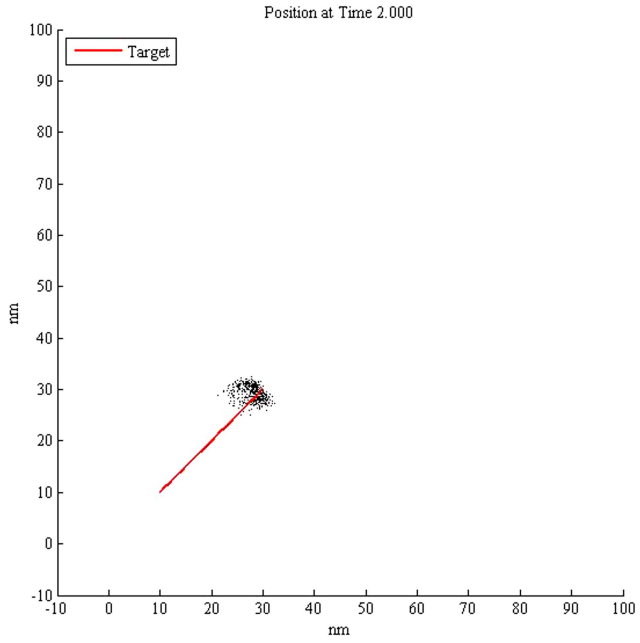


Fig. 7. Position marginal at 2.0 hr.

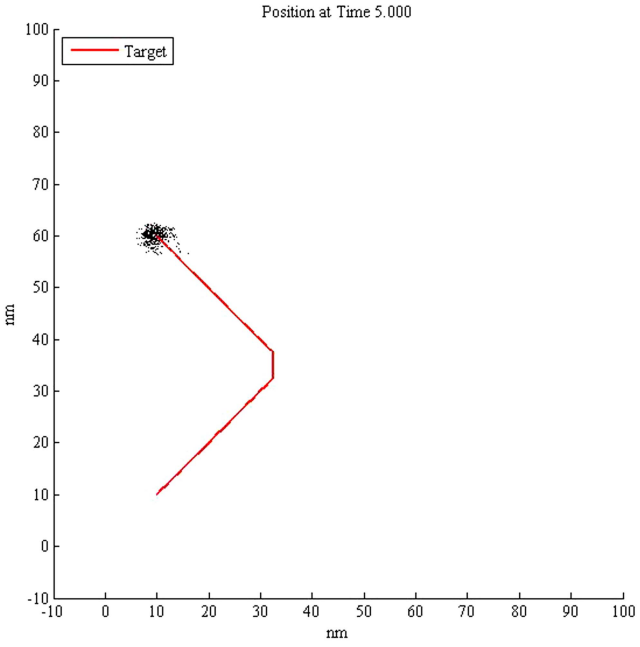


Fig. 9. Postion marginal at 5 hr.

In Section 5, we have shown that quantized measurements can be applied to moving target tracking problems using particle filters and likelihood functions for the quantized measurements in a straightforward, standard Bayesian fashion.

The power of a likelihood function is that it converts measurements from (almost) any measurement space into a function on the target state space. This allows us to incorporate the information in these measurements into the posterior distribution on the target state space. The examples given above illustrate this process with quantized measurements, but the method is applicable to wide range of types of measurements and sensors. In particular, it is applicable to any measurement for which one can compute a likelihood function using the definition in (2). This is why likelihood functions are the common currency of information in Bayesian inference. The examples given above demonstrate this fact.

We have discussed above the virtues of using the simplest solution to a problem. We would be remiss if we did not also point out that there can be drawbacks to unnecessary complexity. For example, if we employ Dempster Shafer methods to handle quantized measurements, then we will be limited in applications to finite

discrete state spaces since there has been no satisfactory extension of Dempster-Shafer theory to continuous state spaces. Even if the state space is finite, the computations involved with Dempster-Shafer methods grow exponentially with the size of the state space which limits its applicability to real problems.

REFERENCES

[1] J. O. Berger
Statistical Decsion Theory and Bayesian Analysis, 2nd ed.
New York, NY: Springer-Verlag, 1985.

[2] Z. Daun, V. P. Jilkov, and X. R. Li
State estimation with quantized measurements: approximate MMSE approach.
In *Proceedings of 11th International Conference on Information Fusion*, Cologne, Germany, July 2008, 1067–1072.

[3] R. Mahler
General Bayes filtering of quantized measurements.
In *Proceedings of 14th International Conference on Information Fusion*, Chicago, USA, July 5–8, 2011, 346–352.

[4] B. Ristic
Bayesian estimation with imprecise likelihoods: Random set approach.
IEEE Signal Processing Letters, 18 (July 2011), 394–398.

[5] B. Ristic, S. Arulampalam, and N. Gordon
Beyond the Kalman Filter.
Boston, MA: Artech House, 2004.

[6] L. D. Stone, R. L. Streit, T. L. Corwin, and K. L. Bell
Bayesian Multiple Target Tracking 2nd ed.
Boston, MA: Artech House, 2014.

[7] http://en.wikipedia.org/wiki/Quantization.

Lawrence D. Stone obtained his B.S. in mathematics from Antioch College in 1964 and his M.S. and Ph.D. in mathematics from Purdue University in 1965 and 1967. He is Chief Scientist at Metron Inc. He is a member of the National Academy of Engineering and a fellow of the Institute for Operations Research and Management Science. In 1975, the Operations Research Society of America awarded the Lanchester Prize to Dr. Stone's text, *Theory of Optimal Search*. In 1986, he produced the probability maps used to locate the *S.S. Central America* which sank in 1857, taking millions of dollars of gold coins and bars to the ocean bottom one and one-half miles below. In 2010 he led the team that produced the probability distribution that guided the French to the location of the underwater wreckage of Air France Flight AF447. He is a coauthor of the 2014 book, *Bayesian Multiple Target Tracking*. He continues to work on a number of detection and tracking systems for the United States Navy and Coast Guard including the Search And Rescue Optimal Planning System used by the Coast Guard since 2007 to plan searches for people missing at sea.

Stephen L. Anderson obtained his B.A. in mathematics from the University of Utah in 1975 and his Ph.D. in mathematics from Brown University in 1980. He is a senior analyst at Metron Inc. His current work is primarily in the development of algorithms to assist in finding isolated personnel such as down pilots. The research focuses on developing motion models and applying search theory to optimize recovery efforts. From 1998 to 2010 Dr. Anderson was the principal developer of the Nodestar algorithm for use aboard US Navy fast attack submarines. The core of Nodestar is a non-linear data fusion methodology for multiple target tracking. The algorithm's capabilities have expanded to include all of the acoustic and non-acoustic organic sources available aboard the submarines. Dr. Anderson has also applied non-linear filtering theory to meteorological problems. He is co-author (with Jeffrey Anderson) of "A Monte Carlo Implementation of the Nonlinear Filtering Problem to Produce Ensemble Assimilations and Forecasts" which appeared in the Monthly Weather Review of the American Meteorological Society in December 1999. In 1995 Dr. Anderson participated in the analysis of flight information databases to develop delay models for the FAA. Additionally, in 1996 he helped in the development analytic models for the effects of unscheduled FAA outages.

Prior to joining Metron, Dr. Anderson was an Associate with Daniel H. Wagner, Associates. For two years, he was the coordinator of the NRL Ocean Surveillance Tracker/Correlator Testbed. In this capacity, he directed the evaluation of ocean surveillance algorithms in all stages of development. The evaluation included review of the algorithm design, verification that the algorithm as implemented reflected the design, and testing of the algorithm on simulated data. Additionally, he participated in the development of an ocean surveillance correlator/tracker used as a baseline in the evaluation process.

Dr. Anderson directed a similar effort for evaluation of Strategic Defense Initiative correlator/tracker algorithms. The basic modules which he designed and assisted in coding are still in use in modified forms, including a version which was used in parallelization studies at Sandia National Laboratory to simulate ten thousand targets. Also for missile defense, he assisted in the development of an algorithm for merging tracks from multiple independent tracking algorithms. Additionally, he was responsible for the testing to verify the basic soundness of the overall concept.

Symmetries in Bayesian Extended Object Tracking

FLORIAN FAION
ANTONIO ZEA
MARCUS BAUM
UWE D. HANEBECK

In this work, we exploit geometric symmetries in extended objects in order to improve Bayesian tracking algorithms that use Spatial Distribution Models, Greedy Association Models as used in curve fitting, and Random Hypersurface Models. The key idea is to describe symmetric objects by solely modeling the non-redundant part of the shape, while the remainder of the shape follows from symmetry. Following this idea, we develop simplified versions for the three models that take advantage of the symmetry. Exploiting symmetries yields two major benefits. First, complex symmetric shapes can be equivalently represented by a fraction of the original shape parameters. Second, when using sample-based filters, such as the widely used Unscented Kalman Filter, symmetry yields a higher effective sample resolution. It is worth mentioning that estimating even simple objects such as a stick, which only have one reflectional symmetry, can be significantly improved.

Manuscript received November 22, 2013; revised June 14, 2014; released for publication September 13, 2014.

Refereeing of this contribution was handled by Peter Willett.

Authors' addresses: F. Faion, A. Zea, and U. Hanebeck, Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), Germany (e-mail: florian.faion@kit.edu, antonio.zea@kit.edu, uwe.hanebeck@ieee.org). M. Baum, University of Connecticut, USA, (e-mail: baum@engineer.uconn.edu).

1 INTRODUCTION

Tracking shape and pose of extended objects based on point measurements is a well-studied task that spans many different fields of research including robotics, human machine interaction, and surveillance. In the context of tracking, the difference between a classical object and an extended object is that the former is, due to its size or sensor resolution, assumed to produce measurements from one distinct source on the shape, while the latter is assumed to produce measurements from more than one source. Depending on the specific task, there are different sensors available to capture point measurements of a given target object. These include laser scanners, depth cameras, and radars, all of which produce data that differ in both the quantity and the quality (i.e. uncertainty) of the measured points. In addition to these sensor-specific characteristics, (self-)occlusions and unknown association of measured points to its generating sources on the object are the main challenges when tracking an extended object.

Many objects in everyday life have known geometric symmetries and incorporating them into tracking algorithms makes sense for two reasons. First, it allows for modeling only a small part of the shape, and then using symmetry to obtain the other parts, which can be exploited to reduce the number of required shape parameters. Second, as point measurements from one part of the shape also contain information about the symmetric counterparts the estimator becomes more robust against occlusions. However, symmetric transformations have to be applied carefully, as they may produce unintended side effects in the measurement model.

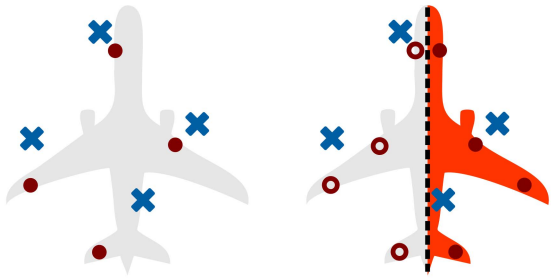


Fig. 1. Common tracking algorithms (a) require a generative model for measurements (blue crosses) that considers sources (red filled circles) over the entire shape. The proposed approach (b) only requires modeling sources in a non-redundant part (orange), so that the rest (red empty circles) of the shape follows from symmetry.

Fig. 1 illustrates the key idea of incorporating symmetry. While common tracking approaches assume measurements originating from sources on the entire airplane (Fig. 1(a)), we propose to model measurement sources just on a single side of the shape (orange), and obtain the rest by mirroring (Fig. 1(b)) in this case. We will denote this orange region as the *non-redundant part* of the shape.

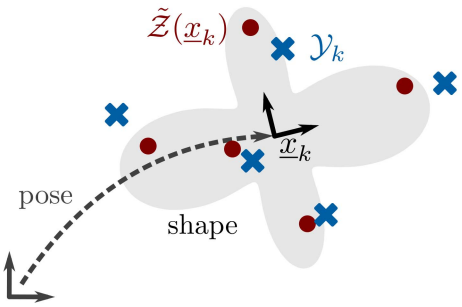


Fig. 2. *Given*: Noisy point measurements $\mathcal{Y}_k$ of sources $\tilde{\mathcal{Z}}(\underline{x}_k)$ on the object, measured at time k. *Desired*: Object parameters $\underline{x}_k$ for the specified model.

Our intention is to explore how different generative models as commonly used in Bayesian extended object tracking can take advantage of these symmetric considerations. Specifically, we will focus on Spatial Distribution Models [9], [24], Greedy Association Models [4], [7] as used in curve fitting, and Random Hypersurface Models [1–3], [6]. Sample-based filtering techniques [10], [23] will be used for implementation.

There are many related approaches that exploit geometric symmetries, e.g., surveyed in [16] and [17]. In the context of [17], our work would be classified as *model acquisition and representation*. Related work on tracking is proposed in [8], where random matrices are used to describe ellipsoidal and non-ellipsoidal [13] extended objects which produce measurement sources according to a known distribution. In [15], reflectional symmetry is incorporated into an image-based tracking algorithm that estimates the bounding box of a moving symmetric object. Reflectional symmetry has also been exploited for segmentation purposes [19]. Simplifying a symmetric mesh was proposed in [22], where the authors incorporate symmetry in the data structure of the mesh in order to remove redundancies. In [20], the authors even proceed one step further by generating a complex shape by back-projecting silhouettes. Treating symmetric multimodalities with directional statistics [12] is also a related field of research, as well as symmetric measurement functions [14] that consider symmetry in the sense of invariance to switching specific parts of the state vector. Note that, while other approaches which exploit geometric symmetries mostly work with image sequences of the object, we only use sparse point measurements. As such, and to the best of our knowledge, this is the first approach to explicitly incorporate geometric symmetries into Bayesian extended object tracking.

The paper is structured as follows. In Sec. 2, the mathematical problem is stated and the model parameters are introduced. Bayesian extended object tracking is discussed in Sec. 4, followed by the key idea of our approach in Sec. 5. For the considered symmetries, as explained in Sec. 6, simplification is further elaborated in Sec. 7 in order to derive a general simplification scheme. The implementation of the proposed approach then is described in Sec. 8. We illustrate these concepts by deriving symmetric models for sticks (Sec. 9) and symmetric star-convex shapes (Sec. 10). These models are evaluated in Sec. 11 and compared against their non-simplified paragons. Finally, conclusions are drawn in Sec. 12.

2 PROBLEM STATEMENT

Tracking an extended object consists of estimating the object state $\underline{x}_k$ as a combination of pose (position and orientation) and shape parameters[1]

$$\underline{x}_k = \begin{bmatrix} \underline{x}_k^{\text{pose}} \\ \underline{x}_k^{\text{shape}} \end{bmatrix}, \tag{1}$$

at each time step k, based on a list of n_k given noisy point observations

$$\mathcal{Y}_k = \{\underline{y}_{i,k} \mid i = 1,\ldots,n_k\} \tag{2}$$

from the object. In general, the dimension d of these points is two or three. The potentially time-varying number n_k of measurements is considered to not contain any information about the object extent. We assume that the measurements originate from source points $\tilde{\underline{z}}_{i,k} \in \tilde{\mathcal{Z}}(\underline{x}_k)$ on the object corrupted by noise in the form

$$\underline{y}_{i,k} = \tilde{\underline{z}}_{i,k} + \underline{w}_{i,k}. \tag{3}$$

The additive noise term $\underline{w}_{i,k}$ is assumed to be drawn from the zero-mean Gaussian distribution $p(\underline{w}_{i,k}) = \mathcal{N}(\underline{0}, \mathbf{C}_{\underline{w}_{i,k}})$, and assumed to be independent from the state and the measurement source. In the following, we denote the set of all possible measurement sources $\tilde{\mathcal{Z}}(\underline{x}_k)$ as the *shape* of the object. Fig. 2 shows a sketch of the estimation task and involved parameters. Note that i) considerations on clutter measurements are not included in this paper, and ii) time indices k and measurement indices i will be dropped when not needed.

2.1 Separation of Shape and Pose

In order to keep geometric considerations simple, we separate pose and shape by using two coordinate frames (see Fig. 2): the *object frame* that represents the object without any pose information, i.e., unrotated and centered on the origin, and the *world frame* that represents how the object is seen from the outside. The function H transforms a given point $\underline{z}^W$, given in the world frame, to the point $\underline{z}^O$ in the object frame, by means of the rigid transformation

$$\underline{z}^O = H(\underline{x}, \underline{z}^W) = \mathbf{R} \cdot \underline{z}^W + \underline{t}, \tag{4}$$

where $\mathbf{R}$ is a rotation matrix and $\underline{t}$ is a translation vector, both derived from $\underline{x}$. Unless otherwise stated, all geometric considerations are assumed to refer to the object frame.

[1] Note that the state can easily be extended to include further parameters.

3 NOTATION

We denote vectors with underline $\underline{x}$ and matrices with capital bold letters $\mathbf{C}$.

$\underline{x}$	object state vector
$\underline{z}$	points $\underline{z} \in \mathbb{R}^d$ in Cartesian coordinates
$\tilde{\mathcal{Z}}(\underline{x})$	shape of the object (set of all measurement sources $\tilde{\underline{z}} \in \tilde{\mathcal{Z}}(\underline{x})$) for a given state $\underline{x}$
$\tilde{\underline{z}}(\underline{x}, s)$	measurement sources $\tilde{\underline{z}} \in \tilde{\mathcal{Z}}(\underline{x})$ in function of the state $\underline{x}$ and index parameters s
$\tilde{\mathcal{Z}}(\underline{x}, t)$	subshapes $\tilde{\mathcal{Z}}(\underline{x}, t) \subseteq \tilde{\mathcal{Z}}(\underline{x})$ in function of an index parameter t
$\mathcal{Y}$	list of measurements $\underline{y}$
$p(\underline{x})$	prior distribution of the object state
$p(\underline{y} \mid \underline{x})$	likelihood for the single measurement $\underline{y}$
$p(\underline{z} \mid \underline{x})$	distribution of points $\underline{z} \in \mathbb{R}^d$ to be measurement sources, given the state $\underline{x}$,
$g(\underline{x}, \underline{z})$	shape function that relates points $\underline{z}$ to the shape $\tilde{\mathcal{Z}}(\underline{x})$
$T(\underline{z})$	aggregation function that maps points $\underline{z}$ to the non-redundant part
$\mathcal{N}(\underline{z}; \underline{\mu}, \mathbf{C})$	Gaussian distribution with mean $\underline{\mu}$, covariance matrix $\mathbf{C}$, evaluated at $\underline{z}$

4 BAYESIAN EXTENDED OBJECT TRACKING

In this section, we introduce the concepts of Bayesian extended object tracking. In a Bayesian estimator, the state $\underline{x}$ to be estimated is modeled as a random vector whose distribution $p(\underline{x})$ represents the uncertain knowledge about the object's pose and shape. In this work, we assume this uncertainty to be Gaussian.

The tracking algorithm consists of two alternating steps. First, the *prediction step* lets the distribution $p(\underline{x})$ evolve over time according to a system model. Second, the *measurement update step* incorporates new measurement points $\mathcal{Y}$ according to Bayes' rule

$$p(\underline{x} \mid \mathcal{Y}) \propto p(\mathcal{Y} \mid \underline{x}) \cdot p(\underline{x}), \tag{5}$$

where $p(\mathcal{Y} \mid \underline{x})$ is the likelihood that describes how likely a measured set of points $\mathcal{Y}$ is, given a state $\underline{x}$.

As we assume all points $\underline{y}_i \in \mathcal{Y}$ to be conditionally independent from the state $\underline{x}$, the likelihood can be rewritten to

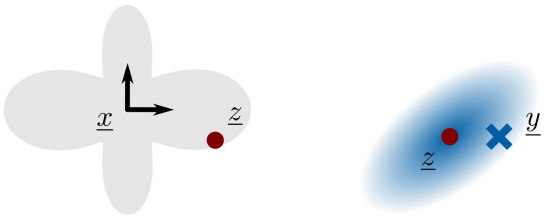
$$p(\mathcal{Y} \mid \underline{x}) = \prod_{i=1}^{n} p(\underline{y}_i \mid \underline{x}), \tag{6}$$

which lets us consider individual likelihoods for single measurements $\underline{y}_i \in \mathcal{Y}$. If more points are available, they can be processed sequentially according to (6).

Furthermore, as the measurement noise is assumed to be independent from the state, the likelihood $p(\underline{y} \mid \underline{x})$ can be divided into

$$p(\underline{y} \mid \underline{x}) = \int_{\mathbb{R}^d} p(\underline{y} \mid \underline{z}) \cdot p(\underline{z} \mid \underline{x}) d\underline{z}, \tag{7}$$

where the *sensor model* $p(\underline{y} \mid \underline{z})$ specifies the distribution of measurements $\underline{y}$ for a given point $\underline{z}$, and the *source model* $p(\underline{z} \mid \underline{x})$ specifies the distribution of points $\underline{z} \in \mathbb{R}^d$ to be the measurement sources for a given state $\underline{x}$. Both are schematically shown in Fig. 3.



(a) source model $p(\underline{z}|\underline{x})$ (b) sensor model $p(\underline{y}|\underline{z})$

Fig. 3. Sketch of source model and sensor model.

According to (3), the sensor model is immediately given by the convolution

$$\begin{aligned} p(\underline{y} \mid \underline{z}) &= \int_{\mathbb{R}^d} p(\underline{y} \mid \underline{z}, \underline{w}) \cdot p(\underline{w}) d\underline{w} \\ &= \int_{\mathbb{R}^d} \delta(\underline{y} - (\underline{z} + \underline{w})) \cdot \mathcal{N}(\underline{w}; \underline{0}, \mathbf{C}_w) d\underline{w} \\ &= \mathcal{N}(\underline{y} - \underline{z}; \underline{0}, \mathbf{C}_w) \end{aligned} \tag{8}$$

For the source model, in cases where the true source is known, e.g., when looking at point objects without extent, it holds that $p(\underline{z} \mid \underline{x}) = \delta(\underline{z} - \tilde{\underline{z}}(\underline{x}))$, where $\tilde{\underline{z}}(\underline{x})$ refers to the source location of the point object. However, as we consider extended objects in this work, each state $\underline{x}$ generates a set of possible sources $\tilde{\mathcal{Z}}(\underline{x})$ rather than a single source. In order to allow more than one measurement source, we introduce $s \in \mathcal{S}$ as an index parameter that iterates through all possible sources $\tilde{\underline{z}}(\underline{x}, s) \in \tilde{\mathcal{Z}}(\underline{x})$ for a given state $\underline{x}$. The parameter s often refers to real-valued scalars from $\mathcal{S} \subseteq \mathbb{R}$ which, e.g., allows for iterating through sources along a line segment. However, in the case of sources on more complex shapes, such as a rectangle, the index parameter can be vector-valued. For example, in Sec. 4.3, we use a parametrization based on two index parameters $s \in \mathcal{S}$, and $t \in \mathcal{T}$, where pairs of (s, t) iterate through the sources $\tilde{\underline{z}}(\underline{x}, s, t) \in \tilde{\mathcal{Z}}(\underline{x})$.

An appropriate source model for extended objects has the form $p(\underline{z} \mid \underline{x}, s) = \delta(\underline{z} - \tilde{\underline{z}}(\underline{x}, s))$, which additionally depends on the index parameter s, and each refers to a single source. The task then is to derive a source model $p(\underline{z} \mid \underline{x})$ based on the individual source models $p(\underline{z} \mid \underline{x}, s)$. As the true s is usually not known in advance, we are faced with the so called *association problem*. In the following, we summarize three popular source models that deal with this problem. Accompanying the textual description, Fig. 4 presents some illustrative examples.

4.1 Spatial Distribution Model

A *spatial distribution model* (SDM) as presented in [9], [24] is a source model $p(\underline{z} \mid \underline{x})$, derived by marginalizing $s \in \mathcal{S}$ out of the individual models $p(\underline{z} \mid \underline{x}, s)$. The intuition of an SDM is to *explicitly* assign a probabil-

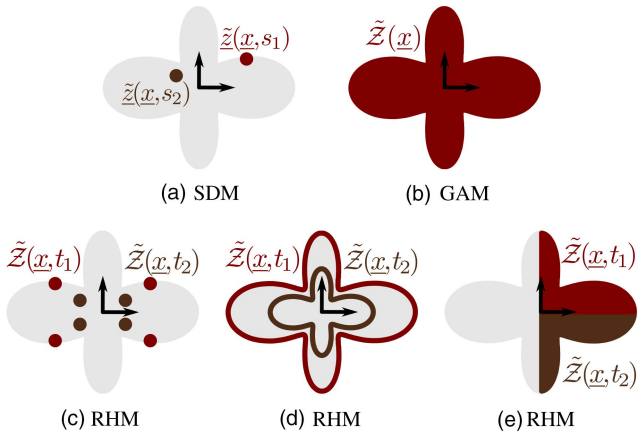


Fig. 4. Sketch of the considered types of source models.

ity $p(s)$ to each individual source $\tilde{\underline{z}}(\underline{x},s) \in \tilde{\mathcal{Z}}(\underline{x})$ of being measured. The marginalization then can be written as

$$p(\underline{z} \mid \underline{x}) = \int_{\mathcal{S}} p(\underline{z} \mid \underline{x},s) \cdot p(s) ds \qquad (9)$$
$$= \int_{\mathcal{S}} \delta(\underline{z} - \tilde{\underline{z}}(x,s)) \cdot p(s) ds.$$

In other words, each potential source $\tilde{\underline{z}}(\underline{x},s)$ is considered as hypothesis and weighted according to its probability.

A popular reformulation [9], [24] of (9) is to substitute the expression $\underline{z} - \tilde{\underline{z}}(\underline{x},s)$ by a distance-related function $\|\underline{z} - \tilde{\underline{z}}(\underline{x},s)\|$ that returns a scalar 0 if $\underline{z}$ equals $\tilde{\underline{z}}(\underline{x},s)$. By means of the distance function, the SDM (9) can be rewritten as

$$p(\underline{z} \mid \underline{x}) = \int_{\mathcal{S}} \delta(\|\underline{z} - \tilde{\underline{z}}(\underline{x},s)\|) \cdot p(s) ds. \qquad (10)$$

A drawback of the SDM is that in most real-life scenarios, $p(s)$ is not known in advance, and deriving it is a non-trivial task, as it depends on factors such as sensor to object geometry, the specific segmentation algorithm, and occlusions, among others. This raises the need for approaches that depend less on $p(s)$.

4.2 Greedy Association Model

Another source modeling technique, popular in curve fitting [7], is to approximate the unknown source distribution by $p(s) = \delta(s - \hat{s})$ where the index $\hat{s}$ specifies the true source. The sifting property then lets us eliminate the integral in (10)

$$p(\underline{z} \mid \underline{x}) = \int_{\mathcal{S}} \delta(\|\underline{z} - \tilde{\underline{z}}(x,s)\|) \cdot \delta(s - \hat{s}) ds \qquad (11)$$
$$= \delta(\|\underline{z} - \tilde{\underline{z}}(x,\hat{s})\|).$$

Although this would be the ideal source model, $\hat{s}$ and in consequence $\tilde{\underline{z}}(x,\hat{s})$ is usually unknown (association problem). As an approximation, $\hat{s}$ can be greedily chosen, such that $\tilde{\underline{z}}(x,\hat{s})$ refers to the source on the shape that is "closest" to $\underline{z}$. In consequence $\|\underline{z} - \tilde{\underline{z}}(x,\hat{s})\|$ returns 0 for all $\underline{z} \in \tilde{\mathcal{Z}}(\underline{x})$. This motivates the use of an *implicit shape function* in the form of

$$g(\underline{x},\underline{z}) = \min_{s \in \mathcal{S}} (\|\underline{z} - \tilde{\underline{z}}(\underline{x},s)\|), \qquad (12)$$

which returns the minimum over all distances between the point $\underline{z}$ and all sources on the shape $\tilde{\mathcal{Z}}(\underline{x})$. Note that finding the minimum implicitly includes the greedy selection of the expected source. Substituting the expression $\|\underline{z} - \tilde{\underline{z}}(x,\hat{s})\|$ in (11) with the shape function $g(\underline{x},\underline{z})$ from (12), we obtain a *Greedy Association Model* (GAM)

$$p(\underline{z} \mid \underline{x}) = \delta(g(\underline{x},\underline{z})). \qquad (13)$$

As an advantage over SDMs, GAMs do not depend on any known distribution $p(s)$ at all, but at the cost of a greedy association. This approximation was shown in [7] to introduce a parameter bias in the estimate in the presence of high noise. A second issue is that estimators using GAMs usually require regularization as they do not penalize overestimates [24].

4.3 Random Hypersurface Model

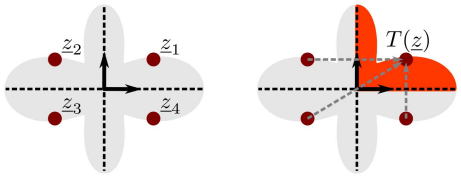
Combining the ideas from the SDM and the GAM, we arrive at *Random Hypersurface Models* (RHMs) [2] that inherit the advantages from both source models. In detail, they depend less on the distribution of individual sources and do not require regularization. To derive the RHM, let us start with an SDM that is parameterized by two indices $s \in \mathcal{S}$ and $t \in \mathcal{T}$, i.e., pairs (s,t) iterate through all possible sources $\tilde{\underline{z}}(x,s,t) \in \tilde{\mathcal{Z}}(\underline{x})$ for a given state $\underline{x}$. Let each source have a given probability $p(s,t)$. Then, marginalizing (s,t) out of $p(\underline{z} \mid \underline{x},s,t)$ an SDM can be written as

$$p(\underline{z} \mid \underline{x}) = \int_{\mathcal{S}} \int_{\mathcal{T}} p(\underline{z} \mid \underline{x},s,t) \cdot p(s,t) ds\, dt \qquad (14)$$
$$= \int_{\mathcal{S}} \int_{\mathcal{T}} \delta(\|\underline{z} - \tilde{\underline{z}}(x,s,t)\|) \cdot p(s \mid t) \cdot p(t) ds\, dt.$$

The goal of using two indices is to choose their parametrization such that for a given t_i, iterating over all sources $\tilde{\underline{z}}(x,s,t_i)$ with $s \in \mathcal{S}$ refers to a reasonable subshape we denote as $\tilde{\mathcal{Z}}(\underline{x},t_i)$. These subshapes, however, are not unique and can be defined in multiple ways. Three illustrative examples are shown in 4(c–e) and range from subshapes composed of a few discrete points (Fig. 4(c)), over a scaled boundary (Fig. 4(d)) to entire parts of the shape (Fig. 4(e)).

An RHM assumes that $p(t)$ is known in advance (SDM), i.e., the probability that a measurement originates from a specific subshape, while the probability of the individual source on the subshape is unknown and has to be determined greedily according to $p(s \mid t) = \delta(s - \hat{s}_t)$ (GAM). Plugging $p(s \mid t)$ into (10) then lets us eliminate the integral over s

$$p(\underline{z} \mid \underline{x}) = \int_{\mathcal{T}} \delta(\|\underline{z} - \tilde{\underline{z}}(x,\hat{s}_t,t)\|) \cdot p(t) dt. \qquad (15)$$



(a) symmetric sources (b) aggregation

Fig. 5. Symmetric points $\underline{z}_1$, $\underline{z}_2$, $\underline{z}_3$, and $\underline{z}_4$ all lie inside or outside of a symmetric shape. The aggregation function $T(\cdot)$ transforms all points $\underline{z}$ to their symmetric equivalents in the the non-redundant part (orange).

Analogously to the GAM, it is common practice [2] to substitute the expression $\|\underline{z} - \tilde{\underline{z}}(x, \hat{s}_t, t)\|$ with an implicit shape function

$$g_t(\underline{x}, \underline{z}) = \min_{s \in \mathcal{S}}(\|\underline{z} - \tilde{\underline{z}}(\underline{x}, s, t)\|) \tag{16}$$

that additionally depends on t and returns the distance to the subshape $\tilde{\mathcal{Z}}(\underline{x}, t)$. Finally, by means of $g_t(\cdot)$, we can rewrite the source model (15) to obtain the RHM

$$p(\underline{z} \mid \underline{x}) = \int_{\mathcal{T}} \delta(g_t(\underline{x}, \underline{z})) \cdot p(t) dt. \tag{17}$$

REMARK 1 **(Set-theoretic Ignorance)**
In this work, we use the term *set-theoretic ignorance* to denote that a source model greedily selects sources from a (sub-)shape instead of assuming an individual probability on each source, as these are assumed to be unknown. As such, SDMs do not assume any set-theoretic ignorance at all, while GAMs assume set-theoretic ignorance over the entire shape. In RHMs, the *degree* of set-theoretic ignorance depends on the extent of the subshapes.

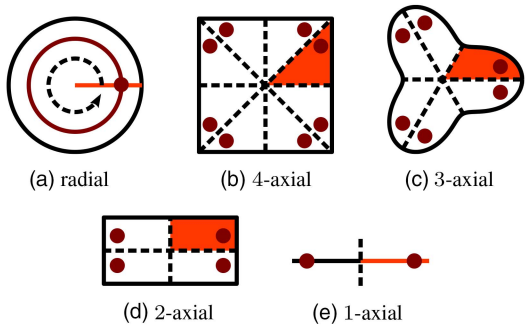
5 KEY IDEA

We will now discuss the key idea of applying symmetric simplification to source models $p(\underline{z} \mid \underline{x})$. While there are many ways to describe symmetries, in this paper we focus on symmetry as *repetition*. As such, a symmetric shape can be seen as being generated by the repeated transformation of a small part of the shape, which we denote as the *non-redundant part*. Then, for these shapes, we can find an *aggregation function* $T(\underline{z})$ that maps each point to its symmetric equivalent in the non-redundant part. More formally, let $T(\cdot)$ be an *idempotent* function, i.e., it always holds that $T(T(\underline{z})) = T(\underline{z})$. We say that a shape $\tilde{\mathcal{Z}}(\underline{x})$ is symmetric by $T(\cdot)$ if for all points $\underline{z} \in \mathbb{R}^d$ it holds that

$$\underline{z} \in \tilde{\mathcal{Z}}(\underline{x}) \Leftrightarrow T(\underline{z}) \in \tilde{\mathcal{Z}}(\underline{x}). \tag{18}$$

In this case, we say that $T(\cdot)$ is an aggregation function of the shape $\tilde{\mathcal{Z}}(\underline{x})$, and its non-redundant part is given by $T(\tilde{\mathcal{Z}}(\underline{x}))$. This relationship is illustrated in Fig. 5 for a 2-axial symmetric shape.

The key idea is to exploit that under certain conditions, the presented source models for symmetric shapes



(a) radial (b) 4-axial (c) 3-axial

(d) 2-axial (e) 1-axial

Fig. 6. Types of symmetry: rotational symmetry (a) and instances of axial symmetry (b–e). The non-redundant part of each shape is marked in orange. As an example, one source in each non-redundant part and all symmetric equivalents is marked in red.

are symmetric too, i.e., it holds that

$$p(\underline{z} \mid \underline{x}) = p(T(\underline{z}) \mid \underline{x}). \tag{19}$$

Specifically, we will show that this identity holds when a source model only has symmetric subshapes. Then, by aggregating symmetric points according to $T(\cdot)$, the original domain $\mathbb{R}^d$ of the source model $p(\underline{z} \mid \underline{x})$ can be reduced to the non-redundant part $T(\mathbb{R}^d)$ of the domain. This allows us to specify a source model exclusively in the non-redundant part and use $p(T(\underline{z}) \mid \underline{x})$ in the estimator. The desired simplification for the models in the top row of Fig. 7 is illustrated in the bottom row.

As can be seen, all shape information is contained in the non-redundant part, e.g., the first quadrant for the 2-axial symmetry.

6 CONSIDERED SYMMETRIES

Some shapes with corresponding symmetries are visualized in Fig. 6. The non-redundant part of each shape is marked in orange. In this work, we focus on rotoreflections with $2n$-fold rotation angle, which means a shape is generated by reflecting its non-redundant part with respect to $n \in \mathbb{N}$ rotated axes that intersect in the origin. This type of symmetry includes special cases such as axial symmetries in Fig. 6(b–e), as well as radial symmetry for $n \to \infty$ (see Fig. 6(a)). Next, we derive explicit formulas for appropriate aggregation functions $T(\cdot)$ for two and three dimensions.

6.1 2D Aggregation Function

When dealing with rotoreflections in 2D, polar coordinates offer a convenient representation. For conversion of a point $\underline{z}$, given in Cartesian coordinates $\underline{z} = [z_1, z_2]^{\mathrm{T}}$, the corresponding representation in polar coordinates is defined as

$$\theta(\underline{z}) = \operatorname{atan2}(z_2, z_1), \tag{20}$$

$$r(\underline{z}) = \sqrt{z_1^2 + z_2^2}, \tag{21}$$

where $\theta(\underline{z})$ is the angle to the z_1-axis and $r(\underline{z})$ is the Euclidean norm of $\underline{z}$. For a rotoreflection, its $2n$-fold

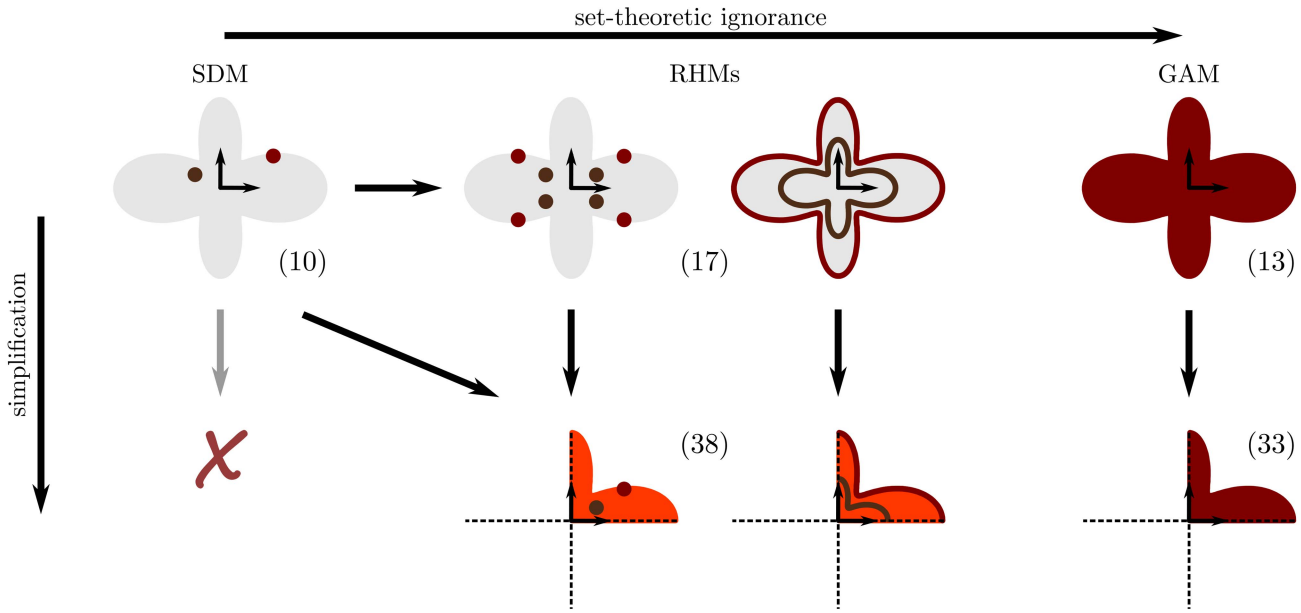


Fig. 7. Overview of the proposed approach. The top row illustrates the considered source models. From the left to the right, the set-theoretic ignorance in the model increases. The bottom row shows the simplified source models for the top row. Sources belonging to the same (sub-)sets of the shape, are colored in brown and red, respectively.

rotation angle is given by

$$\Theta = \frac{\pi}{n}, \tag{22}$$

which means that each of the n axes of symmetry is rotated about Θ. In terms of polar coordinates, we then define the non-redundant part to lie in the period from 0 to Θ. Thus, the corresponding aggregation function $T(\cdot)$ should map all points onto their equivalents in this part. Note that this mapping does not affect the radius $r(\underline{z})$, but rather it requires a modulo operation on the angle $\theta(\underline{z})$ according to

$$\theta(\underline{z},n) = \begin{cases} \mathrm{mod}(\theta(\underline{z}),\Theta) & \text{if } \left\lfloor \frac{\theta(\underline{z})}{\Theta} \right\rfloor \text{ is even} \\ \Theta - \mathrm{mod}(\theta(\underline{z}),\Theta) & \text{if } \left\lfloor \frac{\theta(\underline{z})}{\Theta} \right\rfloor \text{ is odd.} \end{cases} \tag{23}$$

Finally, an appropriate aggregation function can be defined as

$$T(\underline{z}) = \begin{bmatrix} r(\underline{z}) \cdot \cos(\theta(\underline{z},n)) \\ r(\underline{z}) \cdot \sin(\theta(\underline{z},n)) \end{bmatrix}. \tag{24}$$

Special Cases: The general aggregation function (24) for rotoreflections includes several special cases. For example, given a reflectional symmetry with respect to the z_2-axis (see Fig. 6(e)), the non-redundant part is a half-plane, and the aggregation function

$$T(\underline{z}) = [|z_1|, z_2]^{\mathrm{T}} \tag{25}$$

aggregates points $\underline{z}$, according to its absolute values $|.|$ in z_1. Analogously, the aggregation function for a 2-axial symmetry (see Fig. 5 and Fig. 6(d)) maps each point $\underline{z}$ to the first quadrant, according to

$$T(\underline{z}) = [|z_1|, |z_2|]^{\mathrm{T}}. \tag{26}$$

A rotoreflection with $n \to \infty$ axes causes the non-redundant part to collapse into a ray, e.g., the positive z_1-axis, where the aggregation function can be written as

$$T(\underline{z}) = [\|\underline{z}\|, 0]^{\mathrm{T}}, \tag{27}$$

and maps each point $\underline{z}$ to this axis, according to its distance $\|.\|$ to the origin.

6.2 3D Aggregation Function

For rotoreflections in 3D, cylindrical coordinates are chosen. The conversion of a point $\underline{z}$ given in Cartesian coordinates $\underline{z} = [z_1, z_2, z_3]^{\mathrm{T}}$ is very similar to the 2D case. First, (20) can be used to derive polar coordinates from z_1 and z_2 and second, the height component is directly given by z_3. Introducing an additional symmetry in 3D can be achieved by taking the absolute value of z_3. Then, an aggregation function with reflectional symmetry in the height can be defined as

$$T(\underline{z}) = \begin{bmatrix} r(\underline{z}) \cdot \cos(\theta(\underline{z},n)) \\ r(\underline{z}) \cdot \sin(\theta(\underline{z},n)) \\ |z_3| \end{bmatrix}. \tag{28}$$

7 SIMPLIFICATION BASED ON SYMMETRY

In this section, we derive a simplification scheme for the presented source models, given that a target object has a known rotoreflectional symmetry. To achieve this goal, we investigate the symmetric properties of the different models $p(\underline{z} \mid \underline{x})$ and show how to take advantage of it. In doing so, we begin with the GAM, which allows for the simplest simplification, and subsequently look at the RHM and the SDM.

7.1 Simplified Greedy Association Model

Let the source model of an object with state $\underline{x}$ and shape function $g(\cdot)$ be specified by the GAM $p(\underline{z} \mid \underline{x}) = \delta(g(\underline{x},\underline{z}))$. Further, let the object shape have a known rotoreflectional symmetry described by the aggregation function $T(\cdot)$. Examining the symmetric properties of this source model refers to examining those of the shape function. As $g(\cdot)$ returns the minimum distance from points $\underline{z} \in \mathbb{R}^d$ to the shape $\tilde{\mathcal{Z}}(\underline{x})$, it produces equal values for all symmetric equivalents of $\underline{z}$. Specifically, it holds that

$$\begin{aligned} g(\underline{x},\underline{z}) &= \min_{s\in\mathcal{S}}(\|\underline{z} - \tilde{\underline{z}}(\underline{x},s)\|) \\ &= \min_{s\in\mathcal{S}}(\|T(\underline{z}) - \tilde{\underline{z}}(\underline{x},s)\|) \\ &= g(\underline{x},T(\underline{z})), \end{aligned} \tag{29}$$

which immediately implies that the GAM (13)

$$p(\underline{z} \mid \underline{x}) = \delta(g(\underline{x},\underline{z})) = \delta(g(\underline{x},T(\underline{z}))) = p(T(\underline{z}) \mid \underline{x}) \tag{30}$$

is symmetric too. From this follows that in practice the shape function $g(\cdot)$ needs to be evaluated only in the non-redundant part $T(\mathbb{R}^d)$ of the domain $\mathbb{R}^d$.

As simplification, we propose to substitute the original shape function $g(\cdot)$ with a simplified shape function $g^*(\cdot)$, which is exclusively defined in the non-redundant part of the domain. For this purpose, let the indices $s^* \in \mathcal{S}^*$ with $\mathcal{S}^* \subseteq \mathcal{S}$ refer to sources in the non-redundant part $\tilde{\underline{z}}(\underline{x},s^*) \in T(\tilde{\mathcal{Z}}(\underline{x}))$ according to

$$\mathcal{S}^* := \{s \in \mathcal{S} \mid T(\tilde{\underline{z}}(\underline{x},s)) = \tilde{\underline{z}}(\underline{x},s)\}. \tag{31}$$

Then a simplified shape function is given by

$$g^*(\underline{x},\underline{z}) := \min_{s^*\in\mathcal{S}^*}(\|T(\underline{z}) - \tilde{\underline{z}}(\underline{x},s^*)\|) \tag{32}$$

that internally performs the aggregation $T(\cdot)$ and can be employed to define a simplified GAM

$$p(\underline{z} \mid \underline{x}) = \delta(g^*(\underline{x},\underline{z})). \tag{33}$$

Note that this simplification does not introduce any error at all in the source model, as it only exploits the fact that distances from symmetric points to a symmetric shape are equal. For this reason, we can encapsulate all symmetric considerations in the shape function.

7.2 Simplified Random Hypersurface Model

Let the source model $p(\underline{z} \mid \underline{x})$ of an object $\underline{x}$ with shape function $g_t(\cdot)$ and distribution $p(t)$ be specified by the RHM (17). This means that the shape is composed of subshapes $\tilde{\mathcal{Z}}(\underline{x},t)$, where each of them is specified by the shape function $g_t(\cdot)$. Furthermore, let the object shape have a known rotoreflectional symmetry described by the aggregation function $T(\cdot)$. In order to apply symmetric simplification, we require each of the subshapes $\tilde{\mathcal{Z}}(\underline{x},t)$ to be symmetric with respect to a given aggregation function $T(\cdot)$, i.e., for all points $\underline{z}$ it shall hold that

$$\underline{z} \in \tilde{\mathcal{Z}}(\underline{x},t) \Leftrightarrow T(\underline{z}) \in \tilde{\mathcal{Z}}(\underline{x},t). \tag{34}$$

When looking at the example RHMs from Fig. 4, subshapes in (c) and (d) are symmetric, while the subshapes in (e) are not. Then, all symmetric considerations (29) from the GAM apply to the symmetric subshapes $\tilde{\mathcal{Z}}(\underline{x},t)$ and yield the identity

$$g_t(\underline{x},\underline{z}) = g_t(\underline{x},T(\underline{z})). \tag{35}$$

From this can be concluded that, given all subshapes are symmetric, the RHM (17)

$$\begin{aligned} p(\underline{z} \mid \underline{x}) &= \int_{\mathcal{T}} \delta(g_t(\underline{x},\underline{z})) \cdot p(t)dt \\ &= \int_{\mathcal{T}} \delta(g_t(\underline{x},T(\underline{z}))) \cdot p(t)dt \\ &= p(T(\underline{z}) \mid \underline{x}) \end{aligned} \tag{36}$$

is also symmetric. In turn, the shape function $g_t(\cdot)$ needs to be evaluated only in the non-redundant part of the domain $T(\mathbb{R}^d)$, which gives rise to the same simplification technique as applied to the GAM. We propose to substitute the original shape function $g_t(\cdot)$ with a simplified version

$$g_t^*(\underline{x},\underline{z}) := \min_{s^*\in\mathcal{S}^*}(\|T(\underline{z}) - \tilde{\underline{z}}(\underline{x},s^*,t)\|) \tag{37}$$

that only needs to be evaluated in the non-redundant domain $T(\mathbb{R}^d)$. Then, the simplified RHM is given by

$$p(\underline{z} \mid \underline{x}) = \int_{\mathcal{T}} \delta(g_t^*(\underline{x},\underline{z})) \cdot p(t)dt. \tag{38}$$

Again, this simplification does not introduce any error at all, as it only exploits the fact that distances from symmetric points to a symmetric (sub-)shape are equal.

7.3 Simplified Spatial Distribution Model

Let the source model $p(\underline{z} \mid \underline{x})$ of an object with $\underline{x}$ with distribution $p(s)$ over the sources $\tilde{\underline{z}}(\underline{x},s)$ be specified by the SDM (10). Furthermore, let the object shape have a known rotoreflectional symmetry described by the aggregation function $T(\cdot)$. It can be seen that for (10) in general $p(\underline{z} \mid \underline{x})$ and $p(T(\underline{z}) \mid \underline{x})$ do not produce identical results as

$$\|\underline{z} - \tilde{\underline{z}}(\underline{x},s)\| \neq \|T(\underline{z}) - \tilde{\underline{z}}(\underline{x},s)\|. \tag{39}$$

In consequence, an SDM as defined in (10) cannot be simplified using the proposed technique, which is indicated in Fig. 7. In order to allow for simplification, we propose to approximate the SDM by an RHM that uses symmetric subshapes and then simplify this RHM.

Creating Symmetric Subshapes: We can use $T(\cdot)$ to create symmetric subshapes by taking a source $\tilde{\underline{z}}(\underline{x},s^*)$ with $s^* \in \mathcal{S}^*$ that lies in the non-redundant part and then collect all sources $T(\tilde{\underline{z}}(\underline{x},s))$ being mapped to this source. We denote the corresponding subshape as $\tilde{\mathcal{Z}}(\underline{x},s^*)$ and define the *equivalence class* of indices, which contains the indices of all sources in the subshape according to

$$[s^*] := \{s \in \mathcal{S} \mid T(\tilde{\underline{z}}(\underline{x},s)) = \tilde{\underline{z}}(\underline{x},s^*)\}. \tag{40}$$

The illustration of the left RHM in the first row of Fig. 7 shows an example of these subshapes. Note that per definition, only a single point of each subshape lies in the non-redundant part (see Fig. 6). Based on these subshapes, the original SDM (10) can now be approximated by an RHM.

Approximating SDM by RHM: In order to define the desired RHM, we need to specify a shape function $g_{s^*}(\underline{x},\underline{z})$ that returns the minimum distance from a point $\underline{z}$ to a subshape $\tilde{\mathcal{Z}}(\underline{x},s^*)$, as well as the distribution $p(s^*)$. An appropriate shape function is given by

$$g_{s^*}(\underline{x},\underline{z}) = \min_{s\in[s^*]}(\|\underline{z}-\tilde{\underline{z}}(\underline{x},s)\|). \tag{41}$$

The distribution $p(s^*)$ specifies the probability that a source within the subshape $\tilde{\mathcal{Z}}(\underline{x},s^*)$ is measured and can be calculated by integrating the individual probabilities according to

$$p(s^*) = \int_{[s^*]} p(s)ds. \tag{42}$$

Using the shape function g_{s^*} and the distribution $p(s^*)$ the RHM from (17) can then be rewritten to

$$p(\underline{z}\mid\underline{x}) = \int_{\mathcal{S}^*} \delta(g_{s^*}(\underline{x},\underline{z}))\cdot p(s^*)ds^*. \tag{43}$$

Simplifying the RHM: At this point, we arrived at an RHM that uses symmetric subshapes, which in turn lets us substitute the original shape function $g_{s^*}(\cdot)$ with a simplified version

$$g^*_{s^*}(\underline{x},\underline{z}) = \|T(\underline{z})-\tilde{\underline{z}}(\underline{x},s^*)\| \tag{44}$$

that only needs to be evaluated in the non-redundant part $T(\mathbb{R}^d)$ of the domain $\mathbb{R}^d$. We will denote (43) as simplified SDM.

When comparing the simplified SDM (43) to the original SDM (10), there are two differences. First, the aggregation of symmetric sources has introduced a set-theoretic ignorance, as the model only allows distinct probabilities for subshapes $\tilde{\mathcal{Z}}(\underline{x},s^*)$, where sources in the subshapes are associated greedily. Second, the integration variable s^* of the RHM has a reduced domain compared to the original integration variable s of the SDM. When using a sample-based filter, this reduction yields a higher effective sample resolution, which improves the estimation result.

It is important to note that, in practice, the simplified source models in the bottom row of Fig. 7 can be used directly without having to model the traditional versions from the top row.

8 DERIVING THE ESTIMATOR

We will now derive a Bayesian estimator that takes advantage of the symmetric considerations of the previous section. For this purpose, we derive likelihoods based on the simplified source models and then show how the measurement update can be implemented using a *Linear Regression Kalman Filter*.

8.1 Likelihoods and Measurement Equations

In a preliminary step, let us plug the sensor model (8) in the likelihood from (7) and rearrange it to

$$\begin{aligned} p(\underline{y}\mid\underline{x}) &= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} p(\underline{y}\mid\underline{z},\underline{w})\cdot p(\underline{w})d\underline{w}\,p(\underline{z}\mid\underline{x})d\underline{z} \\ &= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \delta((\underline{y}-\underline{w})-\underline{z})\cdot p(\underline{w})d\underline{w}\cdot p(\underline{z}\mid\underline{x})d\underline{z}. \end{aligned} \tag{45}$$

As a reminder, $\underline{y}$ is a point measurement and $\underline{w}$ is a noise variable which describes the sensor uncertainty. Then, as in the previous section, we begin our considerations with the simplest model, i.e., the GAM. Plugging the simplified GAM $p(\underline{z}\mid\underline{x})$ from (33) into the rearranged likelihood (45) yields

$$\begin{aligned} p(\underline{y}\mid\underline{x}) &= \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \delta((\underline{y}-\underline{w})-\underline{z})\cdot p(\underline{w})d\underline{w}\cdot\delta(g^*(\underline{x},\underline{z}))d\underline{z} \\ &= \int_{\mathbb{R}^d} \delta(g^*(\underline{x},\underline{y}-\underline{w}))\cdot p(\underline{w})d\underline{w}, \end{aligned} \tag{46}$$

where the integral over $\underline{z}$ was eliminated by applying the sifting property. Again, it is important to note that all symmetric considerations are encapsulated in the simplified shape function $g^*(\cdot)$ that internally applies the aggregation function to $\underline{y}-\underline{w}$ in order to calculate its distance to the shape.

Usually, the shape function $g^*(\cdot)$ will be defined in object coordinates while measurement minus noise $(\underline{y}-\underline{w})^W$ is given in world coordinates. In order to deal with this issue, the conversion technique from (4) can be applied $(\underline{y}-\underline{w})^O = H(\underline{x},(\underline{y}-\underline{w})^W)$. Then, the likelihood can be written as

$$p(\underline{y}\mid\underline{x}) = \int_{\mathbb{R}^d} \delta(h(\underline{x},\underline{y},\underline{w}))\cdot p(\underline{w})d\underline{w}, \tag{47}$$

where h defines a nonlinear *implicit measurement equation* of the form

$$0 = h(\underline{x},\underline{y},\underline{w}) := g^*(\underline{x},H(\underline{x},\underline{y}-\underline{w})). \tag{48}$$

Within this implicit function, the original measurement $\underline{y}$ acts as an additional function parameter, $\underline{w}$ is a non-additive noise variable, and 0 is an artificial pseudo-measurement.

Likewise, the likelihood based on the simplified RHM (33) can be derived as

$$p(\underline{y}\mid\underline{x}) = \int_{\mathbb{R}^d}\int_{\mathcal{T}} \delta(h(\underline{x},\underline{y},\underline{w},t))\cdot p(t)\cdot p(\underline{w})d\underline{w}\,dt. \tag{49}$$

In this case, the implicit measurement equation

$$0 = h(\underline{x},\underline{y},\underline{w},t) := g^*_t(\underline{x},H(\underline{x},\underline{y}-\underline{w}))) \tag{50}$$

additionally depends on a second non-additive noise variable t. Substituting t with s^* in the formulas (49) and (50) yields the simplified SDM.

Unfortunately, it is usually not possible to evaluate the required integrals in the likelihoods and, in turn, the *measurement update step* analytically. Still, there are

techniques available that can be applied in order to derive an approximate update, such as, e.g., *Monte Carlo* integration [9], [18]. Of particular interest are Linear Regression Kalman Filters (LRKF) such as the *Unscented Kalman Filter* (UKF) [10], or the *Smart Sampling Kalman Filter* (S^2KF) [23], which were successfully applied to extended object tracking [1–4], [6].

8.2 Linear Regression Kalman Filter

We now provide instructions and formulas to implement an approximate sample-based Bayes' update [10], [23] for a (simplified) RHM. Alg. 1 shows the resulting Kalman formulas, where the measurement mean μ_h, measurement covariance $\mathbf{C}_h$ and state-measurement cross-covariance $\mathbf{C}_{xh}$ are obtained through deterministic sampling (statistical linearization). It is important to note that besides state $\underline{x}$ and measurement noise $\underline{w}$, the index t is modeled as a random variable and, thus, subject to sampling. This algorithm can easily be adjusted to derive updates for (simplified) GAMs and SDMs by either dropping the noise parameter t or substituting it by s^*.

ALGORITHM 1 *Sample-based Bayes' update for a (simplified) RHM.*

input: prior distribution of state $p(\underline{x})$, and noise $p(\underline{w})$, $p(t)$, measured point $\underline{y}$
output: posterior distribution of state $p(\underline{x}^e)$
begin
draw L samples $\{[\underline{x}_l^{\mathrm{T}}, \underline{w}_l^{\mathrm{T}}, t_l]^{\mathrm{T}}\}_{l=1}^{L}$ from the joint distribution $p([\underline{x}^{\mathrm{T}}, \underline{w}^{\mathrm{T}}, t]^{\mathrm{T}})$;
calculate sample mean μ_h, sample covariance $\mathbf{C}_h$ and sample cross-covariance $\mathbf{C}_{xh}$ according to

$$\mu_h = \frac{1}{L}\sum_{l=1}^{L} h(\underline{x}_l, \underline{y}, \underline{w}_l, t_l),$$

$$\mathbf{C}_h = \frac{1}{L}\sum_{l=1}^{L} h(\underline{x}_l, \underline{y}, \underline{w}_l, t_l)^2 - \mu_h^2,$$

$$\mathbf{C}_{xh} = \frac{1}{L}\sum_{l=1}^{L} (h(\underline{x}_l, \underline{y}, \underline{w}_l, t_l) - \mu_h) \cdot (\underline{x}_l - \mu_x)^{\mathrm{T}};$$

calculate posterior

$$\underline{\mu}_x^e = \underline{\mu}_x + \mathbf{K}\mu_h$$
$$\mathbf{C}_x^e = \mathbf{C}_x - \mathbf{K}\mathbf{C}_h\mathbf{K}^{\mathrm{T}}$$

with Kalman gain $\mathbf{K} = \mathbf{C}_{xh}\mathbf{C}_h^{-1}$;
return $p(\underline{x}^e) = \mathcal{N}(\underline{\mu}_x^e, \mathbf{C}_x^e)$
end

8.3 Discussion

Based on these derivations, we can conclude several remarks.

REMARK 2 **(Usability)**
As all symmetric considerations are encapsulated in the shape function, the proposed simplification approach does not require any adaptions to the filter being used. In consequence, Alg. 1 can be applied to derive the measurement update, using the common source models as well as their simplified versions, simply by switching the measurement function.

REMARK 3 **(Generalization)**
The proposed approach is a general solution to exploit symmetries in the measurement function and is not restricted to shape estimation. Specifically, whenever an implicit measurement equation of the structure

$$\underline{0} = h(\underline{x}, \underline{\alpha}) \tag{51}$$

is given, and for all $\underline{\alpha}$ the identity

$$h(\underline{x}, \underline{\alpha}) = h(\underline{x}, T(\underline{\alpha})) \tag{52}$$

holds, then $h(\cdot)$ can immediately be substituted by an alternative function $h^*(\cdot)$ that only needs to be defined in a reduced domain $T(\cdot)$.

For example, if the shape color is symmetric with respect to the object geometry, we can easily add a color vector $\underline{c}$ to a spatial point vector $\underline{z}$ and apply aggregation in the form of $T([\underline{z}^{\mathrm{T}}, \underline{c}^{\mathrm{T}}]^{\mathrm{T}})$ that maps the spatial dimensions to its equivalent in the non-redundant part while leaving the color untouched. This generalization also applies to measurements related to curvature or other features, as long as they are symmetric with respect to the object geometry.

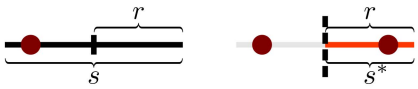
REMARK 4 **(Benefits)**
The proposed approach allows for modeling an entire shape by solely specifying it in the non-redundant part of the domain, which then is "unfolded" to obtain the entire domain. In turn, each measurement forces the estimator to adjust the shape in the entire domain. Depending on the specific source model, shape, and its parametrization, symmetric simplification allows for

- introducing symmetry constraints upon the estimated shape,
- modeling a more detailed shape while maintaining or even reducing the number of shape parameters,
- reducing the overall complexity by reducing the domain of integration variables, and
- increasing robustness against partial occlusion.

Next, we demonstrate the proposed approach by means of two illustrative examples.

9 EXAMPLE 1: STICK OBJECT

A stick object is a one-dimensional line segment, usually embedded in a higher-dimensional space. We will first derive a simplified source model for the one-dimensional space, and then show a straightforward way to extend this model into higher dimensional space. According to (1), the state $\underline{x}$ of an object is separated into pose $\underline{x}^{\mathrm{pose}}$ and shape $\underline{x}^{\mathrm{shape}}$. For the source model, the stick is modeled as being centered on the origin. Pose will be incorporated in the measurement function.



(a) common (b) proposed

Fig. 8. Difference between modeling the stick without (a) and with (b) symmetry.

The only required shape parameter refers to the size of the stick, which is represented by the distance $\underline{x}^{\text{shape}} = r$ from the center to the edge, i.e., $2 \cdot r$ is the stick length.

9.1 No Symmetry

Common approaches [1], [9], [24] use an SDM (9) for the stick (see Fig. 8(a)), where the shape $\tilde{\mathcal{Z}}(\underline{x})$ is parameterized by the sources

$$\tilde{z}(\underline{x}, s) = s \cdot r \tag{53}$$

with $s \in [-1, 1]$. Within this parametrization $\tilde{z}(\underline{x}, -1) = -r$ specifies the left edge of the stick, and $\tilde{z}(\underline{x}, 1) = r$ specifies the right edge, respectively. If the sources are assumed to be uniformly distributed along the stick, the parameter s follows a uniform distribution $p(s) = \mathcal{U}[-1, 1]$. A measurement function is then given by

$$h(\underline{x}, y, w, s) = H(\underline{x}, (y - w)) - s \cdot r, \tag{54}$$

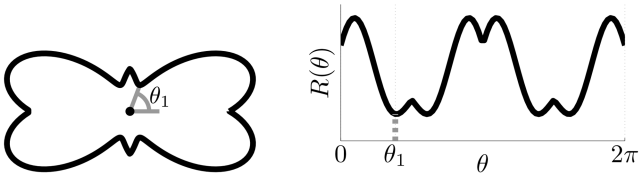
where $H(\underline{x}, (y - w))$ converts points $y - w$ to object coordinates. Even though this model is simple, it has the drawback that estimators based on the linear Gaussian assumption [10], [23] are not capable to estimate r when simultaneously estimating the pose [1], as an effect of the linearization. They propose a *quadratic extension*, in order to design a modified measurement function that overcomes this issue. However, even this approach only works when using advanced filters such as the S^{2}KF [23].

9.2 1-axial Symmetry

We now propose a novel model that can be used for simultaneously estimating shape and pose even with a standard UKF [10]. This model can be obtained by approximating the common SDM by a simplified SDM, which exploits the 1-axial symmetry of the stick.

Specifying Aggregation Function: This symmetry can be described by the aggregation function $T(z) = |z|$ as for all sources $\tilde{z}(\underline{x}, s)$ on the stick, it follows that $|\tilde{z}(\underline{x}, s)|$ also lies on the stick. Based on $T(\cdot)$, the non-redundant part of the stick then falls on the positive z-axis, where the sources are indexed by $\mathcal{S}^* = [0, 1]$.

Creating Symmetric Subshapes: The equivalence class of indices is given by $[s^*] = \{-s^*, s^*\}$ which generates subshapes $\tilde{\mathcal{Z}}(\underline{x}, s^*) = \{-s^* \cdot r, s^* \cdot r\}$ that all consist of two opposing sources, as illustrated in Fig. 8(b). As we assumed the occurrence of sources to be uniformly distributed $p(s) = \mathcal{U}[-1, 1]$, the distribution of the subshapes, according to (42), is also uniformly distributed $p(s^*) = \mathcal{U}[0, 1]$.



(a) object contour (b) radius function

Fig. 9. The polar function $R(\theta)$ specifies a radius for each angle θ. The correspondence between polar function and shape contour is marked for a given θ_1.

Simplifying the RHM: An appropriate simplified shape function $g^*_{s^*}(\underline{x}, z)$, which only needs to be specified on the positive z-axis and returns 0 for subshapes $\tilde{\mathcal{Z}}(\underline{x}, s^*)$ is given by $g^*_{s^*}(\underline{x}, z) = |z| - s^* \cdot r$. This shape function yields the measurement function

$$\begin{aligned} h(\underline{x}, y, w, s^*) &= g^*_{s^*}(\underline{x}, H(\underline{x}, y - w)) \\ &= |H(\underline{x}, (y - w))| - s^* \cdot r, \end{aligned} \tag{55}$$

which can be immediately used in Alg. 1. It is interesting to compare the measurement functions of the common stick SDM (54) and its simplified version (55). Besides the different integration variables, they only differ in the sense that the simplified SDM additionally requires taking an absolute value. The benefit of using this modified model with a sample-based estimator is that s^* only has to be sampled in $[0, 1]$ instead of $[-1, 1]$. This results in doubling the effective sample resolution.

Extension to Two-Dimensional Space: In two-dimensional space, as seen from the object frame, the stick represents a segment parallel to the z_1-axis that is centered on the origin. Extending the measurement function (54) is straighforward and yields

$$h(\underline{x}, \underline{y}, \underline{w}, s^*) = \begin{bmatrix} g_{s^*}(\underline{x}, (y_1 - w_1)^O) \\ (y_2 - w_2)^O \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{56}$$

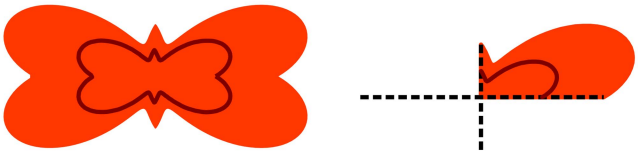
where $[(y_1 - w_1)^O, (y_2 - w_2)^O]^{\mathrm{T}} = H(\underline{x}, \underline{y} - \underline{w})$ represents points transformed into object coordinates using H.

10 Example 2: Star-Convex Object

In [3], an RHM was proposed that models objects with a complex star-convex shape in $\mathbb{R}^2$. Star-convex means that there is a point within the shape, where each line segment to any point on the boundary remains in the shape. This allows for a convenient representation by means of a polar function $R(\theta)$ that gives the radius for a given angle θ from 0 to 2π, as illustrated in Fig. 9. In [3], the radius function $R(\theta)$ is implemented by means of a Fourier series

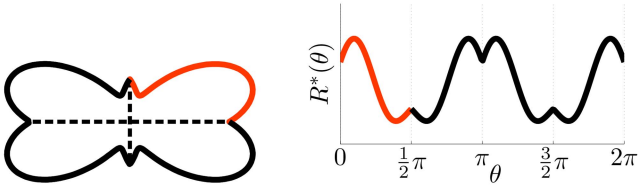
$$R(\underline{x}, \theta) = \frac{a_0}{2} + \sum_{m=1}^{M} a_m \cos\left(m \frac{2\pi}{P} \theta\right) + b_m \sin\left(m \frac{2\pi}{P} \theta\right). \tag{57}$$

This superposition of cosine and sine functions (with $P = 2\pi$) is weighted by a list of $2M + 1$ coefficients that form the shape vector $\underline{x}^{\text{shape}} = [a_0, a_1, b_1, \ldots, a_M, b_M]^{\mathrm{T}}$. The number of coefficients determines the degree of



(a) common, $t = 0.5$ (b) proposed, $t = 0.5$

Fig. 10. Difference between modeling a star-convex shape without (a) and with (b) symmetry.



(a) object contour (b) radius function

Fig. 11. Star-convex object with 2-axial symmetry, represented by a polar function. The non-redundant part of the boundary is colored orange.

shape detail the corresponding polar function is capable to encode. For example if $M = 0$, then $R(\underline{x},\theta) = a_0/2$ is independent of θ and specifies the constant radius of a circle.

As we deal with polar functions, in this section, we will occasionally use polar coordinates for points $\underline{z} \in \mathbb{R}^2$, i.e., the radius $r(\underline{z})$, and angle $\theta(\underline{z})$.

10.1 No Symmetry

In [3], a shape $\tilde{\mathcal{Z}}(\underline{x})$ was partitioned into subsets $\tilde{\mathcal{Z}}(\underline{x},t)$, resulting in an RHM according to (17) with shape function

$$g_t(\underline{x},\underline{z}) = r(\underline{z}) - t \cdot R(\underline{x},\theta(\underline{z})). \tag{58}$$

For each parameter $t \in [0,1]$, the shape function determines a set $\tilde{\mathcal{Z}}(\underline{x},t)$ that corresponds to a scaled boundary. Fig. 10(a) shows a visualization for $t = 0.5$ from the shape considered in Fig. 9. When measurement sources are assumed to be uniformly distributed over the object, then it holds that $p(t^2) = \mathcal{U}[0,1]$.

10.2 2-axial Symmetry

For a given shape, let us assume that prior knowledge is available about it having a 2-axial symmetry. We can derive a simplified RHM (38) by specifying an aggregation function and deriving a simplified shape function.

Specifying Aggregation Function: The 2-axial symmetry refers to an aggregation function according to (24) with $n = 2$ that results in the special case $T(\underline{z}) = [|z_1|,|z_2|]^{\mathrm{T}}$, i.e., it maps each point to its symmetric equivalent in the first quadrant. In consequence, the corresponding angles $\theta(\underline{z},n)$ in polar representation lie between 0 and $\pi/2$, as illustrated in Fig. 11.

Simplifying the RHM: Due to the 2-axial symmetry only one fourth of the shape has to be modeled by the radius function (see Fig. 10(b)). In order to take advantage of this reduced domain, we can change the period of the Fourier series from $P = 2\pi$ to $P^* = 2\pi/n$. Then, let $R^*(\underline{x},\theta)$ be a radius function (57) with the adjusted period P^*. The simplified model has the shape function

$$g_t^*(\underline{x},\underline{z}) = r(\underline{z}) - t \cdot R^*(\underline{x},\theta(\underline{z},n)), \tag{59}$$

The corresponding measurement function that incorporates the pose then becomes

$$\begin{aligned} h(\underline{x},\underline{y},\underline{w},t) &= g_t^*(\underline{x},T(H(\underline{x},\underline{y}-\underline{w}))) \\ &= r((\underline{y}-\underline{w})^O) - t \cdot R^*(\theta((\underline{y}-\underline{w})^O,n)) = 0, \end{aligned} \tag{60}$$

which can be immediately used in Alg. 1. The simplification has the benefit that the shape parameters only need to specify the shape in one fourth of the spatial domain, which in turn reduces the complexity of the estimation problem.

REMARK 5 **(Number of Coefficients and Symmetries)** A crucial challenge when designing an estimator based on a symmetric star-convex model is to find the number of coefficients and symmetries which reflect best the geometry of the underlying object. These concepts are not independent from each other, as increasing the number of symmetries yields a higher shape detail while maintaining the same number of coefficients. As such, even when an object is not perfectly symmetric, assuming symmetry anyway can yield an improved shape estimate. However, then the estimator finds parameters of an intermediate symmetric shape and loses the ability of a perfect shape estimate. In order to explore this trade-off, the evaluation in Sec. 11.2 includes an extensive analysis related to varying number of coefficients and symmetries.

As another short remark, note that it would also have been possible to enforce symmetry by constraining the coefficients of the Fourier series. However, these considerations would be out of the scope of this work, as our intention is to demonstrate the universality of the proposed simplification approach.

11 EVALUATION

In this section, we evaluate the proposed approach by means of a simulated tracking scenario. Our intention is to investigate the effect of applying symmetric simplification to the common models from the previous examples. As such, we exclusively focus in this work on comparing estimators based on a common non symmetric model with its symmetric version. As an example, we chose tracking the symmetric airplane object from the motivating example in Fig. 1 while following the

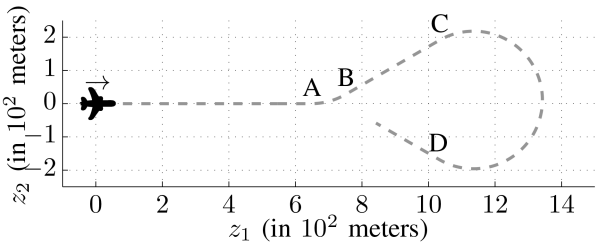
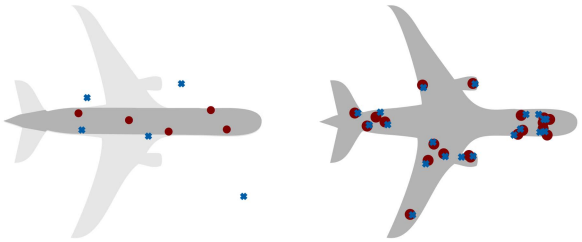


Fig. 12. Evaluation Scenario: plane object moves along the dashed, gray path with constant speed. This path is composed of three straight, and two circular parts. The labels A–D mark changes between the different motion parts.



(a) low information scenario (b) high information scenario

Fig. 13. Few measurements from fuselage with high noise for the stick evaluation (a). Many measurements from entire plane with low noise for the star-convex evaluation (b). Measurement sources are marked by red circles, measurements by blue crosses.

path depicted in Fig. 12. The object moves with constant speed and is observed by a simulated sensor. The evaluation is divided into two parts, consisting of a *low information* scenario using a stick model (Sec. 9), and a *high information* scenario using a star-convex model (Sec. 10). In the following, we differentiate between the *common approach* without using any symmetry and the *proposed approach*, which incorporates the given symmetries.

Sensor: For the low information scenario using the stick model, we implemented a sensor that obtains one to five measurements each time step uniformly from the fuselage of the airplane. The number n_k of measurements is uniformly drawn from $\{1,\dots,5\}$. The fuselage has a size of 100 m length and 20 m meters width, and its measurement sources are uniformly distributed on its extent. During observation, these measurements are corrupted with high noise, modeled as zero-mean Gaussian noise according to $\mathcal{N}(\underline{0}, 10^2 \cdot \mathbf{I})$.

For the high information scenario based on the more detailed star-convex model, the sensor obtains ten to twenty measurements each time step uniformly from the entire plane, which has a size of 100 m length and 90 m width. Again, the number n_k of measurements is uniformly drawn from $\{1,\dots,20\}$. Then, these measurements are disturbed using zero-mean Gaussian noise according to $\mathcal{N}(\underline{0}, 10^0 \cdot \mathbf{I})$. Example measurements from both sensors are illustrated in Fig. 13.

Pose and Motion: In order to reflect the authentic behavior of a plane, the path in Fig. 12 was chosen to switch between straight and circular movements. The speed was chosen to be about 50 m per time step so that it takes 400 time steps to complete one run. At each time step k, the pose model uses a rigid transformation where the parameters to be estimated are $\underline{x}_k^{\text{pose}} = [\phi_k, \underline{t}_k^{\mathrm{T}}]^{\mathrm{T}}$, with ϕ_k as the orientation angle, and $\underline{t}_k \in \mathbb{R}^2$ as the translation vector.

In addition, we use a motion model that assumes a constant speed and a constant turn rate, similar to [21]. This model requires estimating two further state parameters, $\underline{x}_k^{\text{motion}} = [\dot{\phi}_k, v_k]^{\mathrm{T}}$, where $\dot{\phi}_k$ is the angular speed, and v_k is the object speed. The velocity is then assumed to be the vector $[1,0]^T$ rotated by the orientation ϕ_k and scaled by the magnitude v_k. We do not employ any switching structure.

TABLE I
Compared approaches in the stick tracking experiment.

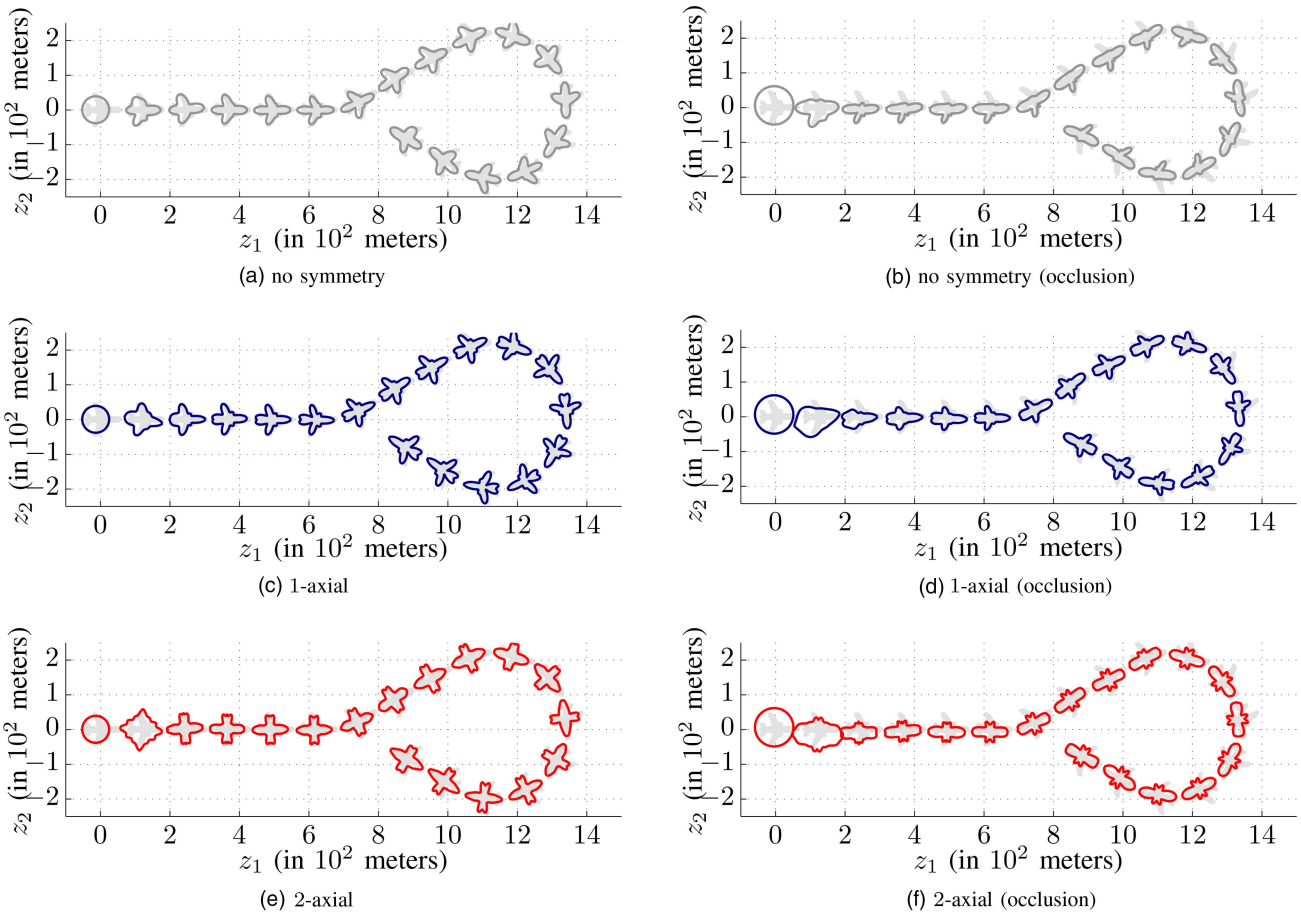
	UKF [10]	S^2KF [23]
no symmetry (quad. extension) [1]	common-1	common-2
1-axial symmetry	proposed-1	proposed-2

11.1 Tracking Stick Object

First, we look at the low information scenario with the stick model. We compare the common approach from [1] to the modified approach that exploits the 1-axial symmetry, both explained in Sec. 9. Both models require estimating the shape parameter $\underline{x}_k^{\text{shape}} = r_k$. The combined state parameters form the 6×1 vector $\underline{x}_k = [(\underline{x}_k^{\text{pose}})^{\mathrm{T}}, (\underline{x}_k^{\text{shape}})^{\mathrm{T}}, (\underline{x}_k^{\text{motion}})^{\mathrm{T}}]^{\mathrm{T}}$. From the initial measurements, where we required a minimum of two points, $\underline{x}_0^{\text{pose}}$ was determined by total least squares, r_k from $\underline{x}_0^{\text{shape}}$ was set to half of the distance between the farthest points, and $\underline{x}_0^{\text{motion}}$ was initialized to $\underline{0}$. The initial state covariance matrix was set to $\mathbf{C}_{\underline{x}_0} = \text{diag}(10^{-2}, 10^3, 10^3, 10, 10^{-3}, 10)$. The system covariance matrix was set to the constant $\text{diag}(10^{-9}, 10^{-7}, 10^{-7}, 1, 10^{-6}, 10^{-3})$. Then, 100 runs of the experiment were performed.

We compare two sample-based estimators, both based on the common model and the proposed model, as depicted in Tab. I. The UKF was implemented according to [10], using two samples per dimension plus one, and parameters $\alpha = 1$, $\beta = 0$, $\kappa = 1/2$. The S^2KF was implemented according to [23] using five samples per dimension.

Results: The *root mean squared errors* (RMSEs) for position, angle and stick length are depicted in Fig. 15. In the estimated pose parameters, all approaches show a quite similar performance, i.e., about 4 m position error and 3° orientation error. Note that all estimators need some time steps to adapt to the motion changes at A, B, C, and D, as we do not use any switching motion models. As an important result, it can be seen that "proposed-1" and "proposed-2" both perform better in estimating the length of the stick. As can be seen in Fig. 15(c), the common SDM [1] imposes special

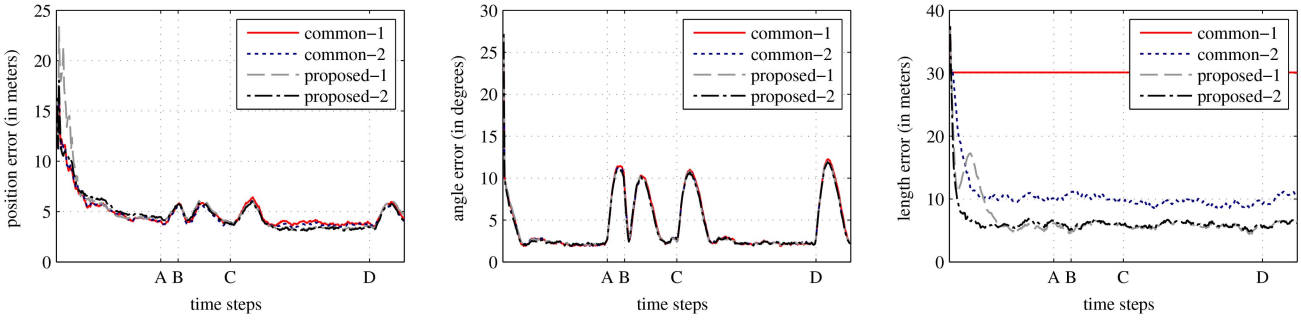


(a) no symmetry (b) no symmetry (occlusion)

(c) 1-axial (d) 1-axial (occlusion)

(e) 2-axial (f) 2-axial (occlusion)

Fig. 14. Estimation result of the star-convex shape when using 13 Fourier coefficients for selected time steps.



(a) position RMSE (b) angle RMSE (c) length RMSE

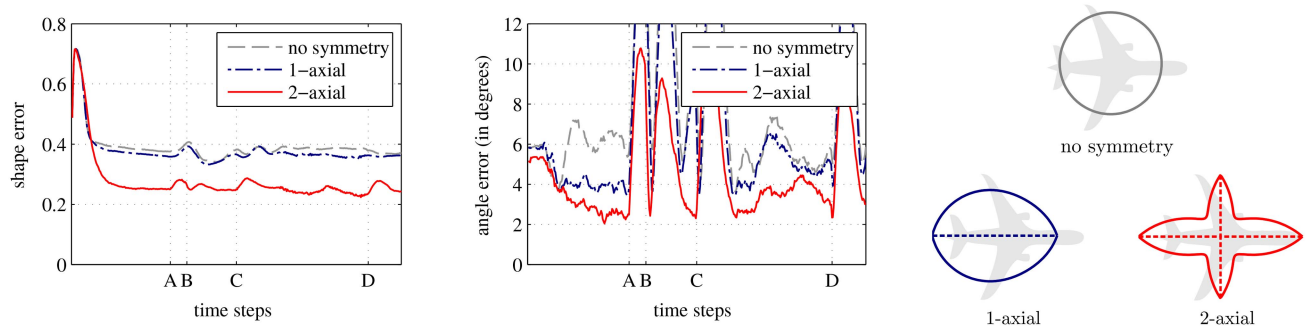
Fig. 15. Evaluation of the stick object.

requirements to the filter being used, as "common-2" estimates the length correctly and "common-1" does not. In contrast, the simplified SDM in "proposed-1" and "proposed-2" has no special requirements to the filter being used. This is a significant improvement, especially because the simple UKF in "proposed-1" also outperforms the advanced S^2KF in "common-2."

11.2 Tracking Star-convex Object

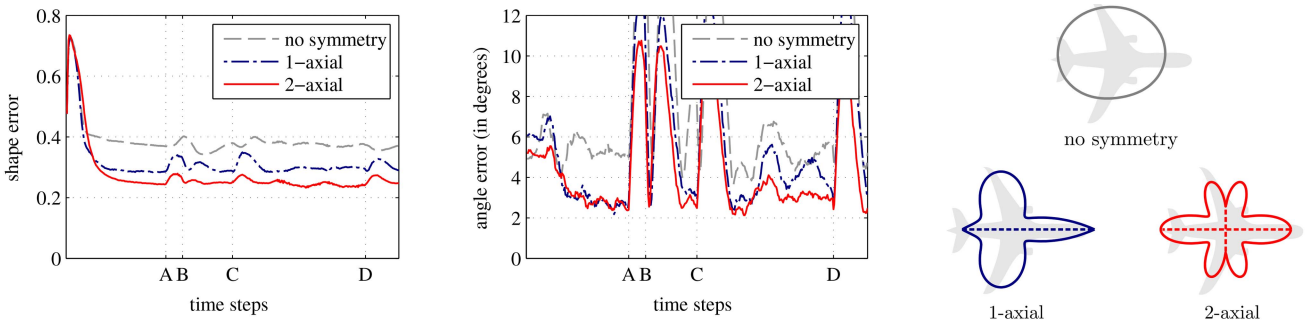
Next, we performed the high information experiment using the star-convex models presented in Sec. 10. We compare the common RHM-approach from [3] to the modified approach based on a simplified RHM that assumes different axial symmetries. In addition, we varied the amount of Fourier coefficients in order to investigate their effects on the estimation result. Specifically, we set up models with "no symmetry," "1-axial symmetry," and "2-axial symmetry," each with $M = 3, 5, 7, 9, 11, 13$, and 15 coefficients, respectively.

A UKF [10] with equal parametrization as in the previous scenario was used for estimating the parameters of all models. The combined state parameters form the $(5 + M) \times 1$ vector $\underline{x}_k = [(\underline{x}_k^{\text{pose}})^{\text{T}}, (\underline{x}_k^{\text{shape}})^{\text{T}}, (\underline{x}_k^{\text{motion}})^{\text{T}}]^{\text{T}}$. From the initial measurements, where we required a mini-



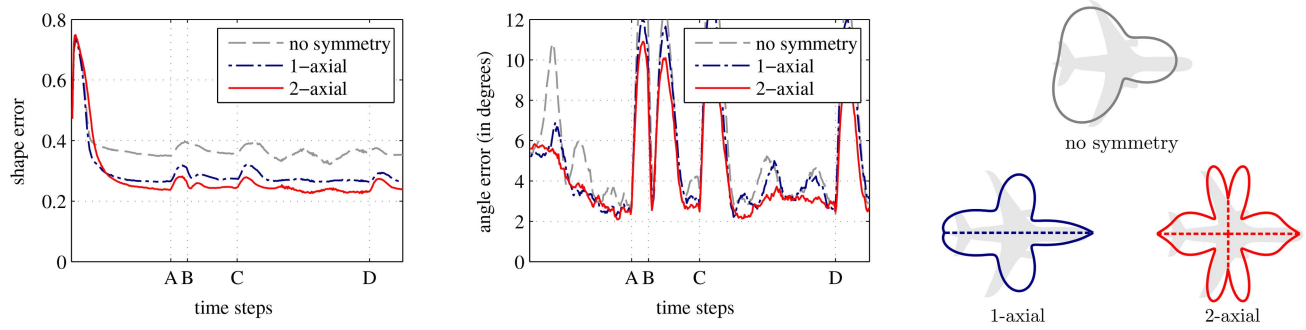
(a) shape RMSE (b) angle RMSE (c) representative estimated shape

Fig. 16. Evaluation of the star-convex object with 3 Fourier coefficients.



(a) shape RMSE (b) angle RMSE (c) representative estimated shape

Fig. 17. Evaluation of the star-convex object with 5 Fourier coefficients.

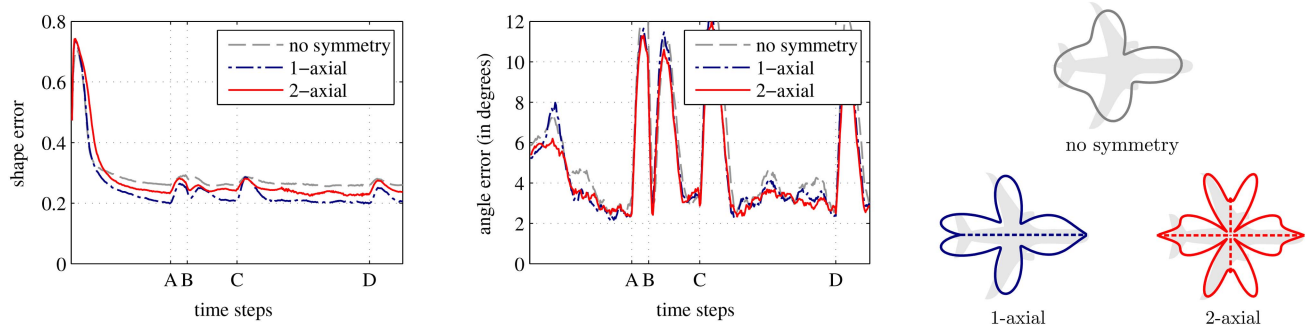


(a) shape RMSE (b) angle RMSE (c) representative estimated shape

Fig. 18. Evaluation of the star-convex object with 7 Fourier coefficients.

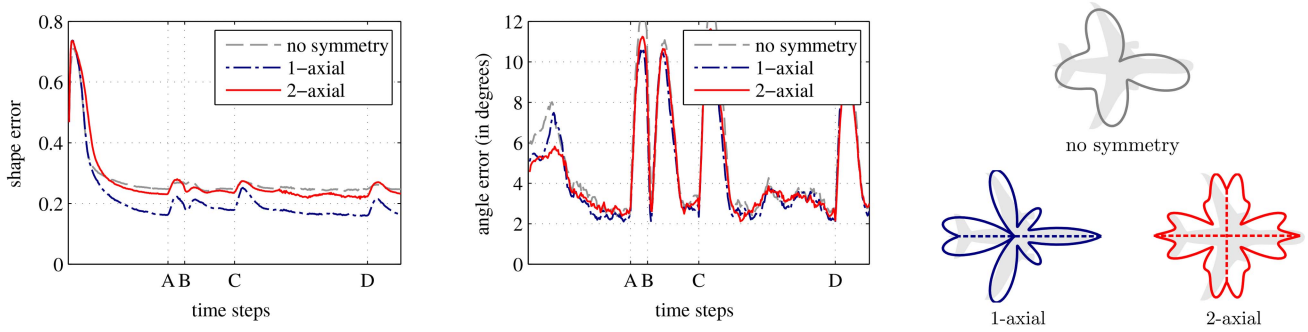
mum of two points, $\underline{x}_0^{\text{pose}}$ was determined as follows. The mean of the points was used as $\underline{t}_0$, and the orientation was set to $\theta_k = 0$. For $\underline{x}_0^{\text{shape}}$, the first Fourier coefficient was set to half of the distance between the farthest points, and all others to zero, which corresponds to a circle with mean $\underline{t}_0$ and radius given by the first coefficient. Again $\underline{x}_0^{\text{motion}}$ was initialized with $\underline{0}$. The initial state covariance was set to $\mathbf{C}_{\underline{x}_0} = \text{diag}(10^{-1}, 10^{3}, 10^{3}, 10^{-1}, \ldots, 10^{-3}, 10)$. The system covariance matrix was set to the constant $\text{diag}(10^{-6}, 10^{-2}, 10^{-2}, 10^{-6}, \ldots, 10^{-6}, 10^{-2})$. Then, 100 runs of the experiment were performed.

Results: In order to compare estimation results of the different approaches we chose the "intersection over union" [5], where we calculated an RMSE from $1 - (\textit{intersection}/\textit{union})$. Its values range from 1 to 0, where 0 means no intersection of the estimate and the groundtruth and 0 means a perfect match. The behavior of this error, as shown Fig. 16–22(a), indicates the convergence behavior of the shape, as well as the position. The estimation process over a run is illustrated in Fig. 14(a,c,e). Representative shape estimates after a single run are depicted in Fig. 16–22(c), respectively. In addition, Fig. 16–22(b) shows the RMSE of the estimated heading angle. All approaches initially converge no later than time step A and, particularly, we did not observe any run, where the estimator diverged. The increased error at time steps A, B, C, D, are caused by the changing motion of the plane.



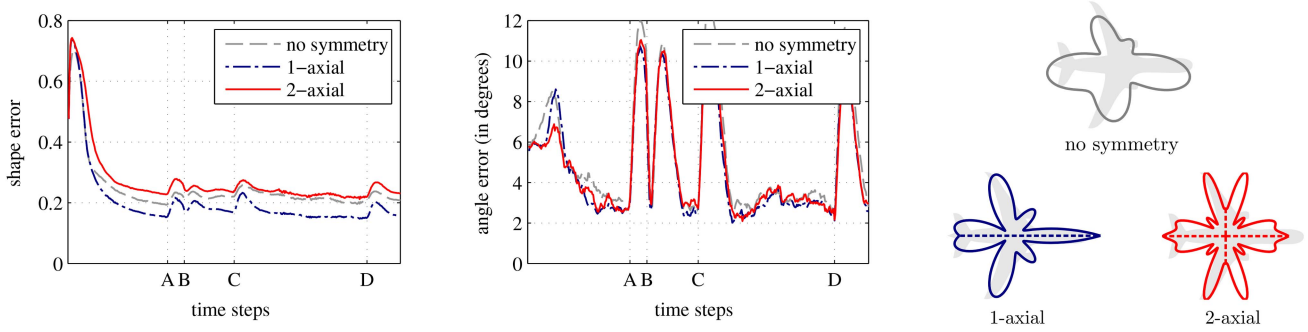
(a) shape RMSE (b) angle RMSE (c) representative estimated shape

Fig. 19. Evaluation of the star-convex object with 9 Fourier coefficients.



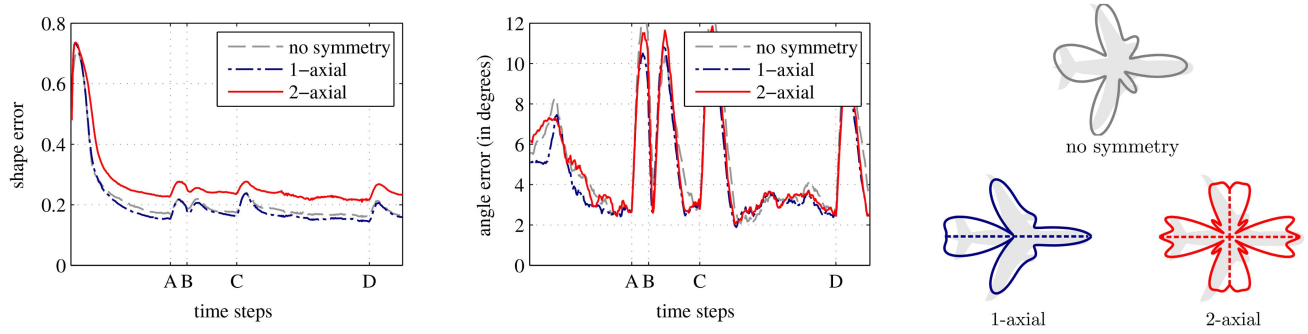
(a) shape RMSE (b) angle RMSE (c) representative estimated shape

Fig. 20. Evaluation of the star-convex object with 11 Fourier coefficients.



(a) shape RMSE (b) angle RMSE (c) representative estimated shape

Fig. 21. Evaluation of the star-convex object with 13 Fourier coefficients.



(a) shape RMSE (b) angle RMSE (c) representative estimated shape

Fig. 22. Evaluation of the star-convex object with 15 Fourier coefficients.

Let us now discuss the trade-off between number of coefficients and symmetries, as mentioned in Remark 5. The proposed approach with 1-axial symmetry outperforms the common approach for all numbers of coefficients. Introducing the second symmetry axis yields a benefit over the 1-axial approach up to seven coefficients as the additional symmetry results in the capability of capturing a higher degree of shape detail while maintaining the number of parameters. This can particularly be seen in Fig. 16(c), where the 2-axial finds a reasonable shape when using only three Fourier coefficients. The 1-axial model (and the common approach) are not capable of representing the shape before five (seven respectively) parameters are introduced.

However, for the case when a higher number of Fourier coefficients is used, the 1-axial model steadily outperforms the 2-axial model, as the underlying shape, i.e., the plane, has only a 1-axial symmetry. Note that the 2-axial approach is never capable of estimating the shape correctly, as it always converges to an intermediate shape. For the common approach, it is worth pointing out that the estimated result never displays any of the underlying symmetries. Again, it should be noted that, as with the stick, incorporating symmetry into the star-convex model yields a superior tracking result while only marginally changing the generative model.

Results Under Partial Occlusion: In order to evaluate the effect of using symmetries in the presence of occlusion, we modified the high information scenario by removing all measurements from the left wings of the plane. Qualitative results are shown in Fig. 14(b,d,f). As expected, the common approach estimates a non-symmetric shape without the left wings. In contrast, the proposed approaches impose symmetric constraints and, thus, estimate shapes which have wings on both sides. It is remarkable that the symmetry axes are found, even though there are no measurements from the left side at any time. This is due to the fact that the model assumes symmetry with respect to the direction of velocity.

12 CONCLUSION

In this work, we presented a concept for designing a simplified Bayesian estimator for extended object tracking that exploits symmetries in the object geometry. We proposed simplification of estimators based on Spatial Distribution Models (SDM), Greedy Association Models (GAM), and Random Hypersurface Models (RHM). The key idea is to aggregate equivalent points according to the object symmetry, such that the source model only needs to be evaluated in the non-redundant part of the spatial domain. In doing so, a simplification scheme was obtained that yields two major improvements:

1. For all source models, simplified versions can be derived that allow for reducing the number of shape parameters while estimating an equally detailed shape. The benefit to expect is proportional to the reduction of the spatial extent from the entire shape to the non-redundant part.
2. Simplifying SDMs additionally requires aggregating symmetric sources and their individual probabilities. This aggregation turns the SDM into an RHM that achieves a higher spatial sample resolution in sample-based filters. The benefit is again in the same order of magnitude, as it also depends on the ratio between the spatial extent of the entire shape and its non-redundant part.

Both simplifications reduce the complexity of the estimation problem and yield an improved estimation result, as we illustrated by means of two examples: A simple stick object and a star-convex object. In particular, we showed that applying the proposed simplification to a common stick model yields an RHM that can be used with a UKF, while the original model requires a more advanced filter. In numerical terms, the proposed new stick RHM reduces the length error about 60%. Using the symmetric star-convex model in the considered scenario reduces the orientation error up to 50%, and the shape error up to 30%, depending on the number of Fourier coefficients. As an additional benefit, we showed that our approach uses measurement information from visible parts of the object in order to infer the shape of occluded, symmetric counterparts.

Future research will include a closer look at the effect that object symmetries have in the estimated state density, particularly in the form of multiple modes. Incorporating a mechanism that automatically chooses the appropriate type of symmetry, e.g., based on a symmetric descriptor [11], is also an interesting research topic. Finally, it is also worth exploring and evaluating other types of symmetry, e.g., periodicity, in order to simplify source models for a wider spectrum of shapes.

REFERENCES

[1] M. Baum, F. Faion, and U. D. Hanebeck
Modeling the Target Extent with Multiplicative Noise.
In *Proceedings of the 15th International Conference on Information Fusion (FUSION)*, pages 2406–2412, Singapore, 2012.

[2] M. Baum and U. D. Hanebeck
Random Hypersurface Models for Extended Object Tracking.
In *2009 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 178–183, Ajman, United Arab Emirates, Dec. 2009. IEEE.

[3] M. Baum and U. D. Hanebeck
Shape Tracking of Extended Objects and Group Targets with Star-Convex RHMs.
In *Proceedings of the 14th International Conference on Information Fusion (FUSION)*, pages 1–8, Chicago, Illinois, USA, 2011.

[4] M. Baum, V. Klumpp, and U. D. Hanebeck
A Novel Bayesian Method for Fitting a Circle to Noisy Points.
In *Proceedings of the 13th International Conference on Information Fusion (FUSION)*, Edinburgh, United Kingdom, 2010.

[5] Benjamin Sapp, Chris Jordan, and Ben Taskar
Adaptive Pose Priors for Pictorial Structures.
In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 422–429, San Francisco, USA, 2010.

[6] F. Faion, M. Baum, and U. D. Hanebeck
Tracking 3D Shapes in Noisy Point Clouds with Random Hypersurface Models.
In *Proceedings of the 15th International Conference on Information Fusion (FUSION)*, pages 2230–2235, Singapore, 2012.

[7] F. Faion, A. Zea, and U. D. Hanebeck
Reducing Bias in Bayesian Shape Estimation.
In *IEEE 17th International Conference on Information Fusion (FUSION)*, pages 1–8, Salamanca, Spain, 2014. IEEE.

[8] M. Feldmann, D. Franken, and W. Koch
Tracking of Extended Objects and Group Targets Using Random Matrices.
IEEE Transactions on Signal Processing, 59(4):1409–1420, Apr. 2011.

[9] K. Gilholm and D. Salmond
Spatial Distribution Model for Tracking Extended Objects.
IEE Proceedings on Radar, Sonar and Navigation, 152(5): 364–371, 2005.

[10] S. J. Julier and J. K. Uhlmann
Unscented Filtering and Nonlinear Estimation.
Proceedings of the IEEE, 92(3):401–422, Mar. 2004.

[11] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz
Symmetry Descriptors and 3D Shape Matching.
In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing—SGP*, pages 115–123, New York, New York, USA, 2004. ACM Press.

[12] G. Kurz, I. Gilitschenski, S. J. Julier, and U. D. Hanebeck
Recursive Estimation of Orientation Based on the Bingham Distribution.
In *Proceedings of the 16th International Conference on Information Fusion (FUSION)*, pages 1487–1494, Istanbul, Turkey, 2013.

[13] J. Lan and X. R. Li
Tracking of Maneuvering Non-Ellipsoidal Extended Object or Target Group Using Random Matrix.
IEEE Transactions on Signal Processing, 62(9):2450–2463, May 2014.

[14] W. F. Leven and A. D. Lanterman
Unscented Kalman Filters for Multiple Target Tracking With Symmetric Measurement Equations.
IEEE Transactions on Automatic Control, 54(2):370–375, Feb. 2009.

[15] W. Li and L. Kleeman
Real Time Object Tracking Using Reflectional Symmetry and Motion.
In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2798–2803, Beijing, China, 2006.

[16] S. Loncaric
A Survey of Shape Analysis Techniques.
Pattern Recognition, 31:983–1001, 1998.

[17] N. J. Mitra, M. Pauly, M. Wand, and D. Ceylan
Symmetry in 3D Geometry: Extraction and Applications.
Computer Graphics Forum, 32(6), 2013.

[18] N. Petrov and L. Mihaylova
A Novel Sequential Monte Carlo Approach for Extended Object Tracking Based on Border Parameterisation.
In *Proceedings of the 14th International Conference on Information Fusion (FUSION)*, pages 306–313, Chicago, Illinois, USA, 2011.

[19] T. Riklin-Raviv, N. Kiryati, and N. Sochen
Segmentation by Level Sets and Symmetry.
In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 1015–1022, New York, NY, USA, 2006.

[20] A. Rivers, F. Durand, and T. Igarashi
3D Modeling with Silhouettes.
ACM Transactions on Graphics, 29(4):1, July 2010.

[21] X. Rong Li and V. Jilkov
Survey of Maneuvering Target Tracking. Part I: Dynamic Models.
IEEE Transactions on Aerospace and Electronic Systems, 39(4):1333–1364, Oct. 2003.

[22] P. Simari, E. Kalogerakis, and K. Singh
Folding Meshes: Hierarchical Mesh Segmentation Based on Planar Symmetry.
In *Eurographics Symposium on Geometry Processing*, 2006.

[23] J. Steinbring and U. D. Hanebeck
S2KF: The Smart Sampling Kalman Filter.
In *Proceedings of the 16th International Conference on Information Fusion (FUSION)*, pages 2089–2096, Istanbul, Turkey, 2013.

[24] M. Werman and D. Keren
A Bayesian Method for Fitting Parametric and Nonparametric Models to Noisy Data.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(5):528–534, 2001.

Florian Faion received his Dipl.-Inform. in computer science from the Karlsruhe Institute of Technology (KIT), Germany, in 2010. Currently, he is working towards a Ph.D. degree at the Intelligent Sensor-Actuator-Systems Laboratory, Karlsruhe Institute of Technology (KIT), Germany. His research interests are in the field of extended object tracking, shape estimation and telepresence.

Antonio Zea received his Dipl.-Inform. in computer science from the Karlsruhe Institute of Technology (KIT), Germany, in 2013. Currently, he is working towards a Ph.D. degree at the Intelligent Sensor-Actuator-Systems Laboratory, Karlsruhe Institute of Technology (KIT), Germany. His research interests are in the field of extended object tracking, shape estimation and telepresence.

Marcus Baum is an assistant research professor at the University of Connecticut. He received his Dipl.-Inform. degree in computer science from the Universitaet Karlsruhe (TH), Germany, in 2007, and his a Dr.-Ing. degree at the Intelligent Sensor-Actuator-Systems Laboratory, Karlsruhe Institute of Technology (KIT), Germany, in 2013. His research interests are in the field of extended object tracking, nonlinear estimation, and sensor data fusion. Since 2014, he is the associate administrative editor for the Journal of Advances in Information Fusion (JAIF).

Uwe D. Hanebeck is a chaired professor of Computer Science at the Karlsruhe Institute of Technology (KIT) in Germany and director of the Intelligent Sensor-Actuator-Systems Laboratory (ISAS). Since 2005, he is the chairman of the Research Training Group RTG 1194 "Self-Organizing Sensor-Actuator-Networks" financed by the German Research Foundation.

Prof. Hanebeck obtained his Ph.D. degree in 1997 and his habilitation degree in 2003, both in Electrical Engineering from the Technical University in Munich, Germany.

His research interests are in the areas of information fusion, nonlinear state estimation, stochastic modeling, system identification, and control with a strong emphasis on theory-driven approaches based on stochastic system theory and uncertainty models. Research results are applied to various application topics like localization, human-robot-interaction, assistive systems, sensor-actuator-networks, medical engineering, distributed measuring system, and extended range telepresence. Research is pursued in many academic projects and in a variety of cooperations with industrial partners.

Uwe D. Hanebeck was the General Chair of the "2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2006)," Program Co-Chair of the "11th International Conference on Information Fusion (Fusion 2008)," Program Co-Chair of the "2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2008)," Regional Program Co-Chair for Europe for the "2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)," and will be General Chair of the "19th International Conference on Information Fusion (Fusion 2016)." He is a Member of the Board of Directors of the International Society of Information Fusion (ISIF), Editor-in-chief of its Journal of Advances in Information Fusion (JAIF), and associate editor for the letter category of the IEEE Transactions on Aerospace and Electronic Systems (TAES). He is author and coauthor of more than 350 publications in various high-ranking journals and conferences.

Simulating Aerial Targets in 3D Accounting for the Earth's Curvature

DAVID FREDERIC CROUSE

Most dynamic models for target tracking are derived for a flat Earth. When simulating the trajectories of targets over long ranges, the curvature of the Earth becomes important. This paper demonstrates how most nonlinear, 3D, continuous-time, flat-Earth dynamic models can be easily adapted for use on a curved Earth without incorrectly spiraling into the poles as is typical of previous techniques. The algorithm is demonstrated with flat-Earth non-maneuvering, turning, weaving, and spiraling dynamic models. The underlying technique can also compute geodesic curves on or above the Earth in a singularity-free manner when used with a non-maneuvering dynamic model.

Manuscript received July 16, 2014; revised October 29, 2014; released for publication October 30, 2014.

Refereeing of this contribution was handled by Huimin Chen.

This paper is an extended version of the conference paper [11].

Author's address: Naval Research Laboratory, Radar Division, Code 5344, 4555 Overlook Ave., SW, Washington, DC 20375-5320, USA, e-mail: (david.crouse@nrl.navy.mil).

THE QUICK AND DIRTY IMPLEMENTATION

The mapping of a flat-Earth, continuous-time dynamic model to a curved-Earth is embodied in the rewritten drift function of (33), which rewrites the function in terms of new, rotating bases. Readers who want to quickly map continuous-time flat-Earth dynamic models quickly to an ellipsoidal Earth approximation without going through the details of the mathematics can skip to the algorithmic summary in Fig. 6. The flat-Earth dynamic model should consist of, at a minimum, position and velocity components as in (6).

I. INTRODUCTION

The modeling of aerospace vehicle dynamics with varying degrees of freedom and accounting for the Earth's gravity has been extensively studied, for example in [84], where a number of models are presented. However, such curved-Earth target models often require knowledge about basic aircraft aerodynamic parameters, such as the lift, thrust, and drag. On the other hand, target tracking algorithms in radar systems must be able to maintain track on targets with known aerodynamic models as well as on targets that cannot be identified. Thus, a number of lower-fidelity dynamic models, such as those surveyed in [40] and [41], are commonly used in tracker simulations. These low-fidelity simulations can often be defined using a number of simple parameters, like the limits on the speed, turn rate, range and ceiling of three well-known fighter aircraft shown in Table I. The simulations in Section V use these bounds to justify the realism of the simulated scenarios. Unfortunately, most of the non-ballistic models in [40] as well as the ballistic models in [41] and [83, Ch. 8, 9] are made ignoring the curvature of the Earth. When tracking targets using networked radars, the curvature and undulation of the Earth can become significant. For example, if a flat-Earth dynamic model is propagated over a long distance, a target in level flight under a flat-Earth dynamic model will tangentially depart a curved Earth. This paper derives a technique for using arbitrary, nonlinear, continuous-time, deterministic, flat-Earth dynamic models on a curved Earth that mitigates this error.

TABLE I
A number of basic parameters for Russian MiG-29, the Chinese and Pakistani FC-1, and the American F-16 fighter aircraft are taken from [30] and [75] (No specific model of F-16 is used). The parameters listed above are not sufficient for high-fidelity modeling of the aerodynamics of the aircraft, but can set bounds for simple models when testing tracking algorithms.

	MiG-29	FC-1	F-16
Positive G-Force Limit	9 G	8.5 G	9 G
Maximum Speed (At Altitude)	679 m/s	347 m/s	670 m/s
Service Ceiling	17 km	16 km	> 15 km
Maximum Range	1,500 km	2,037 km	> 3,222 km

The best-known published attempts to adapt flat-Earth dynamic models for use on a curved Earth for target tracking are [43] and [81], where flat-Earth discrete-time dynamic models are applied in local East-North-Up (ENU) coordinate systems that rotate as the aircraft moves above the surface of the Earth.[1] However, such a constant-bearing approximation is only valid when going East-West or North-South, but not at any other bearings, and the approximation becomes extremely bad near and at the poles. Considering that the sea ice in the Northwest Passage sufficiently melted for the first time in recorded history in 2007 to allow travel through the passage [8], it is increasingly likely that ships (possibly carrying radars) will regularly approach the North Pole in the future.

On the other hand, the curvature of an ellipsoidal Earth has been properly handled in the design of inertial navigation systems, where a wander coordinate system, also known as a wander frame, the wander azimuth coordinate frame, and the wander azimuth navigation frame, is often maintained in place of a local East-North-Up coordinate system. Such a coordinate system keeps one axis aligned with the gravitational (or ellipsoidal) vertical, and the other axes are kept in the local tangent plane, without any particular relation to true North. The wander coordinate system is used in [27], [46] for strapdown inertial navigation systems, where gyroscopes are fixed to the body of an aircraft rather than remaining on a platform that tries to physically rotate to maintain a constant orientation to true North which is increasingly difficult as one approaches the poles.[2] The wander coordinate system is used in [50] for simulating aircraft trajectories under continuous-time dynamic models on a curved Earth, where solutions for four dynamic models are provided. More recently, the method of [50] was rediscovered and applied to curved-Earth target simulation in [62], [63], where the mapping of a flat-Earth model to a curved-Earth is referred to as a "geodetic projection." In this paper, a method of mapping flat-Earth dynamic models to a curved Earth is derived in vector notation, generalized to describe motion on an arbitrary terrain or above an ellipsoidal Earth. Under an ellipsoidal Earth model, when considering similar dynamic models, the algorithm of this paper reduces to the solutions of [50], [62].

In [73, Sec. 5], similar physical principles to those used in the models of [84, Ch. 8, 9, 10] (which require a detailed description of the flight dynamics of the aircraft) are used to obtain simplified dynamic expressions for straight, level flight, for steady turns, and for pitching flight (weaving up and down) above a spherical Earth. However, no expressions for spiraling (corkscrewing) targets are given, with the justification being that the models would have to be overly simplified for a solution to be obtained. In the following sections, it is shown that most flat-Earth dynamic models can be plugged into a simple algorithm to adapt them for use above a curved Earth, including a spiraling model derived in Appendix E.

A constant-bearing course means that a target maintains the same direction with respect to a local ENU coordinate system. As described in [1], a constant-bearing course is called a rhumb, whereas the curve on the surface of the Earth traced out by such a course is a loxodrome. An example of a constant-bearing coarse is a ship traveling 20° North of East. A constant-bearing course is a straight line on a Mercator projection map. However, as clearly illustrated in Fig. 7(c) in Section V and as discussed in detail in [1], constant-bearing courses that do not follow East-West or North-South trajectories spiral infinitely as an object approaches the poles. The algorithms of [43] and [81] cause straight-line targets to follow rhumb lines. A rhumb is seldom the shortest trajectory between any two points and a navigator must constantly update a vehicle's bearing to travel the shortest path.[3]

Before taking the time to implement a complicated curved-Earth simulation routine, one should first check whether the scale of the bias accumulated as a result of ignoring the Earth's curvature matters. As derived in [10], a simple closed-form approximation for the bias, Δ, accumulated using a local flat-Earth approximation can be derived using a spherical Earth approximation,

$$\Delta \approx \sqrt{a^2\left(1-\cos\left[\frac{L}{a}\right]\right)^2+\left(L-a\sin\left[\frac{L}{a}\right]\right)^2}, \quad (1)$$

where L is the distance traveled and a is the radius of the spherical Earth. For example, using the semi-major axis $a = 6,378,137.0$ m from the reference ellipsoid of the U.S. Department of Defense's (DoD) World Geodetic System 1984 (WGS-84) standard [13], a target traveling five kilometers in a straight line would accumulate a bias of only 1.96 m. On the other hand, a target traveling 500 km would accumulate a bias of 19.6 km, which is probably significant in most instances.

Considerable research over the years has focused on following the shortest and straightest path along a curved Earth or other surface. It is reasonable to assume that a target obeying a flat-Earth dynamic model on a curved Earth experiences no accelerations in the local tangent plane to the surface beyond those present in the flat-Earth model. That is, the target is not pulled left or right by any accelerations that are not part of

[1]When considering only straight-line trajectories, algorithms such as [19] are also relevant. However, this paper considers more general dynamic models.

[2]Gyroscopic systems can infer the direction of true North based on the precession of the gyroscope as the Earth rotates, as observed by Foucault in 1852 [22].

[3]For example, a rhumb-line-based navigation technique was implemented for operational use in the U.S. Navy's Navigation Sensor System Interface (for ships) when handling optical celestial observations in the 1990s [31], [32]. However, in the presence of other navigation sources (for example, Global Positioning System [GPS] satellites) and near the poles, one would assume that navigation software would take the shortest path rather than following a rhumb line.

the flat-Earth model, but it must be tipped as it moves so that the vertical axis of its local coordinate system remains aligned with the direction of acceleration due to gravity.[4]

A geodesic curve[5] between two points on a surface is the locally shortest path between those points constrained to the surface. A geodesic curve also represents a curve of zero tangential acceleration as discussed, for example in [60]. Geodesics are studied in the mathematical field of differential geometry, more information on which is available in [15, Sec. 4-4], among many other sources. As discussed in Section II, a target lacking acceleration in the local tangent plane under a constant-velocity flat-Earth model mapped to a curved Earth will follow a geodesic curve.

A common method of describing the curvature or shape of the Earth is in terms of variations in the gravitational field of the Earth. For example, altitudes on maps are usually taken with respect to mean sea level (MSL), which is a hypothetical surface of constant gravitational potential. Geodesy is the study of the shape of the Earth, especially in terms of its gravity potential. A brief overview of geodetic concepts is given in [42], with a more thorough introduction provided in the text [29]. Section IV describes the gravitational shape of the Earth, focusing on a simple ellipsoidal approximation that is used in the development of the flat-Earth-to-curved-Earth mapping algorithm of this paper.

An ellipse of revolution is the three-dimensional surface formed by rotating an ellipse around its semi-major axis . The shape of the Earth is approximately an ellipse of revolution formed by rotating an ellipse about its minor axis. This type of an ellipsoidal is often called an oblate spheroid.[6] On the other hand, an ellipse of rotation formed by rotating an ellipse about its major axis is known as a prolate spheroid. The literature often refers to an approximation of the Earth's shape using an ellipse of revolution as an "ellipsoidal approximation." For the purposes of navigation, a lot of work has been done studying geodesic curves using an ellipsoidal approximation of the Earth. Though navigation literature often uses circular or elliptical curves to approximate geodesic paths on an ellipsoidal Earth, geodesic curves are generally neither elliptical nor circular as is discussed in [79], and this paper does not use circular or elliptical approximations when mapping flat-Earth trajectories to a curved Earth.

Navigators often study the direct and indirect geodetic problems. In the direct geodetic problem, one must determine the ending point of a trajectory when given a starting point, an initial bearing, and a distance. Though the primary focus of this paper is the adaptation of flat-Earth motion models to a curved Earth, the algorithm in this paper can also be used with a non-maneuvering motion model to *solve the direct geodetic problem*, providing Cartesian coordinates of points along the trajectory. Unlike existing algorithms for solving the direct geodetic problem, which are constrained to the surface of the Earth, the algorithm of this paper can solve the problem for arbitrary constant altitudes above the Earth. The algorithm of this paper *does not solve the indirect geodetic problem*. However, algorithms that solve the indirect geodetic problem can be used as a basis for confirming the validity of the direct geodetic solutions obtained by the algorithm in this paper.

In the indirect geodetic problem, one is tasked with finding the shortest path between two points on the surface of the Earth. Algorithms that solve the indirect geodetic problem, which is to determine the initial heading and distance traveled to go from one point (a) to another point (b) along the shortest geodesic path, are commonly used with an ellipsoidal Earth approximation in navigation applications. Since non-maneuvering, level, flat-Earth motion models follow geodesic curves (solving the direct geodetic problem), the validity of the algorithm developed in this paper can be verified for level flight using the solution to the indirect geodetic problem. That is, an initial target state can have position components corresponding to point (a) and an initial velocity vector that points in the direction of the solution to the indirect geodetic problem from (a) to (b). If the model is propagated forward in time such that it covers a distance equal to the distance between (a) and (b) found when solving the indirect geodetic problem, then the target should be located at point (b). Any deviation from point (b) would indicate an error in the target propagation algorithm or numerical precision limitations.

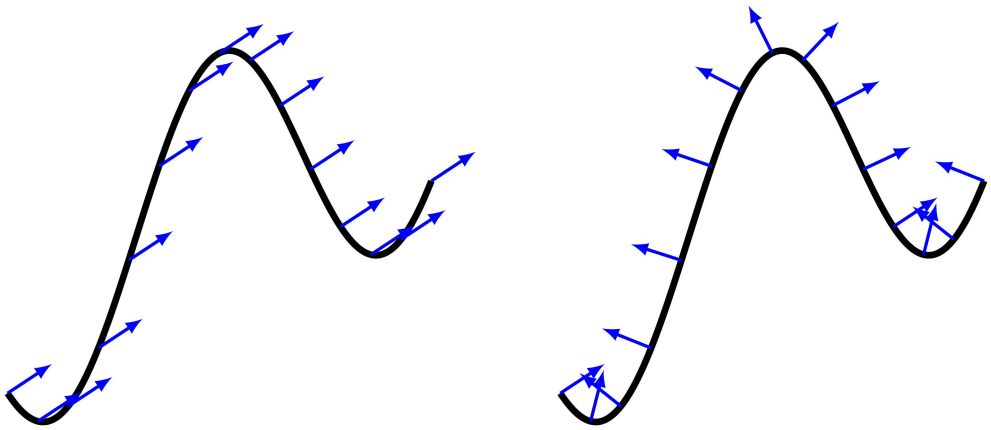
A detailed look at a number of traditional methods for solving the direct and indirect problems is given in [61], including a derivation of Vincenty's equation in [12], which is one of the most commonly used techniques, but for which a derivation is lacking in Vincenty's paper [77]. With modern computers, differential equation solvers can be used to solve both the direct and indirect problems of geodesics [38], [57], [64], [74], and accurate iterative techniques are also available [34], [35].

The majority of the algorithms in the literature for solving the indirect geodetic problem are not suitable for use in any mission-critical navigation system, because they will fail if two points lie nearly on the same meridian (have the same longitude or longitudes approximately 180° apart). However, one robust algorithm

[4]In some instances, not considered in this paper, one might also wish to account for the Coriolis force. The Coriolis force is a small, velocity-dependent force to which a moving object on the surface of the Earth appears to be subjected when viewed from an Earth-fixed coordinate system (a derivation is given in [69, Ch. 7]).

[5]A geodesic curve on a sphere or an ellipsoid is sometimes referred to as an orthodrome (in contrast to a loxodrome). However, the term orthodrome is more frequently applied to geodesic curves on the surface of spheres rather than general ellipsoids.

[6]A more generally shaped ellipsoid is known as a triaxial ellipsoid.



(a) Parallel Transport (b) Darboux Frame

Fig. 1. In (a), a vector is parallel transported along a path in a plane. At all points along the path, the vector remains parallel to the original vector. On the other hand, a vector in the Darboux frame in a plane rotates with the changing direction of the curve as shown in (b). Thus, a vector that is initially perpendicular to the curve remains perpendicular to the curve.

is given in [34], [35]. This algorithm is convenient to use both because it is well documented and because code implementing it is available for Matlab at [36] and for C, Fortran, Java, and a number of other programming languages at [37]. The algorithm provides an initial heading to reach the destination as well as the geodesic distance between the points. However, the geodesic curves used in the simulation section (for validating the solution to the direct geodetic problem provided by the algorithm of this paper) are calculated using the algorithm of [57]. Although the algorithm of [57] is not suitable for mission-critical navigation, it provides the trajectory in a format that can be easily plotted.

The examples in this paper focus on the use of an ellipsoidal Earth approximation, because it is simple and because higher-order gravitational approximations would not be expected to provide more accurate predictions, as air pressure variations begin to dominate at higher precisions. A commercial aircraft that flies at a constant instrument "altitude" reading varies its true height above the ground and above mean sea level, because most commercial aircraft operate in regions where their altitudes are computed based on a reference air pressure and not on current meteorological conditions.[7] As a result, a plane that is flying at a constant altitude according to its own instruments does not fly a precise geodesic curve.

Using the data offered by [52], the WGS-84 geoid, a higher-order approximation than the reference ellipsoid for gravity at sea level, differs from the reference ellipsoid up to +84 m and −107 m, which is less than one flight level used by air traffic control in the United States [21].[8] The geoid undulations are less than altitude variations one would expect to see in an aircraft due to changing meteorological conditions, which are quantified in Appendix A.

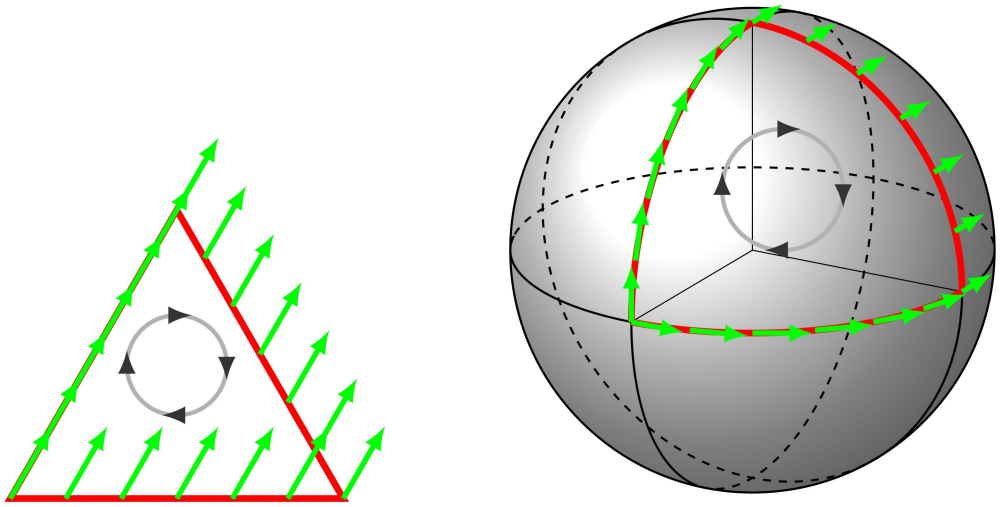
Section II describes the concept of parallel transport from differential geometry as well as its realization through a naturally evolving coordinate system. The naturally evolving coordinate system allows one to make as few changes as possible to a flat-Earth dynamic model to adapt it to a curved Earth. Section III then discusses the adaptation of flat-Earth dynamic models to a curved Earth. Section IV describes the ellipsoidal approximation to the shape of the Earth, and a full algorithm for the ellipsoidal approximation is given in Subsection IV-D. The validity of the algorithm is demonstrated through simulation in Section V, and the results are summarized in Section VI.

II. PARALLEL TRANSPORT ON A SURFACE IN 3D AND NATURALLY EVOLVING COORDINATES

On a flat surface, an individual walking along a curved path generally perceives oneself as moving and turning, not that one is still while the rest of the world moves and rotates. The perception of self-motion means that, as illustrated in Fig. 1(a), a direction vector in the local coordinate system that one carries will not rotate as one moves. That is, while the vector might move with the observer, for example representing an axis in a moving local coordinate system, the vector at one time is always parallel to the vector at a future time. It is "parallel transported" along the line of motion. Such an evolution of vectors in the plane can be contrasted, for example, with the coordinate system given by the Frenet formulas [15, Ch. 1–5] (the Frenet frame), whereby the direction of the local coordinate system evolves such

[7] In the United States, aircraft below 18,000 ft (5,486.4 m) determine their altitudes based on their height determined from the local pressure and the current pressure reading on the ground, translating into a "true" altitude; whereas aircraft operating above 18,000 ft assume the presence of a standard atmosphere, so their true altitude varies with local pressure variations [21]. Similar rules apply in other countries. For example, in Britain the switch to a standard atmospheric assumption occurs at 3,000 ft.

[8] Aircraft in controlled airspace are allocated flight levels that are 500 ft (≈ 152 m) apart.



(a) Parallel Transport Around a Flat Triangle

(b) Parallel Transport Around a Triangle on a Sphere

Fig. 2. In (a) and (b), a vector is parallel transported around a triangle in the direction of the arrows on the circle (clockwise) starting from the lower left corner of the triangle. In (a), the triangle is in a plane and the vector has the same orientation at the beginning of the loop as at the end. In (b), the vector is pointed in a different direction (in the local tangent plane) after having completed the loop.

that one axis always points in the direction along a curve of motion. In the Frenet frame, the evolution of the coordinate system along a curve is determined by the curvature of a curve, not of the surface of which the curve might be located. The Darboux frame, on the other hand, can be viewed as a modified Frenet frame for a curve on a surface, where one axis of the local coordinate system follows a curve along the surface and another axis remains normal to the surface [15, Ch. 4-4]. Figure 1(b) shows an example of how a vector would evolve under the Darboux frame on a flat surface.

The notion of parallel transporting a coordinate system along a curve has been considered in [5], [16], [28], where in [16] tips for numerically implementing such a system are provided. The parallel transport frame along a curve has even been used to describe the local coordinate system of an unmanned aerial vehicle (UAV) [26]. On the other hand, differential geometry for describing motion on a curved surface has been applied to geodetic problems [7]. However, the work of [7] only considers the problem of simulating targets constrained to a surface, and does not specifically consider the use of arbitrary dynamic models.

The notion of parallel transport can be extended to curved surfaces, just as the Darboux frame is a curved-surface equivalent to the Frenet frame. Parallel transport on a curved surface arises naturally as one would expect that the local coordinate system associated with a person walking along an arbitrary path on the surface of the Earth would experience no acceleration in a plane tangential to the surface of the Earth; the world does not "rotate" around the person, but the direction of the "up" vector always stays the same. Figure 2 illustrates how parallel transport on a surface (in this case, a sphere) depends both on the direction traveled as well as on the curvature of the surface. In Fig. 2(a), a vector is shown parallel transported clockwise around a triangle, remaining parallel at all times; whereas in Fig. 2(b), a vector is parallel transported counterclockwise around a triangle on a sphere, returning to the starting point at a different orientation. The topic of parallel transport is discussed in the following subsection.

A geodesic curve is the locally shortest distance between two points on a surface.[9] When the vector being parallel transported with a local observer is the observer's own velocity vector and the observer follows a trajectory of constant velocity, the observer follows a geodesic curve. The concepts of parallel transport and geodesics are covered formally in detail in [15, Sec. 4-4] and [25, Ch. 8].

Subsection II-A covers the topic of 2D motion on a 3D surface in a differential geometric framework. The evolution of the local coordinates of an observer on a surface is shown to depend on the parameterization of the surface. The concept of a naturally evolving coordinate system is introduced in Subsection II-B as a means of describing the evolution of a local coordinate system without the need for surface-dependent acceleration terms in the local observer's coordinate system.

A. Parallel Transport on a Surface in 3D

Let $\mathbf{r}$ be a vector in a global orthogonal (Cartesian) coordinate system (such as Earth-centered, Earth-fixed) from the origin to a point on the surface under consideration. In this paper, the point is on the surface of

[9]The distance is locally shortest, because no minor variations to the path will make it shorter. However, it is not necessarily the globally shortest path. For example, two possible geodesic curves can go from New York to Paris. One goes East and the other goes West. However, the eastward path is the shorter geodesic. When using an optimization algorithm to minimize a distance function to find the shortest path, a geodesic curve is a locally optimal solution.

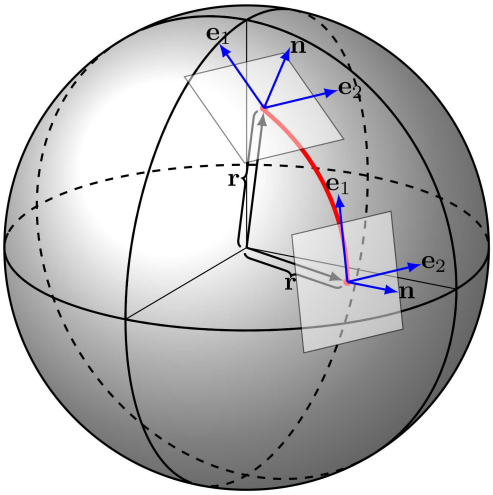


Fig. 3. At a particular point on a surface (specified by a position vector $\mathbf{r}$ at two possible points along a curve), one can have a local coordinate system defined by vectors in the tangent plane, $\mathbf{e}_1$ and $\mathbf{e}_2$, and a normal vector $\mathbf{n}$. When moving along a path on a surface, all of the vectors must change their direction so that the normal to the surface remains pointing "up." Whereas the direction of the normal vector is uniquely defined everywhere, any two orthogonal vectors in the local tangent plane can be used for $\mathbf{e}_1$ and $\mathbf{e}_2$. Parallel transport of the basis vectors, which is the principle behind the naturally evolving coordinate system, means that there is no acceleration in the tangent plane as one moves along the curve; that is, $\mathbf{e}_1$ and $\mathbf{e}_2$ must rotate to remain in the local tangent plane, but none of the rotation is about the normal vector (there is no acceleration in the local tangent plane).

the Earth. Assume that the surface can be parameterized using N_d parameters denoted by $x_1,\ldots,x_{N_d}$. For example, if the shape is the reference ellipsoid for the Earth, N_d is two, even though $\mathbf{r}$ is a three-dimensional position vector, because any point on an ellipsoid can be expressed in terms of geodetic latitude ϕ and longitude λ. However, such a choice of parameterization is not unique.

A vector $\mathbf{v}$ in the tangent plane to the surface of a shape can be expressed as a weighted sum of basis vectors in the tangent plane to the surface. Tangent vectors are given by derivatives of $\mathbf{r}$ with respect to the parameterization of the surface. Specifically,

$$\mathbf{v} = \sum_{i=1}^{N_d} v_i^s \overbrace{\frac{\partial \mathbf{r}}{\partial x_i}}^{\mathbf{e}_i}, \tag{2}$$

where the v_i^s are not the elements of $\mathbf{v}$, (which represents a 3D Cartesian vector), but instead represent weightings in the tangent space of the surface at a particular point. They can be considered the "local" coordinates of $\mathbf{v}$ as seen by an observer on the surface. The ith tangential basis vector is $\mathbf{e}_i$. Figure 3 illustrates the position vector and local basis vectors when considering a trajectory on the surface of a sphere. The concern of this subsection is the mathematical description of how vectors in the local tangent plane evolve over time as one moves on a surface.

Since $\mathbf{v}$ is constrained to a tangent plane to the Earth's surface (for example, it could be a heading), the 3D Cartesian direction of $\mathbf{v}$ must depend on position. The total derivative of $\mathbf{v}$ with respect to a coordinate x_j parameterizing the location on the surface is

$$\frac{d\mathbf{v}}{dx_j} = \underbrace{\sum_{i=1}^{N_d} \frac{\partial v_i^s}{\partial x_j}\mathbf{e}_i}_{\substack{\text{Change of}\\ \text{Components}}} + \underbrace{\sum_{i=1}^{N_d} v_i^s \overbrace{\frac{\partial^2 \mathbf{r}}{\partial x_j \partial x_i}}^{\partial \mathbf{e}_i/\partial x_j}}_{\substack{\text{Change of}\\ \text{Basis Vectors}}}. \tag{3}$$

In general, the second term in (3) does not need to be limited to the tangent plane and consequently can be written as

$$\frac{\partial \mathbf{e}_i}{\partial x_j} = \sum_{k=1}^{N_d} \Gamma_{i,j}^k \mathbf{e}_k + \mathbf{n}_{ij}, \tag{4}$$

where $\Gamma_{i,j}^k$ is a Christoffel symbol of the second kind, a projection (inner product) of $\partial^2\mathbf{r}/\partial x_j \partial x_i$ onto the basis vector $\partial \mathbf{r}/\partial x_k$,

$$\Gamma_{i,j}^k = \frac{\partial \mathbf{e}_i}{\partial x_j} \cdot \left(\frac{\mathbf{e}_k}{\|\mathbf{e}_k\|} \right), \tag{5}$$

where the dot indicates an inner product and $\mathbf{n}_{ij}$ consists of any second derivative components that do not lie in the tangent plane. The two vertical lines denote the magnitude of the enclosed vector (the l_2-norm). Note that $\Gamma_{ij}^k = \Gamma_{ji}^k$.

As long as $\mathbf{v}$ remains on the surface, it is always confined to the tangent plane, suggesting that derivative components that cannot be expressed in terms of tangential basis vectors are not necessarily essential to defining the evolution of the coefficients on the surface. The covariant derivative is simply the total derivative with the components $\mathbf{n}_{ij}$, which are orthogonal to the tangent plane, discarded. The covariant derivative is thus

$$\nabla_{x_j} \mathbf{v} = \sum_{i=1}^{N_d} \frac{\partial v_i^s}{\partial x_j}\mathbf{e}_i + \sum_{i=1}^{N_d}\sum_{k=1}^{N_d} v_i^s \Gamma_{i,j}^k \mathbf{e}_k. \tag{6}$$

A zero covariant derivative implies zero acceleration in the local tangent plane. To use a flat-Earth dynamic model on a curved Earth, zero tangential acceleration means that no extra terms need to be added to the flat-Earth model in adapting it to the curved Earth; only the basis vectors change.

The notion of parallel transport arises when considering how a vector changes direction as an object moves along a curve on the surface. If the instantaneous velocity vector $\mathbf{v}$ of a non-maneuvering object is parallel transported along the surface in the direction of $\mathbf{v}$, then the object will follow a geodesic curve. In general, however, $\mathbf{v}$ can be any vector (such as acceleration) in the local coordinate system of the target moving on an arbitrary path on the surface. Consequently, the concept of parallel transport shall be described for how $\mathbf{v}$ evolves as an object traverses an arbitrary curve.

Let the curve on the surface that the target follows be parameterized by t, which could be time. Consequently, the x-coordinates parameterizing $\mathbf{r}$ on the surface are functions of t. The covariant derivative of $\mathbf{v}$ with respect to t can be evaluated using the chain rule:

$$\nabla_t \mathbf{v} = \sum_{j=1}^{N_d} \nabla_{x_j} \mathbf{v} \frac{dx_j}{dt} \tag{7a}$$

$$= \sum_{j=1}^{N_d}\sum_{i=1}^{N_d} \frac{\partial v_i^s}{\partial x_j}\mathbf{e}_i \frac{dx_j}{dt} + \sum_{j=1}^{N_d}\sum_{i=1}^{N_d}\sum_{k=1}^{N_d} v_i^s \Gamma_{i,j}^k \mathbf{e}_k \frac{dx_j}{dt} \tag{7b}$$

$$= \sum_{i=1}^{N_d} \frac{dv_i^s}{dt}\mathbf{e}_i + \sum_{j=1}^{N_d}\sum_{i=1}^{N_d}\sum_{k=1}^{N_d} v_i^s \frac{\partial \mathbf{e}_i}{\partial x_j} \cdot \left(\frac{\mathbf{e}_k}{\|\mathbf{e}_k\|}\right)\mathbf{e}_k \frac{dx_j}{dt} \tag{7c}$$

$$= \sum_{i=1}^{N_d} \frac{dv_i^s}{dt}\mathbf{e}_i + \sum_{i=1}^{N_d}\sum_{k=1}^{N_d} v_i^s \frac{d\mathbf{e}_i}{dt} \cdot \left(\frac{\mathbf{e}_k}{\|\mathbf{e}_k\|}\right)\mathbf{e}_k. \tag{7d}$$

Parallel transport occurs when the v_i terms evolve such that $\nabla_t \mathbf{v} = \mathbf{0}$. If t is time, then the terms dx_j/dt are related to the velocity of the observer in the observer's local coordinate system.

The elements of $\nabla_t \mathbf{v}$ are in a global Cartesian coordinate system and are thus independent of how the x terms parameterize $\mathbf{r}$ to obtain the tangential basis vectors $\mathbf{e}$. For the purpose of this discussion, a parameterization of $\mathbf{r}$ is chosen such that the basis vectors in the tangent plane are always orthonormal. It is not necessary to know how to obtain such a set of basis vectors practically, only to know that they exist and could be used. That the magnitudes of the basis vectors remain constant implies that the derivative of a basis vector with respect to t must be orthogonal to the basis vector. The time evolution of the basis vectors as a function of t must be a rotation lest the magnitudes of the vectors change, and all basis vectors must experience the same rotation to remain orthogonal.[10] A 3D rotational dynamic model allows the derivative of a basis vector to be written as

$$\frac{d\mathbf{e}_i}{dt} = \Omega \times \mathbf{e}_i, \tag{8}$$

where $\times$ denotes the cross product, the direction of Ω is the axis of rotation, and the magnitude of Ω specifies the rotation rate. (How Ω is determined for a particular surface is discussed in Subsection II-B.) Consequently, when considering a 3D space, if $\nabla_t \mathbf{v} = \mathbf{0}_{3,1}$, where $\mathbf{0}_{3,1}$ is a 3×1 vector of zeros, (7d) becomes

$$\mathbf{0}_{3,1} = \sum_{i=1}^{N_d} \frac{dv_i^s}{dt}\mathbf{e}_i + \sum_{i=1}^{N_d}\sum_{k=1}^{N_d} v_i^s(\mathbf{e}_k \cdot (\Omega \times \mathbf{e}_i))\mathbf{e}_k \tag{9a}$$

$$= \sum_{i=1}^{N_d} \frac{dv_i^s}{dt}\mathbf{e}_i + \sum_{i=1}^{N_d}\sum_{k=1}^{N_d} v_i^s(\Omega \cdot (\mathbf{e}_i \times \mathbf{e}_k))\mathbf{e}_k, \tag{9b}$$

where the vector triple product identity $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \mathbf{b} \cdot (\mathbf{c} \times \mathbf{a})$ is used. Rearranging terms yields

$$\sum_{i=1}^{N_d} \frac{dv_i^s}{dt}\mathbf{e}_i = -[\mathbf{e}_1 \dots \mathbf{e}_{N_d}] \begin{bmatrix} \sum_{i=1}^{N_d} v_i^s((\mathbf{e}_i \times \mathbf{e}_1) \cdot \Omega) \\ \vdots \\ \sum_{i=1}^{N_d} v_i^s((\mathbf{e}_i \times \mathbf{e}_{N_d}) \cdot \Omega) \end{bmatrix}. \tag{10}$$

In three dimensions, two tangent vectors are sufficient to parameterize a surface ($N_d = 2$). On applying the identity $\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$, (10) reduces to

$$\mathbf{e}_1 \frac{dv_1^s}{dt} + \mathbf{e}_2 \frac{dv_2^s}{dt} = -[\mathbf{e}_1 \quad \mathbf{e}_2] \begin{bmatrix} -v_2^s(\mathbf{e}_1 \times \mathbf{e}_2) \cdot \Omega \\ v_1^s(\mathbf{e}_1 \times \mathbf{e}_2) \cdot \Omega \end{bmatrix} \tag{11a}$$

$$= -(\Omega - (\Omega \cdot \mathbf{e}_i)\mathbf{e}_i - (\Omega \cdot \mathbf{e}_j)\mathbf{e}_j) \times \mathbf{v}. \tag{11b}$$

When assuming that $\mathbf{e}_1$ and $\mathbf{e}_2$ are mutually orthonormal, the components of $d\mathbf{v}^s/dt$ can be extracted using dot products with the basis vectors to get

$$\frac{dv_1^s}{dt} = -(\Omega \times \mathbf{v}) \cdot \mathbf{e}_1 = v_2 \Omega \cdot (\mathbf{e}_1 \times \mathbf{e}_2), \tag{12}$$

$$\frac{dv_2^s}{dt} = -(\Omega \times \mathbf{v}) \cdot \mathbf{e}_2 = -v_1 \Omega \cdot (\mathbf{e}_1 \times \mathbf{e}_2). \tag{13}$$

That is, the local coordinates change according to the projection of the cross product of the axis of rotation of the basis vectors onto the local coordinates. The magnitude of the local coordinates is preserved through this operation, which can be demonstrated by writing

$$\begin{bmatrix} \frac{dv_1^s}{dt} \\ \frac{dv_2^s}{dt} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\Omega \cdot (\mathbf{e}_1 \times \mathbf{e}_2) \end{bmatrix} \times \begin{bmatrix} v_1^s \\ v_2^s \\ 0 \end{bmatrix}, \tag{14}$$

where, due to the cross product, it can be seen that the derivative is orthogonal to the component vector. Since the basis vectors are assumed to maintain unit magnitude, the vector $\mathbf{v}$ maintains a constant magnitude through parallel transport.

After substituting the derivatives of the local coordinates in (12) and (13) into the total derivative in (3), the evolution in the direction of $\mathbf{v}$ over time is

$$\frac{d\mathbf{v}}{dt} = -\sum_{i=1}^{2}((\Omega \times \mathbf{v}) \cdot \mathbf{e}_i)\mathbf{e}_i + \sum_{i=1}^{2} v_i^s \Omega \times \mathbf{e}_i \tag{15}$$

$$= \underbrace{\Omega \times \mathbf{v}}_{\substack{\text{General} \\ \text{3D Rotation}}} + \underbrace{v_2 \Omega \cdot (\mathbf{e}_1 \times \mathbf{e}_2)\mathbf{e}_1 - v_1 \Omega \cdot (\mathbf{e}_1 \times \mathbf{e}_2)\mathbf{e}_2}_{\text{Undo Rotation in Tangent Plane}}. \tag{16}$$

[10] The ability to express the evolution of the basis vectors as a continuous rotation assumes that there is no bifurcative changes in the basis vectors that redefine the relative relationship between the vectors. For example, suddenly changing from a right-handed coordinate system to a left-handed coordinate system would be a bifucative change in the relationship between the vectors. A bifurcation would also arise when crossing the North Pole using normalized ENU coordinates.

Thus, as $\mathbf{v}$ parallel transports along the surface, the change in the direction of $\mathbf{v}$ depends only on how the normal vector to the tangent plane changes, not on how the bases are rotated within the tangent plane. If the orientation of the tangent plane does not change, any rotation of the bases does not matter; $\mathbf{v}$ does not change. The irrelevance of rotations outside of the local tangent plane means that the rotation vector Ω to describe the evolution of the orientation of the tangent plane is not unique.

B. Naturally Evolving Coordinates

If the vector $\mathbf{v}$ being parallel transported in (16) is, for example, a velocity vector, then the local coordinates must evolve according to (12) and (13). However, forcing the local coordinates to evolve depending on the motion on the surface can be inconvenient if one wishes to use more than a straight-line dynamic model. It is more convenient if the local coordinates never changes as the target moves.

The vector $\mathbf{v}$ does not need to be a velocity vector. For example, it could represent an axis (a basis vector) in the coordinate system of the local observer moving on the surface. If motion along the surface does not induce any change in the components of $\mathbf{v}$ (any basis vector) in the *local* coordinate system of the target, v_1 and v_2, then the target does experience any local acceleration beyond that induced by the flat-Earth dynamic model. Examining (14), v_1 and v_2 remain constant if the axis of rotation is orthogonal to the local normal vector (if there is no rotation in the local tangent plane). When the basis vectors parameterizing $\mathbf{r}$, $\mathbf{e}_1$, and $\mathbf{e}_2$, are chosen to have zero rotation in the local tangent plane, then (16) implies that

$$\frac{d\mathbf{v}}{dt} = \Omega \times \mathbf{v}. \tag{17}$$

The elimination of the extra terms in (16) to obtain (17) means that the basis vectors in which the coordinates of $\mathbf{r}$ are expressed depend on the path taken by the object. Such a path-dependent local coordinate system is defined to be "naturally evolving coordinates." The set of local tangent-plane basis vectors for natural coordinates are designated as $\mathbf{u}_1$ and $\mathbf{u}_2$ to differentiate them from the vectors parameterization of the surface that have thus been expressed as $\mathbf{e}_k$.

Let $\mathbf{g}$ be a vector orthogonal to the local tangent plane that defines the local "downward" direction on the two-dimensional surface. The down vector $\mathbf{g}$ is not necessarily a unit vector and could be acceleration due to gravity. To define a right-handed local coordinate system in terms of three orthonormal basis vectors, let $\mathbf{u}_3$ be a unit vector defining the local vertical, which is expressed in terms of $\mathbf{g}$ as

$$\mathbf{u}_3 = -\frac{\mathbf{g}}{\|\mathbf{g}\|}. \tag{18}$$

In other words, the unit "up" vector is the negative of the "down" vector divided by its magnitude. To complete a right-handed coordinate system, the remaining orthogonal unit vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ in the local tangent plane must satisfy the defining cross product relations

$$\mathbf{u}_1 \times \mathbf{u}_3 = -\mathbf{u}_2 \qquad \mathbf{u}_1 \times \mathbf{u}_2 = \mathbf{u}_3 \qquad \mathbf{u}_2 \times \mathbf{u}_3 = \mathbf{u}_1. \tag{19}$$

One coordinate system that satisfies all of these relations is a local East-North-Up (ENU) coordinate system. One would like to have vectors that are parallel transported with a moving target naturally evolve. Though an ENU coordinate system can be used to define a target's initial local coordinate system, it does not naturally evolve as a target moves. Natural evolution of a vector is synonymous with parallel transport reducing to (17) and the basis vectors evolving with no rotation in the tangent plane.

Expressions for how the orthonormal basis vectors $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ evolve over time as a target moves shall be derived. The derivative of (18) with respect to time t is

$$\dot{\mathbf{u}}_3 = -\frac{\dot{\mathbf{g}}}{\|\mathbf{g}\|} + \frac{\mathbf{g}}{\|\mathbf{g}\|^3}(\dot{\mathbf{g}} \cdot \mathbf{g}) = \Omega \times \mathbf{u}_3, \tag{20}$$

where the dot indicates differentiation with respect to t. The simplification to a cross product expression is possible, because the derivative must be orthogonal to the vector, lest the magnitude of the basis vector change. An infinite number of solutions for the rotation vector Ω exist. For the naturally evolving coordinate system, the vector Ω is constrained to the local tangent plane (so that $\mathbf{u}_3$ has no rotational components in the local tangent plane):

$$\Omega = c_1\mathbf{u}_1 + c_2\mathbf{u}_2. \tag{21}$$

Consequently,

$$\dot{\mathbf{u}}_3 = (c_1\mathbf{u}_1 + c_2\mathbf{u}_2) \times \mathbf{u}_3. \tag{22}$$

Using (19), $\dot{\mathbf{u}}_3$ becomes

$$\dot{\mathbf{u}}_3 = \overbrace{\begin{bmatrix} -\mathbf{u}_2 & \mathbf{u}_1 \end{bmatrix}}^{\mathbf{A}} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}. \tag{23}$$

Equation (23) is an overdetermined system for which a unique solution should exist. Thus, the coefficients c_1 and c_2 can be found using

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \mathbf{A}^\dagger \left(-\frac{\dot{\mathbf{g}}}{\|\mathbf{g}\|} + \frac{\mathbf{g}}{\|\mathbf{g}\|^3}(\dot{\mathbf{g}} \cdot \mathbf{g}) \right), \tag{24}$$

where the $\dagger$ operator represents the matrix pseudoinverse. The matrix pseudoinverse $\mathbf{A}^\dagger$ can be evaluated using `pinv` in Matlab, which is based on a singular value decomposition technique for stability. A similar matrix pseudoinverse algorithm is described in [24, Ch. 5] and [23]. A simpler but less numerically robust alternative is just to use $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$, where, as described in [23], the possible numerical instability arises when the matrix $\mathbf{A}^T\mathbf{A}$ is nearly singular.

To evaluate (24) to get Ω in (21), one must know $\dot{\mathbf{g}}$, which is given by

$$\dot{\mathbf{g}} = \begin{bmatrix} \frac{\partial \mathbf{g}}{\partial r_1} & \frac{\partial \mathbf{g}}{\partial r_2} & \frac{\partial \mathbf{g}}{\partial r_3} \end{bmatrix} \dot{\mathbf{r}}, \tag{25}$$

where $\mathbf{r} = [r_1, r_2, r_3]'$ are in global Cartesian coordinates and $'$ denotes the transpose operator. The partial derivatives in (25), which are constrained to the surface, can be written using the chain rule

$$\frac{\partial \mathbf{g}}{\partial r_i} = \sum_{j=1}^{2} \frac{\partial \mathbf{g}}{\partial x_j} \frac{\partial x_j}{\partial r_i}, \tag{26}$$

where the x_j are the terms parameterizing $\mathbf{r}$ on the surface. However, the evaluation of the partial derivatives in (26) can be problematic, because the parameterization of the surface can have singularities. For example, when using ENU coordinates, singularities exist at the poles, because longitude does not have any meaning at the poles. Such singularities can be avoided either by obtaining a closed-form solution for the product of the partial derivatives, which is difficult, or by using a numerical differentiation algorithm. Section IV-C describes how, following the technique of [78], an explicit solution for Ω can be found when using an ellipsoidal Earth model.

In summary, the naturally evolving local coordinate system of a target moving on a surface is defined as an orthonormal coordinate system at an initial point on the surface with axes given by the unit vectors $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$. For example, one might use the ENU axes described in Section IV when on an ellipsoidal Earth. Then, the axes evolve according to

$$\dot{\mathbf{u}}_i = \Omega \times \mathbf{u}_i,, \tag{27}$$

where Ω is given by (21) and (24) (or for an ellipsoidal Earth model from Section IV-C), which depend on the derivative of the local "down vector" $\mathbf{g}$ in (25). Each component of the partial derivatives can be evaluated using (26), which must be simplified to avoid singularities, or using numerical differentiation to avoid singularities. Techniques for numerical differentiation, besides simple differencing of two points, are derived using Taylor series expansions [6, Ch. 4.1]. As described in [80], finite difference formulae with $2N-1$ points can be automatically generated in Mathematica.

III. APPLYING NATURALLY EVOLVING COORDINATES TO 2D AND 3D DYNAMIC MODELS

Let $\mathbf{v}^{\text{local}} = [v_1^l, v_2^l, v_3^l]^T$ be a vector in the local coordinate system of the target, for example, velocity or acceleration. A vector $\mathbf{v}^{\text{global}}$, which is $\mathbf{v}^{\text{local}}$ as seen in the global observer's coordinate system, can be expressed as

$$\mathbf{v}^{\text{global}} = \sum_{i=1}^{3} v_i^l \mathbf{u}_i. \tag{28}$$

Let $\mathbf{x}_t^g$ and $\mathbf{x}_t^l$ be the state vectors of the target in the global (curved-Earth) and local (flat-Earth) coordinate systems at time t, respectively. A noise-free flat-Earth dynamic model is generally expressed in terms of a differential equation,

$$\dot{\mathbf{x}}_t^l = \mathbf{a}^l(\mathbf{x}_t^l, t), \tag{29}$$

where $\dot{\mathbf{x}}_t^l$ is the derivative of the local state with respect to time and $\mathbf{a}^l$ is a possibly time-variant function of $\mathbf{x}_t^l$ known as the drift.

It is assumed that the local dynamic model under consideration obtains the position component of the model by integrating a velocity component of the state. The rest of the dynamic model is arbitrary, but is assumed to be independent of position. Consequently, $\mathbf{x}_t^l$ and $\mathbf{a}^l$ have the forms

$$\mathbf{x}_t^l = \begin{bmatrix} \mathbf{r}_t^l \\ \dot{\mathbf{r}}_t^l \\ \mathbf{x}_t^{l,\text{rest}} \end{bmatrix} \qquad \mathbf{a}^l(\mathbf{x}_t^l, t) = \begin{bmatrix} \dot{\mathbf{r}}_t^l \\ \mathbf{a}^{l,\text{rest}}(\dot{\mathbf{r}}_t^l, \mathbf{x}_t^{l,\text{rest}}, t) \end{bmatrix}, \tag{30}$$

where $\mathbf{r}_t^l$ is the position of the target in its local coordinate system at time t, $\dot{\mathbf{r}}_t^l$ is the velocity of the target in its local coordinate system, and the vector $\mathbf{x}_t^{l,\text{rest}}$ is an arbitrary collection of additional state elements.

The simplest approach for adapting (29) to a global, curved-Earth model is to use a mixed state, where the flat-Earth position $\mathbf{r}_t^l$ is replaced by the curved-Earth position $\mathbf{r}_t^g$. The mixed state $\mathbf{x}_t^m$ with associated dynamic model is thus

$$\mathbf{x}_t^m = \begin{bmatrix} \mathbf{r}_t^g \\ \dot{\mathbf{r}}_t^l \\ \mathbf{x}_t^{l,\text{rest}} \end{bmatrix} \qquad \dot{\mathbf{x}}_t^m = \mathbf{a}^m(\mathbf{x}_t^m, t), \tag{31}$$

where

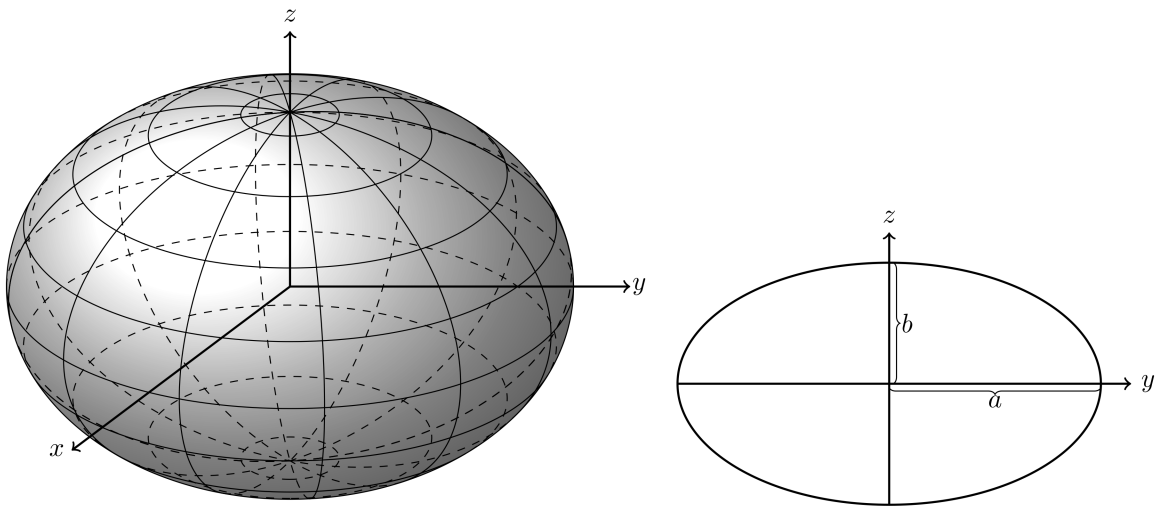
$$\mathbf{a}^m(\mathbf{x}_t^m, t) = \begin{bmatrix} \dot{\mathbf{r}}_t^g \\ \mathbf{a}^{l,\text{rest}}(\dot{\mathbf{r}}_t^l, \mathbf{x}_t^{l,\text{rest}}, t) \end{bmatrix}. \tag{32}$$

Since the global velocity can be linked to the local velocity using (28), (32) can be rewritten as

$$\mathbf{a}^m(\mathbf{x}_t^m, t) = \begin{bmatrix} [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{u}_3]\dot{\mathbf{r}}_t^l \\ \mathbf{a}^{l,\text{rest}}(\dot{\mathbf{r}}_t^l, \mathbf{x}_t^{l,\text{rest}}, t) \end{bmatrix}. \tag{33}$$

If in the target's local coordinate system the target experiences no vertical motion (the velocity components multiplying $\mathbf{u}_3$ in (33) are always zero), then the target can be viewed as remaining at a constant altitude and, ignoring air pressure, would be expected to follow a surface of constant gravitational potential. Thus, the basis vectors should evolve "naturally" as in (27), because the effects of the surface curvature enter the dynamic equations only through a rotation in the local bases, not by changing the local states, and the target is constrained to the surface.

On the other hand, suppose that $\dot{\mathbf{r}}_t^l$ has a nonzero component in the $\mathbf{u}_3$ direction. The target no longer remains at a constant gravitational potential, since it is



(a) Ellipsoidal Earth Approximation (b) $y-z$ Cut ($x=0$)

Fig. 4. The Earth is approximately shaped like an oblate spheroid; that is an ellipse rotated about the z axis (a type of ellipsoid). The non-spheroidal nature of the Earth's shape has been exaggerated in the diagrams. The horizontal lines in (a) represent lines of latitude; vertical lines represent lines of longitude. Longitude is an angle measured with respect to the x-axis in the xy plane. The DoD's WGS-84 standard [13] and IERS conventions [58], among many other sources, define reference ellipsoids for the Earth. In (b), a is the semi-major axis (equatorial radius), and b is the semi-minor axis (polar radius).

ascending or descending along the local vertical. The local surface of constant gravitational potential should be what defines the local tangent plane. However, $\mathbf{g}$ changes orientation as one moves vertically. The change of direction of $\mathbf{g}$ with motion in the direction of $\mathbf{g}$ is known in geodesy as the "curvature of the plumb line" [29, Ch. 2.3]. Curvature of the plumb line implies that the parallel transport equations of Section II must be rederived to determine the "straightest" path when moving in an arbitrary direction through a field where the local coordinate system rotates everywhere. However, when using the ellipsoidal Earth approximation of Section IV, the plumb line does *not* curve.

In an ENU coordinate system, moving up does not change the direction of what is considered to be up. In this instance, if one uses (27) in Section II-B to predict forward the axes of the local coordinate system, $\mathbf{u}_1$ and $\mathbf{u}_2$ always remain in the local tangent plane, and the projection of a straight-line motion target onto the reference ellipsoid follows a geodesic. Consequently, the suggested technique for handling 3D flat-Earth dynamic models on a curved Earth under an ellipsoidal approximation is to make $\mathbf{g}$ the negative of the normal vector of the closest point on the reference ellipsoid.

IV. THE ELLIPSOIDAL APPROXIMATION SOLUTION

A. The Gravitational Potential and the Reference Ellipsoid

The Earth is approximately oblate spheroidal in shape (an ellipsoidal shape), as illustrated in Fig. 4. However, the shape of the Earth as defined by the geoid, a hypothetical surface of equal gravitational potential defining mean sea level (MSL), is only approximately ellipsoidal.

The global coordinate system used by the U.S. DoD is the WGS-84 standard [13]. The latest version of WGS-84 [51] agrees with the alignment and location of the axes in the ITRF2008, the latest version of the International Terrestrial Reference Frame (ITRF). The ITRF is set by the International Earth Rotation and Reference Systems Service (IERS) [13], [51] and is described in the IERS conventions in [58]. Parameters for the ITRF can be downloaded from [67]. The x-axis of the ITRF points in the direction of the IERS reference meridian, which is the same as the prime meridian, zero degrees longitude, which passes through Greenwich, England. The z-axis points in the direction of the IERS reference pole, which is the geographic North Pole, and is approximately the rotational axis of the Earth. Differences between WGS-84 and ITRF arise from differences in points of reference used to implement the standards, as discussed in [29, Ch. 2.11], and are due to the WGS-84 standard using a different semi-major radius for the reference ellipsoid.

The WGS-84 standard defines a reference ellipsoid for the shape of the Earth, which is derived from parameters for the Earth's gravitational field and is a low-order approximation of the geoid. Table II lists the parameters for the reference ellipsoid, and Fig. 4(b) shows how the parameters relate to the ellipse of revolution. The flattening factor is related to the semi-major (a) and semi-minor (b) axes through the equation

$$f = \frac{a-b}{a}. \tag{34}$$

The reference ellipsoid can be directly related to an approximation of the gravitational potential including the effects of the Earth's rotation, ignoring the direction-dependent Coriolis effect (see [29, Ch. 2.7]).

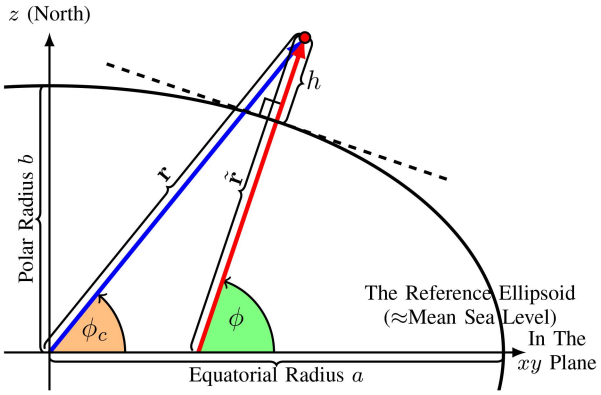


Fig. 5. An illustration of how latitude is measured for a point an altitude h above the reference ellipsoid. Unless specified otherwise, the term latitude generally means the geodetic latitude ϕ, which is taken with respect to a normal to the reference ellipsoid. On the other hand, the geocentric latitude ϕ_c, which is seldom used, is the angle measured with respect to the origin. The horizontal axis is anywhere in the xy plane, which comes out of the page in the diagram.

TABLE II
Defining parameters of the reference ellipsoid in the WGS-84 standard [13]. The presence of GM and ω allow one to reconstruct the gravitational potential of the equipotenial surface implied by the ellipsoid, as derived in [29, Ch. 2.7].

Quantity	Symbol	Value
Gravitational Constant times Earth's Mass (with atmosphere)	GM	$3.986004418 \times 10^{14} (\mathrm{m}^3/\mathrm{s}^2)$
Earth's Semi-Major Axis	a	6378137.0 m
Earth's Inverse Flattening Factor	$1/f$	298.257223563
Nominal Angular Velocity of Earth's Rotation	ω	$7.2921150 \times 10^{-5} (\mathrm{rad/s})$

B. Coordinates on the Reference Ellipsoid

Geodetic and geocentric latitudes are angles measured with respect to the equatorial plane. The geocentric latitude ϕ_c of a point is the angle formed between the equatorial plane and a line from the center of the Earth to the point. The geodetic latitude ϕ is the angle formed between the equatorial plane and a normal to the reference ellipsoid that passes through the point. Figure 5 illustrates the difference between geocentric and geodetic latitudes. Unless otherwise specified, maps generally show geodetic latitude. On the other hand, if one draws a line from the coordinate origin to a point and projects the line onto the xy (equatorial) plane, the angle the line forms with the x-axis (the prime meridian) is the longitude λ. Geocentric latitude and longitude are equivalent to elevation and azimuth in a spherical coordinate system.

When using the ellipsoidal gravitational model, the normal to the reference ellipsoid coincides with the vertical direction defined by the negative of the acceleration due to gravity. That is, "up" is in the direction of $\tilde{\mathbf{r}}$ in Fig. 5. Latitude, longitude, and altitude (height) form an orthogonal set of coordinates. Thus, a set of basis vectors derived from local ellipsoidal coordinates can be used to define $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ initially, but not over time. Rather, the basis vectors must evolve as discussed in Section II-B.

A target's position is given in Earth-Centered, Earth-Fixed (ECEF) coordinates in terms of a vector taken from the center of the Earth, such as $\mathbf{r}$ in Fig. 5. Such a position can be converted into geodetic latitude, longitude, and height coordinates (ϕ, λ, h) using the formulae of [66], which are summarized in Appendix B. Figure 5 shows a point above the reference ellipsoid with its height h measured with respect to the normal to the reference ellipsoid. The conversion of a point given in geodetic latitude, longitude, and height coordinates (ϕ, λ, h) to ECEF Cartesian coordinates can be done as

$$x = (N_e + h)\cos(\phi)\cos(\lambda) \tag{35}$$

$$y = (N_e + h)\cos(\phi)\sin(\lambda) \tag{36}$$

$$z = (N_e(1 - e^2) + h)\sin(\phi) \tag{37}$$

$$N_e = \frac{a}{\sqrt{1 - e^2(\sin(\phi))^2}}, \tag{38}$$

where the first numerical eccentricity of the ellipsoid is e,

$$e \triangleq \sqrt{2f - f^2}, \tag{39}$$

and N_e is the normal radius of curvature [29, Ch. 5.6], also known as the radius of curvature in the prime vertical.

A set of local orthogonal basis vectors obtained by differentiating (35), (36), and (37) with respect to ϕ, λ, and h are

$$\frac{d\mathbf{r}}{d\lambda} = c_1 \mathbf{u}_1 = \begin{bmatrix} -(N_e + h)\cos(\phi)\sin(\lambda) \\ (N_e + h)\cos(\phi)\cos(\lambda) \\ 0 \end{bmatrix} \tag{40}$$

$$\frac{d\mathbf{r}}{d\phi} = c_2 \mathbf{u}_2 = \begin{bmatrix} \left(\cos(\phi)\frac{dN_e}{d\phi} - (N_e + h)\sin(\phi)\right)\cos(\lambda) \\ \left(\cos(\phi)\frac{dN_e}{d\phi} - (N_e + h)\sin(\phi)\right)\sin(\lambda) \\ (N_e(1 - e^2) + h)\cos(\phi) + (1 - e^2)\frac{dN_e}{d\phi}\sin(\phi) \end{bmatrix} \tag{41}$$

$$\frac{d\mathbf{r}}{dh} = \mathbf{u}_3 = \begin{bmatrix} \cos(\phi)\cos(\lambda) \\ \cos(\phi)\sin(\lambda) \\ \sin(\phi) \end{bmatrix}, \tag{42}$$

where

$$\frac{dN_e}{d\phi} = \frac{ae^2\cos(\phi)\sin(\phi)}{(1 - e^2(\sin(\phi))^2)^{3/2}}. \tag{43}$$

The constants c_1 and c_2 in (41) and (40) represent the fact that the vectors do not have unit magnitudes. It can be verified that the vectors form an orthogonal basis. This basis is ENU, because $d\mathbf{r}/dh$ points up, $d\mathbf{r}/d\phi$

points in the direction of geographic North along the surface of the ellipse, and $d\mathbf{r}/d\lambda$ points East along the surface of the ellipse. The ENU coordinate system refers to the normalized forms of the bases in (40) through (42).

Since (40) has zero magnitude at the poles, but (41) and (42) have nonzero magnitudes, $\mathbf{u}_1$ is defined by

$$\mathbf{u}_1 = \mathbf{u}_2 \times \mathbf{u}_3 \tag{44}$$

to guarantee that it always exists. The longitude λ determines the orientation of the North and East axes at the poles.

C. Computing the Tangent Plane Rotation Vector

In the case of an ellipsoidal Earth model, the rotation vector Ω needed to propagate naturally evolving coordinates in Section II-B can be found directly as is done as an intermediate step in propagating wander coordinates in [50, Pgs. 54–56]. The instantaneous turn rate ω of an object that is rotating about an axis is equal to the speed of the object divided by the radius of the object from the axis. Here, we wish to find the instantaneous turn rate of an observer moving above an ellipsoidal Earth. Under an ellipsoidal approximation, the Earth has a different radius of curvature going North than going East. In [61, Sec. 3.5], the different radii of curvature of an ellipsoid are derived in detail. The radius of curvature in the prime vertical, which is the one relevant for North-South motion, is given by N_e in (38). The radius of curvature in the meridian, which is relevant to East-West motion, is given by

$$M_e = N_e \frac{1-e^2}{1-e^2\sin(\phi)^2} \tag{45}$$

Given a global velocity vector of a target expressed in terms of local East, North, and Up coordinates

$$\mathbf{v} = v_{\text{East}}\mathbf{u}_{\text{East}} + v_{\text{North}}\mathbf{u}_{\text{North}} + v_{\text{Up}}\mathbf{u}_{\text{Up}} \tag{46}$$

the instantaneous (right-handed) turn rate about the local East axis (which is needed to correct for North-south motion) is

$$\omega_{\text{East}} = \frac{-v_{\text{North}}}{M_e + h} \tag{47}$$

where h is the height of the target above the reference ellipsoid. Similarly, the instantaneous rotation rate about the local North-axis, which is needed to correct for (East-West motion) is

$$\omega_{\text{North}} = \frac{v_{\text{East}}}{N_e + h} \tag{48}$$

Thus, the total rotation vector needed to propagate naturally evolving coordinates is

$$\Omega = \omega_{\text{East}}\mathbf{u}_{\text{East}} + \omega_{\text{North}}\mathbf{u}_{\text{North}} \tag{49}$$

As the velocity vector $\mathbf{v}$ will be parameterized in terms of the naturally evolving basis vectors

$$\mathbf{v} = v_1^s\mathbf{u}_1 + v_2^s\mathbf{u}_2 + v_3^s\mathbf{u}_3 \tag{50}$$

the local East North and Up components necessary to use (49) can be found using dot products:

$$v_{\text{East}} = \mathbf{v}'\mathbf{u}_{\text{East}} \tag{51}$$

$$v_{\text{North}} = \mathbf{v}'\mathbf{u}_{\text{North}} \tag{52}$$

$$v_{\text{Up}} = \mathbf{v}'\mathbf{u}_{\text{Up}} \tag{53}$$

On the surface of the reference ellipsoid, the naturally evolving coordinate system coincides with the wander coordinate system of navigation.

On the other hand, in Section V, to compare with continuous-time extensions of the algorithms of [43], [81], which rotate a local East-North-Up coordinate system rather than using naturally evolving coordinates. In such a coordinate system, $\mathbf{u}_1 = \mathbf{u}_{\text{East}}$, $\mathbf{u}_2 = \mathbf{u}_{\text{North}}$, and $\mathbf{u}_3 = \mathbf{u}_{\text{Up}}$ and (33) can be used to propagate the state without an explicit differential equation for the basis vectors, because Subsection IV-B provides formulae for the East, North, and Up vectors all over the globe.

D. Constructing the Full Propagation Algorithm

Figure 6 consolidates the concepts in the previous sections to summarize the state propagation algorithm for fitting a flat-Earth dynamic model to a curved Earth under an ellipsoidal gravitational approximation. The Runge-Kutta algorithm, a standard technique for solving deterministic differential equations, is described in [6, Ch. 5.4, 5.5] as well as in the tutorial [9], and an adaptive function is built into Matlab under the name `ode45`.

V. DEMONSTRATING THE ALGORITHMS THROUGH SIMULATION

The validity of the method of adapting a flat-Earth dynamic model to a curved Earth causes the path of a constant-velocity flat-Earth motion model to follow the same geodesic curve that is obtained using established techniques given the same initial heading. Given a target state of $\mathbf{x}_t = [\mathbf{r}_t', \dot{\mathbf{r}}_t']'$, where $\mathbf{r}_t$ is the Cartesian position as a function of time and $\dot{\mathbf{r}}_t$ is the Cartesian velocity vector, the drift term for constant-velocity, flat-Earth state propagation is

$$\mathbf{a}^l(\mathbf{x}_t, t) = \begin{bmatrix} \dot{\mathbf{r}}_t^l \\ \mathbf{0}_{3,1} \end{bmatrix}. \tag{54}$$

Using the notation of Section III, (54) implies that $\mathbf{a}^{l,\text{rest}}(\dot{\mathbf{r}}_t^l, \mathbf{x}_t^{l,\text{rest}}, t) = \mathbf{0}_{3,1}$.

Given two points on the surface of the Earth (height h in ellipsoidal coordinates is zero) whose longitudes neither coincide nor are 180° apart, the geodesic algorithm of [57] provides latitude and longitude pairs along a geodesic curve connecting the points, along with a derivative of latitude with respect to longitude. That derivative can be transformed into an initial heading for a geodesic curve that would connect those points. Such a heading is generally expressed as radians or degrees North of East. North and East define the local tangent

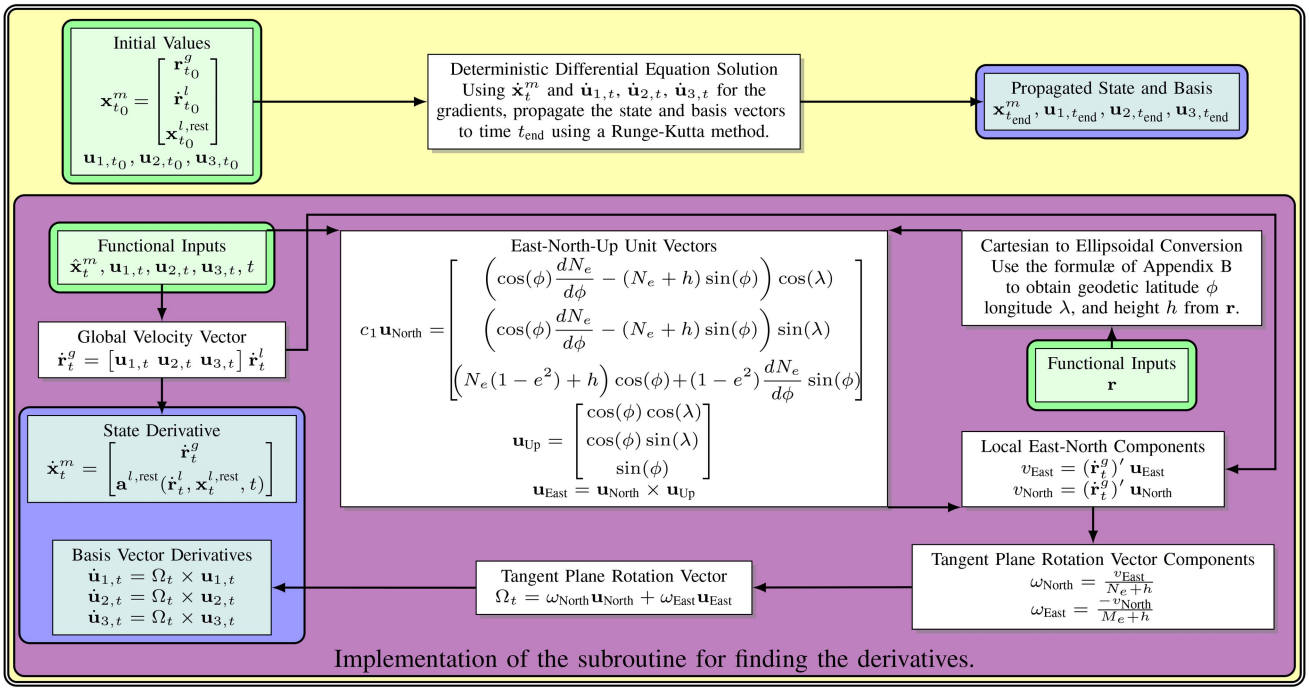


Fig. 6. The steps for predicting forward a flat-Earth dynamic model on a curved Earth when using an ellipse of revolution approximation for the gravitational shape of the Earth. $\mathbf{r}^g_t$ is the target position in the global (curved-Earth) coordinate system; $\dot{\mathbf{r}}^l_t$ is the target velocity in the flat Earth dynamic model and $\mathbf{x}^{l,\text{rest}}_t$ are other state components in the flat-Earth dynamic model. $\mathbf{u}_{\text{North}}$ is obtained by dividing the quantity on the right by its magnitude. $\mathbf{a}^{l,\text{rest}}$ is the function providing derivatives of the velocity and other state components in the flat-Earth dynamic model. The simplest initial set of basis vectors is $\mathbf{u}_{2,t_0} = \frac{d\mathbf{r}}{d\phi} \Big/ \left\| \frac{d\mathbf{r}}{d\phi} \right\|$ (North), $\mathbf{u}_{3,t_0} = \frac{d\mathbf{r}}{dh} \Big/ \left\| \frac{d\mathbf{r}}{dh} \right\|$ (Up), and $\mathbf{u}_{1,t_0} = \mathbf{u}_{2,t_0} \times \mathbf{u}_{3,t_0}$ (East), using the expressions of Section IV-B.

plane in the ENU coordinate system, and an angle North of East is an angle measured from the East axis in the direction of the North axis. By initializing the local coordinate axes $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ at time 0 to be ENU, then if θ is the initial bearing provided by the algorithm of [57], the corresponding initial local (flat-Earth) velocity vector is[11]

$$\dot{\mathbf{r}}^l_t = v \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \end{bmatrix}, \tag{55}$$

where the speed $v = \|\dot{\mathbf{r}}^l_t\|$ can be set arbitrarily and determines how long it takes to travel between the points.

When using the algorithm in Fig. 6, the speed and number of steps needed for the Runge-Kutta algorithm to propagate the state a certain distance are related through the time duration of the step size. If the geodesic distance between the points on the surface of the Earth provided by the algorithm of [57] is s, and one wishes to use N_{steps} of time duration Δ_t for the Runge-Kutta method to integrate from the starting point to the desired ending point, then

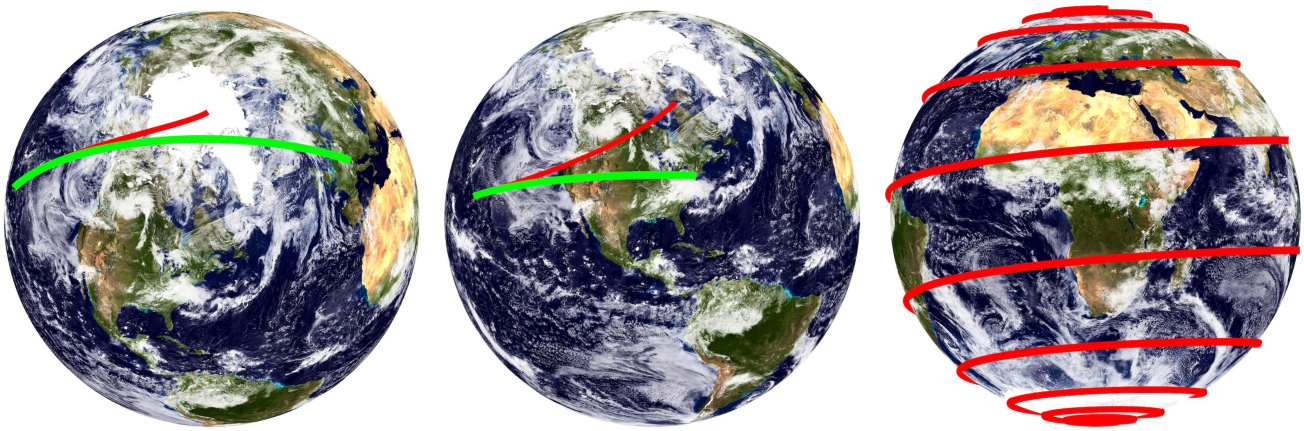
$$\|\dot{\mathbf{r}}^l_t\| = v = \frac{s}{N_{\text{steps}} \Delta_t} \tag{56}$$

is the speed needed for the target state.

[11]The algorithm of [35] and [34] is more robust than [57], but does not directly provide the latitude versus longitude of the curve to plot.

The thick, green line in Fig. 7(a) is the geodesic curve on the surface of the Earth between the Mauna Kea Observatory in Hilo, Hawaii, which is located at approximately $-155.470°$ East longitude and $19.823°$ North latitude [55, Sec. J], and Neuschwanstein Castle in Füssen, Germany, which, according to Google Maps,[12] is approximately located at $47.5575°$ latitude and $10.7500°$ longitude. The geodesic curve (which assumes a constant zero altitude above the reference ellipsoid) is determined using the algorithm of [57]. When the algorithm of Fig. 6 is used with a constant-velocity motion model propagated for the same distance (approximately 12,416 km) and the same heading as that determined by the algorithm of [57], the results differ by 0.0276 cm (1,000 Runge-Kutta steps spaced one second apart were used for the propagation). In Fig. 7(b), the geodesic trajectory under consideration goes from the Mauna Kea Observatory to New York City, which according to Google Earth is located at approximately $40.67°$ latitude and $-73.94°$ longitude. Again, the difference between the stopping point of the geodetic estimation algorithm and the propagation algorithm of Fig. 6 is 0.00727 cm after a travel distance of 7,903 km.

[12]Note that although Google Maps can be used to get approximate positions, its maps use the "web Mercator projection," which can result in positional biases to be as high as 40 km in some areas compared to a correct WGS-84-coordinates [53].



(a) Hilo to Füssen (b) Hilo to New York (c) A Constant-Bearing Path

Fig. 7. In (a) and (b), the thicker, green curve is a geodesic curve from Hilo, Hawaii to Füssen, Germany in (a) and to New York City in (b). At the scale shown, the geodesic curve, which is determined using the algorithm of [57], is indistinguishable from the prediction of a flat-Earth constant-velocity motion model applied to the spheroidal Earth when initialized with the same bearing. The red curve in (a) and (b) is a curve of constant bearing starting with the same bearing as the geodesic curve. That is the curve that the algorithms of [43], [81] mapping a non-maneuvering flat-Earth model to a curved Earth would follow. The curve gets stuck at the North Pole in (a) and deviates greatly North in (b). In (c), a curve of constant bearing heading 5° North of East starting in Antarctica is shown and clearly does not follow a geodesic path. The globe image is derived from an equirectangular projection provided by Visible Earth, which is run by the EOS Project Science Office at the NASA Godard Space Flight Center. Full credits for the image are at [68], [71]. The projection is re-mapped to a spherical Earth using the code of [44] (though the code is labeled for a Mercator projection, it actually uses an equirectangular projection), with minor modifications to use the WGS-84 ellipsoid. The images were made in Matlab and touched-up in Gimp, since Matlab tends to render three-dimensional lines in a broken, jagged manner.

The primary source of the small difference in the estimates between the traditional geodetic algorithm of [57] and the algorithm of Fig. 6 appears to stem from numerical precision errors that accumulate over the integration period. For example, after converting the location of Hilo to Cartesian coordinates using the formulae of Section IV-B and then back to ellipsoidal coordinates using the formulae of Appendix B, the heights differ by 9.3×10^{-10} m and the latitudes differ by 4.2×10^{-11} degrees. In comparison, the ratio of the error of the final Cartesian location to the distance traveled for the algorithm of Fig. 6 is 2.2×10^{-11} for the trajectory to Neuschwanstein and 9.1×10^{-12} for the path to New York. Consequently, the algorithm in this paper, which is implemented using double-precision arithmetic under a constant-velocity motion model, agrees with the geodesic algorithm of Appendix B to a reasonable precision.

The solid red lines in Fig. 7 are constant-bearing trajectories. Using the tracking techniques of [43] and [81], a target under a constant-velocity motion model propagated forward in time would follow a constant-bearing trajectory. Similarly, for purposes of trajectory generation, one might consider using a constant-bearing trajectory on a curved Earth. In Figures 7(a) and 7(b), the initial bearing matches that of the geodesic. However, it is clear that the constant-bearing trajectories deviate greatly from the geodesic path, because following a constant heading in a local ENU coordinate system always deviates toward the poles, except when following lines of constant latitude. In Fig. 7(c) a track maintaining a constant-bearing trajectory is started at a low angle of 5° North of East at a latitude of $\phi = -89°$, a longitude of $\lambda = 0°$, and a height of 0 m, which places it somewhere in Antarctica. The resulting rhumb line is observed to spiral greatly going up the globe, and is clearly not a geodesic curve. In other words, constant-bearing trajectories are not suitable for modeling non-maneuvering target motion near the poles or over long distances.

Next, the algorithm in this paper is considered for mapping a maneuvering target state to a curved Earth. Three models representing well-known maneuvers are chosen. The first model is a level coordinated turn model, as described in Appendix C. The second is a weaving motion model derived in Appendix D, and the third is a spiraling motion model, as derived in Appendix E. Appendices D and E also describe how to make the flat-Earth weaving and spiraling models travel a certain distance in a desired direction, since those designing simulations will generally want to know how to make a target go from one desired location on the Earth to another. It will be shown that while the algorithm derived in this paper correctly keeps the targets from flying off into space, the curvature of the Earth induces a slight drift in the trajectories in the local tangent plane compared to a geodesic starting in the same direction when adapting flat-Earth navigation methods to a curved Earth. Nonetheless, the flat-Earth equations for the overall trajectory to have a particular offset along an initial bearing as given in the appendices are good approximations to the curved-Earth result.

For the coordinated turn model, an example of an aircraft circling the Mauna Loa Volcano in Hawaii is considered. The Mauna Loa volcano is located at a

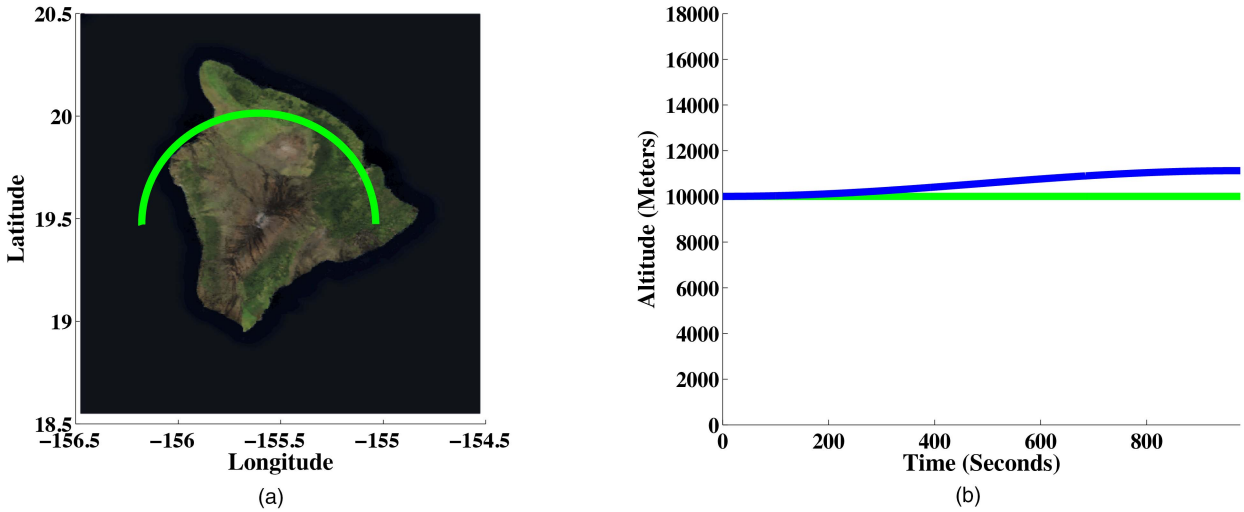


Fig. 8. The trajectory of a coordinated-turning plane around Hawaii as found using the algorithm of Fig. 6 is plotted in (a). At this angle and scale, the flat-Earth approximation would appear to be the same. Using the algorithm of Fig. 6, the target only differs from its starting altitude by 2.8 nm at the end due to numerical precision errors. On the other hand, using the flat-Earth model, the plane ends 1.13 km too high, as illustrated in (b), where the green line is the correct curved-Earth model and the blue-line is a flat-Earth approximation centered at the start of the target trajectory. The land image is again from Visible Earth [68], [72] and the image's contrast was increased to make the land more visible. The land image was displayed in Matlab using the `imread` and `imagesc` commands. To save the resulting plot while preventing Matlab from rendering the trajectory and axes as (pixellated) bitmaps, the trajectory and axes were saved separately from the land in encapsulated postscript (EPS) format, which is a format that supports vector graphics. The axes and trajectory were then combined with the plotted land image using PowerPoint, and the mixed vector/bitmap image was exported in portable document format (PDF). Had everything been plotted together, Matlab would have rasterized the vector graphics, even when saved in EPS format, causing the axes either to be very pixellated or the file size to be excessively large.

latitude of 19.475°, a longitude of −155.608° and is 4,170 m high [76]. The aircraft circling the volcano starts at a location 60 km along a geodesic East of the volcano (19.4741° latitude,−155.0365° longitude) at a heading of due North and an altitude of 10 km.[13]

To circle the volcano under a flat-Earth model, the radius of rotation r in Appendix C, should be approximately 60 km. The radius of rotation and the target speed are related by

$$\|\dot{\mathbf{r}}^l\| = v = r\omega, \tag{57}$$

where ω is the turn rate parameter in the coordinated turn model. Assuming that v = 193 m/s, which is a reasonable speed for a chartered private jet touring the island, as it is approximately the maximum cruise speed of an Eclipse 550 [17], the corresponding turn rate is approximately 0.184301°/s. The turn rate is only approximate when mapped to a curved surface, because Euclidean geometry is no longer valid and the time to go in a complete circle mapped to the surface of the ellipsoidal Earth is not necessarily 360° divided by the turn rate. Nonetheless, the flat-Earth (Euclidean) geometric approximation for the travel time is a reasonable approximation. Under Euclidean geometry, it would take about 33 minutes to circle the volcano.

If the aircraft is initially heading due North, at a constant speed for 16 minutes, the trajectory computed by the algorithm of Fig. 6 is plotted in Fig. 8 as a function of latitude and longitude using $N_{\text{steps}} = 1{,}000$ Runge-Kutta steps of duration $\Delta_t = 160$ ms. This trajectory maintains a constant altitude above the ground to within 2.8 nm at the final point in the trajectory (due to numerical precision errors). On the other hand, when using a flat-Earth coordinated turn model centered at the starting point[14] as described in Appendix C, the final target location is 1.13 km too high, with a total Cartesian offset of 1.15 km.

The second model under consideration is the weaving motion model of Appendix D. A target might travel in a weaving or other erratic manner over long distances to avoid detection while approaching a strategic destination. Here, two flat-Earth weaving models are mapped to the curved Earth. In the first model, the target weaves within the local tangent plane (xy plane) and should maintain a constant altitude above the ground. In the second model, the target performs weaves in the local vertical direction, and the weaves should not drift upward or downward over time.

[13]The location a desired distance along a geodesic East of the volcano can be found in the same way that the geodesic curves for Fig. 7 are evaluated. In this instance, the initial point is the volcano location, but set to zero altitude. One establishes an initial set of axes as an ENU coordinate system, and the initial local velocity is in the positive x direction. The algorithm in Fig. 6 is then used to travel at an arbitrary speed until the desired distance is traversed.

[14]The use of a local level coordinate system for putting flat-Earth models at any point on the Earth is described in [84, Ch. 3.2.2.7], where the flat-Earth model is placed on the local tangent plane of the starting point.

When using the weaving motion model of Appendix D, a target is considered flying a weaving trajectory at an altitude of 10 km from Mauna Loa to Honolulu (21.3000° latitude, $\lambda = -157.8167°$ longitude), according to Google Maps. The target's speed is set to 680 m/s, which is approximately Mach 2 and might just be attainable by the MiG in Table I. The relative amplitude of the weave is $\beta = 0.5$ with an initial phase of $\theta_0 = 0$ starting from time $t = 0$. The target performs $N_w = 6$ weaves along the path.

The geodesic distance from Mauna Loa to Honolulu is determined using the algorithm of [57] and is approximately $s \approx 306.552$ km. With the given weave parameters over this distance, Appendix D specifies the maximum acceleration felt by the pilot, ignoring gravity. Using the maximum gravity-free acceleration coupled with the G-force equations of Appendix C-B that account for gravity, the maneuver in the local tangent plane corresponds to maximum 4.2 G (load factor) turns. When the maneuver is in the vertical plane using the maximum acceleration of the turn with the maximum G-force equation of a vertical turn in Appendix C-B, one obtains an upper bound on the G-force felt by the pilot of 5.2 G. Both of those bounds are well within the structural limits of the aircraft listed in Table I, highlighting the realism of the maneuver.

The duration of the maneuver to travel the geodesic distance between Mauna Loa and Honolulu is determined using the method of Appendix D-B and is found to be approximately 529 s. However, when the target maintains an altitude of 10 km, it will stop short of the destination—the indirect geodetic problem solved with [57] is only valid for paths on the surface of the Earth. At higher altitudes, the target must travel farther to reach the same latitude and longitude.

To perform weaving motion in the local tangent plane, the axis of rotation used in the weaving model (the direction of Ω_t) is the local vertical $[0,0,1]'$, the z-axis. When the weaves are in the local vertical plane, then the axis of rotation is the normalized cross product of the initial (level-flight) velocity and the local vertical. In the simulations, 3500 Runge-Kutta steps were taken, which corresponds to a duration of approximately $\Delta_t =$ 151.2 ms per step in (56).

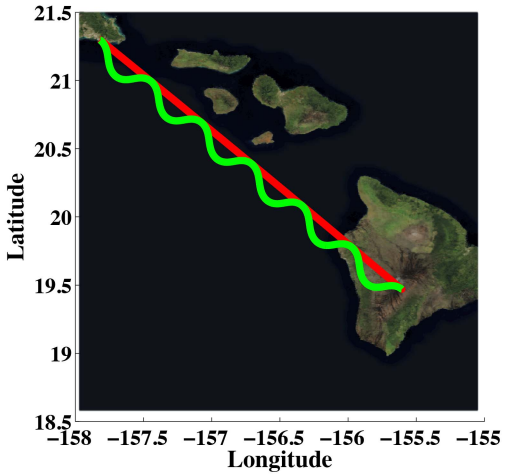
Figures 9(a) and 9(c) show the weaving trajectory after traveling the ground distance of the geodesic curve (as computed on the surface of the reference ellipsoid) for a target at an initial altitude of 10 km using the algorithm of Fig. 6. In Figures 9(a) and 9(b), the target weaves in the local tangent plane. At the final point, the target's altitude differs from the initial point by approximately 1.86 μnm when performing horizontal weaves and 57 nm when performing vertical weaves, which one would expect to be entirely due to numerical precision problems. However, the target's Cartesian location at the end differs from the end of the geodesic curve raised to the same altitude by 481 m when performing horizontal weaves and 798 m when performing vertical weaves. This offset in the local tangent plane is primarily because the geodesic curve is computed at zero altitude whereas the target flies at 10 km altitude, but also because the flattening factor of the Earth means that deviations in one direction do not perfectly cancel by deviation in another.

The third model considered is the spiraling model of Appendix E. The spiraling model is of interest, because full physics-based dynamic model derivations, such as in [73, Sec. 5], often omit expressions for a spiraling model on a curved Earth as being too difficult. While the flat-Earth model of Appendix E might not be the best physical representation of a spiraling target, it is sufficient to demonstrate the efficacy of the algorithm. As shown in Figures 9(e) and 9(f), the spiraling model varies its position in the horizontal and vertical planes, but does not drift upward.

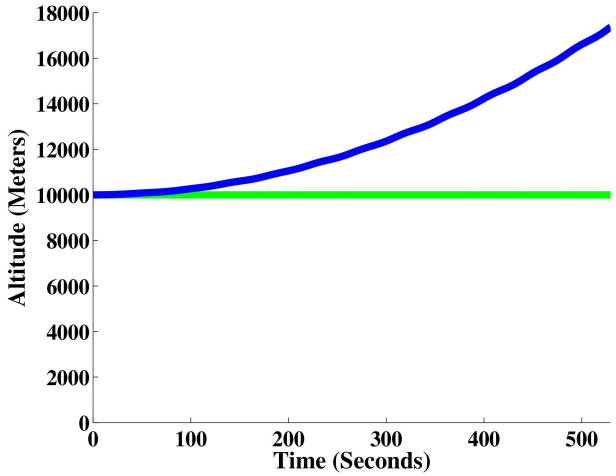
The same trajectory is chosen for the spiraling model as for the weaving models, though the initial altitude is increased to 20 km. The spiraling model is implemented with 3500 Runge-Kutta steps over a period of $T = 600$ s. The angular velocity of the turn component is chosen to be $\omega = 2\pi N_w/T$ with $N_w = 6$ so that there would be 6 spiral cycles over the trajectory. The linear speed of the spiral track is the distance between the points divided by the time, which is approximately $\|\mathbf{v}_l\| = 481$ m/s. The magnitude of the orthogonal velocity causing the rotation is set to cause the radius of the spiral to be 5 km, which is approximately $\|\mathbf{v}_s\| = 314$ m/s. The total speed of the target is consequently 574 m/s, which is less than Mach 2 and is attainable by two of the fighter jets in Table I. The upper bound on the G-force of the target as described in Appendix E is approximately 3 G and is tolerable by all of the aircraft in Table I.

VI. CONCLUSIONS

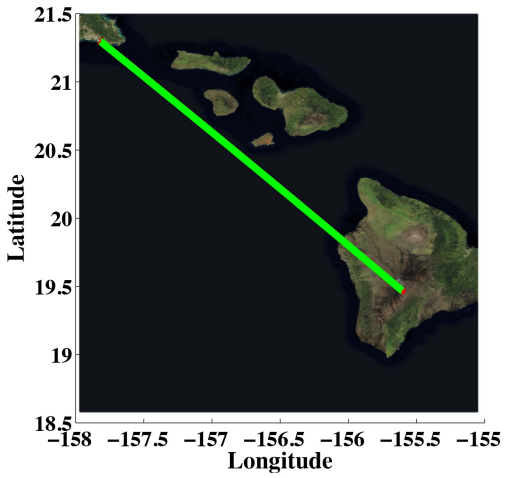
This paper shows how deterministic flat-Earth dynamic models can be used on a curved Earth via a naturally evolving coordinate system. The technique avoids the highly unrealistic tendency of many other algorithms to spiral toward the poles. When the target is constrained to the surface of an arbitrary-shaped Earth, the solution is exact, assuming that a set of coordinates can be defined on the surface. Similarly, when the Earth is approximated as an ellipse of revolution, as in the WGS-84 standard, then a precise algorithm for simulating targets at arbitrary altitudes is available, because no curvature of the plumb line exists in the ellipsoidal model. The moving local coordinate system is equivalent to the wander coordinate system from the inertial navigation literature. The algorithm of Fig. 6 is a summary of the technique for use on a reference ellipsoid. When used with a non-maneuvering motion model, the algorithm can be used to trace out geodesic curves given an initial heading without encountering singularities in any direction or at the poles. Unlike traditional geodesic



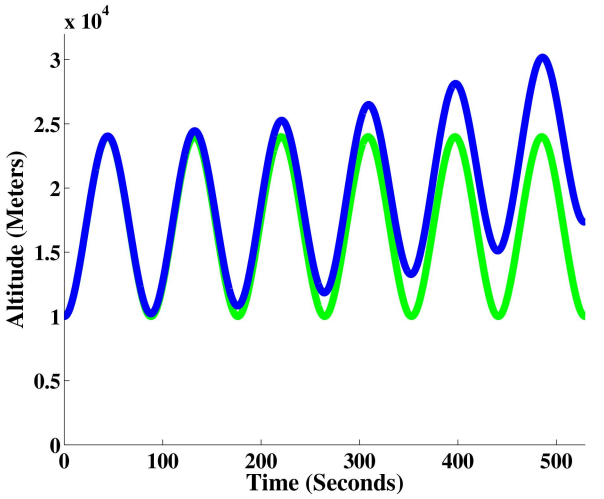
(a) Horizontal Sinusoid



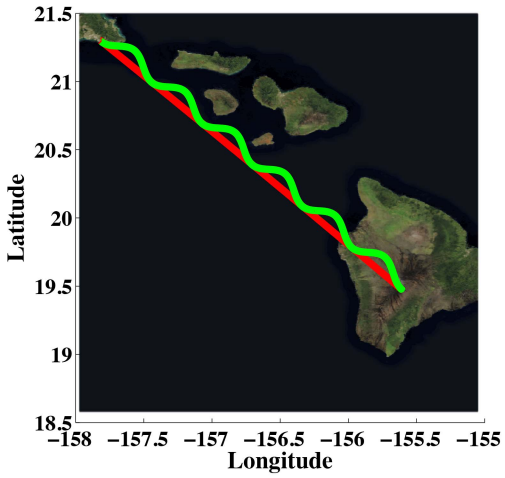
(b) Horizontal Sinusoid



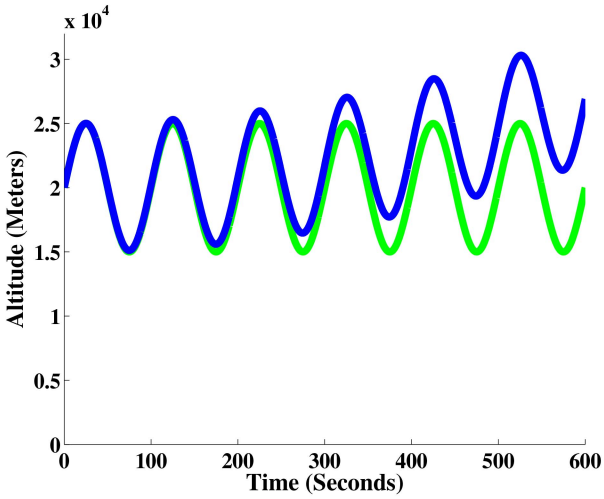
(c) Vertical Sinusoid



(d) Vertical Sinusoid



(e) Spiral



(f) Spiral

Fig. 9. A target traveling between Mauna Loa and Honolulu. In (a) and (b), the target follows a weaving trajectory in the local tangent plane; in (c) and (d), the weaving is in the vertical plane, and in (e) and (f) the target spirals. In (a), (c) and (e) the straight red curve is the geodesic curve; in (c), the geodesic is covered by the trajectory. In (b), (c) and (d) the green trajectory, which does not rise, is the model correctly accounting for the curvature of the Earth; the blue lines, which rise, are local flat-Earth models centered at the starting point of the trajectory. It can be seen that ignoring the curvature of the Earth introduces a large bias in altitude. Again, the land image is from Visible Earth [68], [72].

algorithms, it can compute geodesic curves at constant altitudes above the reference ellipsoid.

Through simulation, the algorithm of Fig. 6 is shown to agree with a traditional technique for determining geodesic curves when applied to a non-maneuvering, constant-velocity motion model. The algorithm did not exhibit a tendency to veer toward the poles, as occurs when using a constant heading model to adapt tracks to a curved Earth as is the case when using more traditional ENU methods. The generality of the algorithm is demonstrated with coordinated turn, weaving, and spiraling motion models. The limiting factor in the approach appears to be the accumulation of numerical precision errors over time, resulting in small (nanometer) deviations from the target height.

In addition to presenting a technique for mapping arbitrary flat-Earth dynamic models for aircraft to a curved Earth, this paper also analyzes the extent to which the ellipsoidal Earth approximation is valid when accounting for the fact that aircraft generally use pressure altitude for navigation purposes. This analysis is in Appendix A. This paper also derived flat-Earth weaving and spiraling models in Appendices D and E that do not appear to exist in the literature. The models were derived along with methods for determining the G-forces felt by a pilot and determining how to travel a desired distance in a particular direction while maneuvering.

The most advanced algorithms for simulating target dynamics are probably those built into commercial and military flight simulators, such as the many listed in Jane's online database of Simulation and Training Systems (part of https://janes.ihs.com). However, for simple design of target tracking algorithms, the cost and computational resources required for the high-fidelity models in Janes might not be practicable, which is probably why such advanced models appear to be absent from the target tracking literature, where simple, generic flat-Earth models are commonly used. Moreover, highly precise models often have a large number of parameters to be estimated, not all of which are observable by a radar (comments on the observability of one sophisticated model are given in [40]). Consequently, medium-fidelity dynamic models might be more useful in many tracking algorithm designs. Thus, the technique presented in this paper can fall into line with those of [50], [62] for long-range maneuvering target simulations. However, unlike the methods of [50], [62], the algorithm of this paper can also simulate targets on non-ellipsoidal surfaces.

Future work can focus on developing state prediction algorithms that use meteorological data to model properly the altitude variations experienced by aircraft navigating using pressure altitude. Similarly, real aircraft do not fly deterministic trajectories. Ergo, future work can generalize the technique of this paper to map stochastic motion models to a curved Earth. Nonetheless, deterministic flat-Earth trajectories mapped to a curved Earth can specify nominal maneuvering trajectories used by real control systems on aircraft or missiles.

ACKNOWLEDGEMENTS

This research is supported by the Office of Naval Research through the Naval Research Laboratory (NRL) Base Program. The author would like to thank Dr. Eric Mokole, and Mr. Steven Brockett from NRL for their editorial feedback on this manuscript, Dr. Jimmy Alatishe and Mr. Jerry Kim for their discussions on differential geometry, and Mr. Scott R. Duncan for showing where free, high-resolution Earth imagery could be obtained.

APPENDIX A. HOW AIR PRESSURE VARIATIONS AFFECT CIVILIAN AVIATION

Aircraft in controlled airspace normally fly at altitudes computed from measuring a local barometric pressure and comparing it to a reference mean sea level value. The reference pressure at sea level in controlled airspace is usually set to the standard pressure of 29.92 inches of mercury [21]. The pressure altitude is normally determined using a standard atmospheric model. A number of global and regional standard atmospheric models are reviewed in [2], not all of which are relevant to aviation.[15] For civil aviation, the International Civil Aviation Organization (ICAO) Standard Atmosphere, the International Standard Atmosphere (ISA), from the International Organization for Standardization (ISO), and the U.S. Standard Atmosphere are the most relevant. As noted in [2], the ISA is identical to the ICAO standard atmosphere and the World Meteorological Organization (WMO) Standard Atmosphere up to an altitude of 32 km. The development of the U.S. Standard atmosphere is described in [47], where it is noted that the U.S. Standard Atmosphere and the ICAO standard atmosphere are identical up to an altitude of 32 km. The defining document for the U.S. Standard atmosphere is [54].[16] In the United States, civilian aircraft are limited to an altitude of 40,000 ft (12,192 m), with a single exemption for the Airbus A380-800, which can fly up to 43,000 ft (13,106.4 m) [14].

The U.S. Standard Atmosphere [54] provides equations relating altitude and pressure within the altitudes concerning civilian aviation as well as beyond those bounds. Three equations relating altitude and pressure are given in the standard. One is for altitudes from 11 km to 20 km and 47 km to 51 km. Another is for

[15] Additionally, some of the models are for Mars, Neptune, Titan, and Venus, rather than for the Earth.

[16] The ICAO standard atmosphere and the ISA are not directly cited, because the U.S. Standard Atmosphere can be freely downloaded, whereas the other two standards are rather expensive. For example, one can purchase the ISA from the ISO at http://www.iso.org/iso/catalogue_detail?csnumber=7472 for 210 Swiss Francs ($\approx$ 230 U.S. dollars). Since all three standards agree from ground level to 32 km altitude, the freely available U.S. Standard Atmosphere generally suffices for civilian aviation purposes worldwide.

altitudes above 85 km, and the third is for all other altitudes, but with different parameters in different regions. Consequently, two different sets of equations are necessary to describe all altitudes up to 43,000 ft, the maximum altitude of a commercial aircraft in the United States. However, to simplify the analysis here, only the model from ground level to 11 km (36,089 ft) is considered.

The relationship between the altitude above mean sea level z and the atmospheric pressure for altitudes from ground level to 11 km is

$$z = \frac{r_0 h}{r_0 - h}, \tag{58}$$

where h is the geopotential altitude given by

$$h = \frac{T_0}{L_0}\left(\left(\frac{P}{P_0}\right)^{-L_0 R/g_0 M_0} - 1\right) \tag{59}$$

and

$$r_0 = 6{,}356{,}766 \text{ m} \qquad g_0 = 9.80665 \frac{\text{m}}{\text{s}^2} \tag{60}$$

$$R = 8314.32 \frac{\text{N m}}{\text{kmol K}} \quad M_0 = 18.9644 \frac{\text{kg}}{\text{kmol}} \tag{61}$$

$$P_0 = 101{,}325.0 \text{ Pa} \qquad T_0 = 288.15 \text{ K} \tag{62}$$

$$L_0 = -6.5 \times 10^{-3} \frac{\text{K}}{\text{m}}. \tag{63}$$

The value r_0 is the effective radius of the Earth and g_0 is the magnitude of the acceleration due to gravity at sea level, which differ from the more modern values derived from the WGS-84 standard [13]. The value R is the ideal gas constant, also known as the molar gas constant. The value of R in (61) is provided in [54], which differs slightly from the modern value standardized by the Committee on Data for Science and Technology (CODATA) in [48]. The quantities M_0 and P_0 are respectively the mean molecular weight of the atmospheric constituents at the surface of the Earth and the standard atmospheric pressure at sea level. The air pressure at the surface of the Earth P_0 is approximately equal to 29.92 inches of mercury on a barometer at the standard temperature $T_{M,0} = 15°\text{C}$. Note that many sources erroneously use the geopotential altitude in (59) as the true altitude instead of (58).

The relationship between the geopotential altitude h and the true altitude above sea level z is

$$h = \frac{1}{g_0}\int_0^z \|\mathbf{g}\| dz, \tag{64}$$

where $\mathbf{g}$ is the acceleration due to gravity at altitude z and g_0 is the magnitude of the acceleration due to gravity at mean sea level.[17] Equation (59) comes from using a very simple approximation to the magnitude of the acceleration due to gravity at height z, specifically,

$$\|\mathbf{g}\| = g_0\left(\frac{r_0}{r_0 + z}\right)^2. \tag{65}$$

In other words, by modern standards, the U.S. Standard atmosphere is not a high-precision model. However, it forms the basis of pressure-based altimeters for civilian aviation and is a good approximation for basic analysis.

To determine how much local pressure variations can affect the true altitude of an airplane flying according to a pressure altitude, an example of an aircraft flying at a pressure altitude of 7,620 m (25,000 ft) is considered. Using (58) with the standard parameters shown, the pressure at the aircraft's altitude is $P = 52{,}991.6$ Pa. The maximum and minimum (excluding tornadoes) recorded sea-level air pressures are 108,330 Pa and 87,000 Pa [39]. Using $P = 52{,}991.6$ Pa for an aircraft at a constant pressure altitude of 7,620 m, but varying P_0 to both of the extremes, the true altitude varies from 5,953.45 m (19,532 ft) to 8,328.2 m (27,323 m).

In comparison, in the United States, aircraft are placed in 500 ft ($\approx$ 152 m) flight levels. A flight level is the height in feet divided by 100. The use of flight levels begins at 18,000 ft, corresponding to flight level 180, denoted FL180. For pilots operating under instrument flight rules (IFR), which is generally all pilots flying in flight levels, air traffic controllers space the aircraft through FL410 a distance of 1,000 ft apart vertically. That is, they are two flight levels apart. Above FL410, aircraft are spaced 2,000 ft apart, except for supersonic and military aircraft [21]. If a pilot sets his altimeter to use the standard MSL air pressure and the air pressure is at the record minimum, then the pilot will fly about 10.9 flight levels lower, compared to the ground, than on a normal day. On the other hand, if the air pressure is the record high, then the pilot will fly about 4.6 flight levels too high.

In practice, the extreme low is unrealistically extreme, since it was taken in the middle of Typhoon Tip in 1979, and generally only research and military aircraft would knowingly venture into a strong hurricane. A more realistic lower bound for the air pressure is 98,000 Pa, which is the very bottom of the pressure range of a category 1 hurricane on the Saffir-Simpson scale [3], and is the value used as a lower bound on the atmospheric pressure in [18], where the feasibility of reducing the minimum vertical separation between planes (which was larger back then) was studied. In this instance, the minimum air pressure extreme leads to an altitude of 7261.4 m (23,823 ft), which is an offset of 2.4 flight levels. Additionally, when the air pressure extrapolated to mean sea level in a location is above 31 inches of mercury ($\approx$ 104,966 Pa), air traffic control in the United States is supposed to instruct pilots to set their altimeters to use 31 inches of mercury as P_0 [21];

[17] Mean sea level is defined as a surface of constant gravitational potential. Surprisingly, it is *not* a surface where the magnitude of the gravitational acceleration $\|\mathbf{g}\|$ is constant. In [29, Ch. 2.9], the magnitude of the gravitational acceleration on an ellipsoidal approximation of the Earth is derived and varies with latitude. Thus, the assumption that g_0 is a single constant everywhere is just an approximation.

so in many instances the vertical offset due to pressure changes also spans fewer flight levels.

This variability in altitude of an aircraft traveling at a constant barometric altitude exceeds the variability in the height above the reference ellipsoid due to gravitational variations when using an ellipsoidal approximation of the Earth (Using the data in [52], the reference ellipsoid differs from mean seal level by up to +84 m and −107 m). Consequently, if one wants more precise aircraft prediction models on a curved Earth than the technique presented in this paper, one must consider the use of meteorological data, not just the use of a more precise geoid approximation than an ellipsoid.

APPENDIX B. CONVERTING FROM CARTESIAN TO GEODETIC ELLIPSOIDAL COORDINATES

The conversion from Cartesian (x_0, y_0, z_0) to ellipsoidal (ϕ, λ, h) coordinates is a difficult problem. A table of 80 references addressing the topic is given in [20]. Generally, an explicit, numerically stable solution is the most desirable. In this appendix, the solution of [66], which is a stabler form of [65], is reviewed. When using typical parameters for the Earth's ellipse, it is noted in [65] and [66] that this conversion is valid for all points outside of a small ellipsoid around the center of the Earth, whose radius is about 43 km. Considering that no one has ever managed to drill through the 6 km of crust of the Earth to reach the mantle, though it might be technically feasible [70], this restriction is in practice meaningless for all applications of this coordinate conversion outside of, perhaps, seismic research.

It is assumed that the semi-major axis a and semi-minor axis b of the reference ellipsoid are known and that the reference ellipsoid is centered on the Cartesian origin. Section IV describes the relationship between the different formulations of ellipsoidal parameters. Given the Cartesian point (x_0, y_0, z_0), the corresponding latitude, longitude, and ellipsoidal height are

$$\phi = \arcsin\left(\frac{z(\epsilon^2+1)}{N_e}\right) \tag{66}$$

$$\lambda = \arctan 2(y_0, x_0) \tag{67}$$

$$h = r_0\cos(\phi) + z_0\sin(\phi) - \frac{a^2}{N_e}, \tag{68}$$

where z, ϵ, N_e, and r_0 are found by computing

$$e^2 = 1 - \frac{b^2}{a^2} \qquad \epsilon^2 = \frac{a^2}{b^2} - 1 \tag{69}$$

$$r_0 = \sqrt{x_0^2 + y_0^2} \qquad p = \frac{|z_0|}{\epsilon^2} \tag{70}$$

$$s = \frac{r_0^2}{e^2\epsilon^2} \qquad q = p^2 - b^2 + s \tag{71}$$

$$u = \frac{p}{\sqrt{q}} \qquad v = \frac{b^2u^2}{q} \tag{72}$$

$$P = 27\frac{vs}{q} \qquad Q = \left(\sqrt{P+1} + \sqrt{P}\right)^{2/3} \tag{73}$$

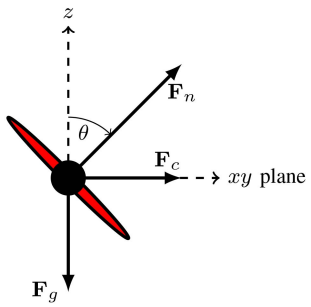


Fig. 10. The force diagram used to derive the relationship between the turn rate and the load factor. The force $\mathbf{F}_g$ is the force due to gravity and is aligned with the negative z-axis. The force $\mathbf{F}_c$ is the centripetal force, which points toward the center of the circle of rotation when turning. During a level, coordinated turn, the aircraft remains in the xy plane, which is perpendicular to the z-axis, with θ being the bank angle of the aircraft. In the ideal scenario, a lift force $\mathbf{F}_n$ is only produced in a direction perpendicular to the wings of the aircraft. When the aircraft is not turning, the lift is straight up and counters $\mathbf{F}_g$. When the aircraft is in a coordinated turn, the lift is such that the plane does not move in the z direction during the turn and such that the required centripetal force $\mathbf{F}_c$ is present to make the turn occur.

$$t = \tfrac{1}{6}(1 + Q + Q^{-1})$$

$$c = \sqrt{u^2 - 1 + 2t} \tag{74}$$

$$w = \frac{c-u}{2} \tag{75}$$

$$z = \operatorname{sign}(z_0)\sqrt{q}\left(w + \sqrt{\sqrt{t^2+v} - uw - \frac{t}{2} - \frac{1}{4}}\right) \tag{76}$$

$$N_e = a\sqrt{1 + \frac{\epsilon^2 z^2}{b^2}}. \tag{77}$$

The term e is known as the first numerical eccentricity; ϵ is the second numerical eccentricity. The four-quadrant inverse tangent in (67) is not uniquely defined at the poles, when $x = 0$ and $y = 0$. However, many implementations of the four-quadrant inverse tangent, such as the `atan2` function in Matlab, will return zero in that instance. On the other hand, the `ArcTan` function in Mathematica correctly returns an indeterminate quantity. In a practical system, it is often preferable for a zero longitude to be returned at the poles rather than something indeterminate.

Care must be taken with the outer square root in (76), because when the argument is near zero, finite precision problems have been observed to cause the argument to become slightly negative. A practical implementation should check for negativity and insert zero instead. A similar check might be necessary for the square root in (74).

APPENDIX C. A FLAT-EARTH PLANAR TURN MODEL

A. Deriving a Model for Circular, Planar Turns

A coordinated turn is one where an aircraft maintains a constant altitude and speed. Figure 10 is an ap-

proximate free body diagram (ignoring drag, wind, and the Coriolis force) of the forces affecting the center of mass of a turning aircraft. The primary forces are gravity $\mathbf{F}_g$, which in a flat-Earth model can be aligned with the local z-axis, and lift $\mathbf{F}_n$, which is assumed to act normally to the body of the aircraft. To maintain a constant altitude, $\mathbf{F}_g + \mathbf{F}_c$ should act perpendicularly to the gravity vector. If $\|\mathbf{F}_c\|$ is constant, then the shape of the aircraft's maneuver will be a circle.

Constant-speed circular motion causes the acceleration to be orthogonal to the velocity. If $\ddot{\mathbf{r}}_t$ denotes acceleration, then

$$\ddot{\mathbf{r}}_t = \Omega_t \times \dot{\mathbf{r}}_t, \tag{78}$$

where $\times$ denotes the cross product, the direction of Ω is the axis of rotation, and the magnitude of Ω_t determines the turn rate at time t. For a coordinated turn on a flat Earth, the turn must remain in the xy plane. Consequently, the full dynamic model is

$$\mathbf{a}(\mathbf{x}_t, t) = \begin{bmatrix} \dot{\mathbf{r}}_t \\ \Omega_t \times \dot{\mathbf{r}}_t \\ 0 \end{bmatrix} \qquad \Omega_t = \begin{bmatrix} 0 \\ 0 \\ \omega_t \end{bmatrix} \tag{79}$$

for a target state $\mathbf{x}_t = [\mathbf{r}_t, \dot{\mathbf{r}}_t, \omega_t]'$. The model in (79) is the same as the basic two-dimensional horizontal coordinated turn model in [40]. This model can easily be expanded to a turn in an arbitrary plane by setting $\Omega_t = \omega_t \mathbf{u}_{\text{rot}}$, where $\mathbf{u}_{\text{rot}}$ is the rotation axis desired. To make the model a flat-Earth approximation in a local tangent plane on a curved Earth, set $\mathbf{u}_{\text{rot}}$ to the local vertical at the point of reference. In Section V, the point of reference was taken to be the starting point of the trajectory.

B. Relating Turn Rates to G-Forces

Section V related the turns in the simulations to the force felt by the pilot so as to justify the realism of the turn rates used. One feels 1 G when standing on the ground. A pilot will typically lose consciousness for turns producing between 4.5 and 5 Gs, whereas military pilots wearing "anti-G" suits can sustain up to 8 Gs. Combat aircraft are usually designed for load factors of at least 8 Gs [45]. Obviously, unmanned aircraft could be designed to withstand turns that would render a pilot unconscious. A reasonable baseline for simulation design can be the parameters for the F-16 Fighting Falcon, which are given in Table I.

The term ω_t is the turn rate in radians per second and can be related to the force felt by the pilot using simple equations for centripetal acceleration. In the coordinated turn model, ignoring wind and drag, the force that the aircraft must generate to maintain altitude and speed during the turn must counteract both the gravitational force and the centripetal force, as shown in Fig. 10. The centripetal force can be written from the acceleration in (79) as

$$\mathbf{F}_c = m\Omega_t \times \dot{\mathbf{r}}, \tag{80}$$

where m is the mass of the aircraft. The force that the aircraft must generate to maintain altitude and speed during the turn must counteract both the gravitational force and the centripetal force, as shown in Fig. 10. This normal force, designated by $\mathbf{F}_n$, is the lift produced by the plane such that a centripetal force of $\mathbf{F}_c$ is obtained. The normal force is thus

$$\mathbf{F}_n = \mathbf{F}_c - \mathbf{F}_g \tag{81a}$$

$$= m\Omega_t \times \dot{\mathbf{r}} - m\mathbf{g}, \tag{81b}$$

where $\mathbf{g}$ is the acceleration due to gravity ($\|\mathbf{g}\| \approx 9.81\text{m/s}^2$) on the surface of the Earth.

Dividing out the mass in (81b), the acceleration is obtained. When the magnitude of the acceleration is divided by $\|\mathbf{g}\|$, the load factor, which is the G-force felt by the pilot, is obtained[18]

$$\text{G-force} = \frac{\|\Omega_t \times \dot{\mathbf{r}} - \mathbf{g}\|}{\|\mathbf{g}\|}. \tag{82}$$

Equation (82) is useful in that it can express the G-force felt by the pilot not only for coordinated turns, but also for circular turns in any axis. In such an instance, the vector Ω_t need not point to the local vertical. Equation (82) was used in Section V to find the maximum G-force of the turns under the weaving maneuver model of Appendix D for weaves outside of the local tangent plane.

When performing a level coordinated turn, that is Ω_t is as in (79), then (82) can be simplified to

$$\text{G-force} = \frac{\sqrt{(\omega_t)^2 \|\dot{\mathbf{r}}\|^2 + \|\mathbf{g}\|^2}}{\|\mathbf{g}\|}. \tag{83}$$

In such an instance, the G-force can be related to the bank angle θ of the aircraft in Fig. 10 by substituting (83) into the expression for the tangent of θ to obtain

$$\text{G-force} = \sqrt{(\tan[\theta])^2 + 1}. \tag{84}$$

On the other hand, if the turn is in the vertical direction, then the maximum G-force is felt when the cross product of the turn axis and velocity vector aligns with the gravitational acceleration and is

$$\text{G-force} = 1 + \omega \frac{\|\dot{\mathbf{r}}\|}{\|\mathbf{g}\|}. \tag{85}$$

More sophisticated aircraft dynamic models, such as [49] and other models based on aircraft aerodynamics as surveyed in [40], require significantly more elements in the target state. Aircraft turns need not always be circular in nature, especially if the turn is not in the tangent plane. For example, parabolic trajectories are often used

[18] Equation (82) does not differentiate between positive (feel pushed down in one's seat) and negative (feeling pulled out of one's seat) G-forces (load factors). Aircraft typically have different tolerances for positive and negative load factors before they break. As per Part 23, Section 337 of the FAA's Federal Aviation Regulations, the negative load factor that small aircraft in the United States must withstand is only 40%–50% the magnitude of the maximum tolerable positive load factor.

to obtain weightless environments without going into space [33]. Nonetheless, the model in this appendix can be useful, because it requires fewer parameters than the sophisticated model of [49] and can thus be initialized with a smaller number of measurements if used in a target tracker. For general circular turns, the orientation of Ω_t would also have to be estimated.

APPENDIX D. A FLAT-EARTH WEAVING TARGET MODEL

A. Deriving a Model for Variable-Rate Planar Turns

Weaving target models often arise when designing proportional navigation systems for missiles, such as in the examples discussed in [82, Chs. 21, 26, 30] and [56], where weaving is considered as a countermeasure. A simple weaving target model can be obtained by varying the turn rate ω_t in the coordinated turn model of Appendix C as a function of time.

Consider setting

$$\omega_t = A\cos(\alpha t + \theta_0), \tag{86}$$

where α is the weave rate, θ_0 is a phase offset, and A is the weave amplitude. The weave amplitude determines how much the target varies from a straight trajectory. For simplicity, it is assumed that $A > 0$. A reasonable upper bound on the weave amplitude would be such that the target traces out a series of 180° turns that alternate in direction so that the target moves forward. A weave amplitude resulting in total turns greater than 180° would have the target doubling back on its path in parts and not necessarily following a clear trajectory.

To determine the upper bound on A, consider the case where $\theta_0 = 0$, without loss of generality. In such an instance, at time $t = 0$, ω_t is at its maximum. The integral of ω_t over time gives the total turn. For a target to make a series of 180° turns of alternating directions, the accumulated angular change should be a quarter of a circular rotation; that is

$$\frac{\pi}{2} = \int_0^{\pi/2\alpha} A\cos(\alpha t)dt. \tag{87}$$

Consequently, the maximum allowable value of A before the trajectory starts describing loops is

$$A \leq \frac{\pi\alpha}{2}. \tag{88}$$

Smaller values of A make the track straighter. To make the overall direction of the weaving track follow a desired path, set the initial velocity point in the overall direction of the path and set $\theta_0 = -\alpha t$ at the initial time t when the weaving trajectory begins.

B. Determining How to Arrive At a Desired Point

In the simulations, the parameters were chosen such that a desired integer number of weaves were made from the starting position to get to an ending position a known distance away. In this section, the time and speed needed to perform the desired number of weaves and end up at the desired ending position when the weaves are made is derived. It is assumed that the track begins with $\alpha t + \theta_0 = 0$ and, without loss of generality, that the x-axis is aligned with the initial direction of motion and the weave is in the xy plane. The assumption that $\alpha t + \theta_0 = 0$ can be replaced by setting the initial time to 0 and $\theta_0 = 0$. These assumptions mean that the trajectory reaches the desired point at a time when $y = 0$. Using (86) and (79), along with these assumptions, the accelerations in the x and y coordinates can be written

$$\ddot{r}_{x,t} = -A\dot{r}_{y,t}\cos(\alpha t) \tag{89}$$

$$\ddot{r}_{y,t} = A\dot{r}_{x,t}\cos(\alpha t). \tag{90}$$

Solving for $\dot{r}_{y,t}$ in (89),

$$\dot{r}_{y,t} = -\frac{\ddot{r}_{x,t}}{A\cos(\alpha t)}, \tag{91}$$

and taking the derivative of (91) with respect to time yields

$$\ddot{r}_{y,t} = -\frac{\dddot{r}_{x,t} + \alpha\ddot{r}_{x,t}\tan(\alpha t)}{A\cos(\alpha t)}. \tag{92}$$

By combining equations (90) and (92), one obtains the differential equation

$$0 = \frac{\dddot{r}_{x,t}}{A\cos(\alpha t)} + \frac{\ddot{r}_{x,t}}{A\cos(\alpha t)}(\alpha\tan(\alpha t)) + A\dot{r}_{x,t}\cos(\alpha t). \tag{93}$$

It can be verified that a solution to (93) is

$$\dot{r}_{x,t} = c_1\cos\left(\frac{A\sin(\alpha t)}{\alpha}\right) + c_2\sin\left(\frac{A\sin(\alpha t)}{\alpha}\right). \tag{94}$$

The values of c_1 and c_2 can be directly found from the initial conditions at time $t = 0$: $\dot{r}_{x,t}|_{t=0} = \dot{r}_{x,0}$ and $\dot{r}_{y,t}|_{t=0} = 0$. Applying these initial conditions to (91) and (94) yields

$$c_1 = \dot{r}_{x,0} \qquad c_2 = 0, \tag{95}$$

so that (94) becomes

$$\dot{r}_{x,t} = \dot{r}_{x,0}\cos\left(\frac{A\sin(\alpha t)}{\alpha}\right). \tag{96}$$

Next, it is known that at the final time on the trajectory t_{end}, which is still unknown, the y position of the target should be equal to the initial y position (it ends with an integer number of weaves). Without loss of generality, assume that $r_{y,0} = 0$, which implies

$$r_{y,t_{\text{end}}} = \int_0^{t_{\text{end}}} \dot{r}_{y,t}dt = \int_0^{t_{\text{end}}} \dot{r}_{x,0}\sin\left(\frac{A}{\alpha}\sin(\alpha t)\right)dt = 0. \tag{97}$$

The ending position with respect to the y axis is zero when

$$\alpha = \frac{2\pi N_w}{t_{\text{end}}}, \tag{98}$$

where N_w is the positive integer number of weaves desired, because the integral is over an integer number of periods of the inner sine in (97) and the outer sine has

no phase offset. Thus, for every t such that the argument of the integral has a positive value, there exists a t offset by a fraction of the period of the inner sine to cancel that.

Using (96) and (98), and assuming without loss of generality that the $r_{x,0} = 0$, the distance traveled in the x direction up to time t_{end} is

$$\begin{aligned} r_{x,t_{\text{end}}} &= \int_0^{t_{\text{end}}} \dot{r}_{x,t} dt \\ &= \dot{r}_{x,0} \int_0^{t_{\text{end}}} \cos\left(\frac{A t_{\text{end}}}{2\pi N_w} \sin\left(\frac{2\pi N_w}{t_{\text{end}}} t\right)\right) dt. \end{aligned} \quad (99)$$

The distance traveled $r_{x,t_{\text{end}}}$ is a known quantity that gets one to the desired location along a geodesic curve from the initial location. The only unknown term in (99) is the travel time t_{end}.

From (98) and (88), the bound on A becomes

$$A \leq \frac{\pi^2 N_w}{t_{\text{end}}}. \quad (100)$$

If A is equal to the maximum value in (100), the argument of the cosine function in (99) is within the range $\pm\pi/2$, so the cosine function is always positive. However, if a larger A is used, then the cosine function can be negative, causing the target to go backwards in the x direction. The backwards motion of the target with A larger than the limit in (100) confirms the limit on A of Subsection D-A. To simplify the notation in the following optimization routine, A is expressed as a scale factor β ($0 < \beta \leq 1$) times the maximum allowable value of A; that is,

$$A = \frac{\pi^2 N_w}{t_{\text{end}}} \beta. \quad (101)$$

Figure 11 shows how the shape of the curve changes with different values of β.

Given bounds on t_{end} ($t_{\text{end}} < 3 r_{x,t_{\text{end}}} / \dot{r}_{x,0}$ was used as an *ad-hoc* upper bound in the simulations) and noting that (99) increases monotonically for allowable values of β, one can numerically estimate t_{end} with the optimization

$$\hat{t}_{\text{end}} = \arg\min_{t_{\text{end}}} \left| r_{x,t_{\text{end}}} - \dot{r}_{x,0} \int_0^{t_{\text{end}}} \cos\left(\beta \frac{\pi}{2} \sin\left(\frac{2\pi N_w}{t_{\text{end}}} t\right)\right) dt \right|. \quad (102)$$

Equation (102) can be numerically solved using a line search technique, such as the golden section search described in [4, Appendix C], with a numerical integration technique, such as those described in [6, Ch. 4]. In Matlab, one can use the function `fminbnd` to perform the minimization with `integral` for the numerical integration. In Mathematica, one can use the `NMinimize` command for the minimization with the `NIntegrate` command for the integration. Consequently, given a speed $\dot{r}_{x,0}$, a desired distance traveled along the initial direction of motion $r_{x,t_{\text{end}}}$, a value of β determining the relative amplitude of the weave, and the number of weaves N_w, the time necessary to travel the desired distance can be determined.

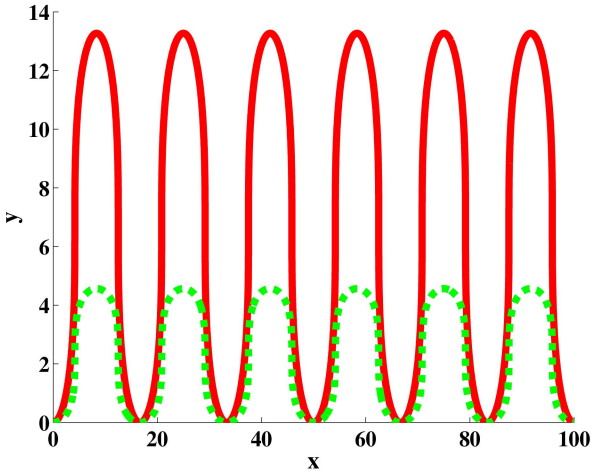


Fig. 11. The weaving motion for the same displacement, speed, and number of weaves varying only β, whereby t_{end} also changes so the same endpoint is reached. The solid red line is $\beta = 1$; the dashed green line is $\beta = 1/2$. The axes are in unit of meters, the speed is a constant 10 m/s, and $N_w = 6$.

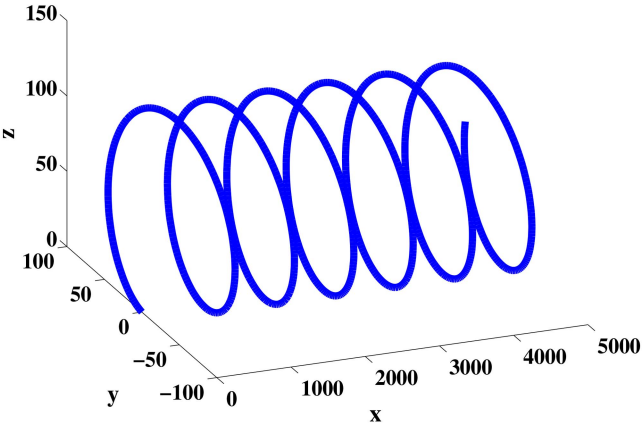


Fig. 12. An example of a spiraling trajectory starting from the origin with $\mathbf{v}_l = [100 \text{ m/s}, 0, 0]'$, $\mathbf{v}_s = [0, 50 \text{ m/s}, 0]'$, and $\omega = \pi/4$. The axes are in units of meters.

On the other hand, if one wishes to create a simulation that goes a particular distance $r_{x,t_{\text{end}}}$ weaving N_w times with amplitude scale factor β over a fixed period of time t_{end}, then the necessary speed is simply

$$\dot{r}_{x,0} = \frac{r_{x,t_{\text{end}}}}{\int_0^{t_{\text{end}}} \cos\left(\beta \frac{\pi}{2} \sin\left(\frac{2\pi N_w}{t_{\text{end}}} t\right)\right) dt}. \quad (103)$$

APPENDIX E. A FLAT-EARTH SPIRALING DYNAMIC MODEL

In [9], a non-ballistic flat-Earth spiraling dynamic model is derived by rotating the axis of a coordinates turn about the velocity vector.[19] One formulation of this

[19] Readers are advised to consider [59] if a ballistic spiraling model is desired.

model is

$$\ddot{\mathbf{r}} = \boldsymbol{\Omega} \times \dot{\mathbf{r}} \tag{104}$$

$$\dot{\boldsymbol{\Omega}} = \tilde{\omega} \left(\boldsymbol{\Omega} \times \frac{\ddot{\mathbf{r}}}{\|\dot{\mathbf{r}}\|} \right), \tag{105}$$

where $\boldsymbol{\Omega}$ is the instantaneous turn axis that is orthogonal to the velocity and $\tilde{\omega}$ is a constant affecting the rate of the spiral. Generally, $\boldsymbol{\Omega}$ does not have unit magnitude. The spiraling dynamic model of [9] is a constant-velocity model and ignores the effects of gravity. Unlike the weaving model of Appendix D, it is difficult to determine the actual direction in which the target will travel during the spirals.

Thus, an alternative constant-velocity spiraling model is presented here. Again, this is an approximate model that is not directly derived from aerodynamic and gravitational equations. The new spiraling model is

$$\dot{\mathbf{r}} = \mathbf{v}_l + \mathbf{v}_s \tag{106}$$

$$\dot{\mathbf{v}}_s = \omega \left(\frac{\mathbf{v}_l}{\|\mathbf{v}_l\|} \right) \times \mathbf{v}_s. \tag{107}$$

A drift equation corresponding to this model is

$$\mathbf{a}(\mathbf{x}_t, t) = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}}_s \\ \dot{\mathbf{v}}_s \end{bmatrix} \tag{108}$$

for a target state of $\mathbf{x}_t = [\mathbf{r}, \dot{\mathbf{r}}, \mathbf{v}_s]'$ ($\dot{\mathbf{r}}$ and $\mathbf{v}_s$ differ by a constant offset, so two of the derivative vectors in the drift equation are equal). Figure 12 shows an example of a trajectory arising from the spiraling dynamic model.

In practice, since $\mathbf{v}_s = \dot{\mathbf{r}} - \mathbf{v}_l$, the $\mathbf{v}_s$ term does not need to be stored directly in the state. In the model of (108), the velocity $\dot{\mathbf{r}}$ is the sum of orthogonal linear $\mathbf{v}_l$ and turning $\mathbf{v}_s$ components. The linear component is time-invariant. The turning component rotates about the linear component with a turn rate of ω rad/s. To make a target spiral in a particular direction, $\mathbf{v}_l$ should point in the desired direction of motion. The magnitude of $\mathbf{v}_l$ is the speed at which the target advances along the average trajectory about which the spiral turns, and $\mathbf{v}_s$ has to be orthogonal to $\mathbf{v}_l$. When programming a simulation of a spiraling target where the spiral neither ascends nor descends, it is reasonable to set the initial direction of $\mathbf{v}_s$ to the local vertical. In a flat-Earth model, this is generally the z-axis, since $\mathbf{v}_l$ would usually be placed in the xy plane. The magnitude of $\mathbf{v}_s$ determines the distance of the target from the center of the spiral as well as the G-forces felt by the pilot.

The relationship between the distance of the target from the center of the spiral and the magnitude of $\mathbf{v}_s$ can be determined by solving the differential equations for position. This determination can be made by assuming that $\mathbf{v}_l = [v_x^l, 0, 0]'$ and $\mathbf{v}_s = [0, v_s^y, v_s^z]'$. In such an instance, it can be verified that a solution to the differential equation in (107) has the form

$$v_s^y = c_1 \cos(\omega t) + c_2 \sin(\omega t) \tag{109}$$

$$v_s^z = c_2 \cos(\omega t) - c_1 \sin(\omega t), \tag{110}$$

where c_1 and c_2 are constants determined by the initial conditions and t is time. Assuming that at time $t = 0$, $v_s^y = 0$ and $v_s^z = v_M$, then $c_1 = 0$ and $c_2 = v_M$. Substituting $\mathbf{v}_s$, with y and z components given by (109) and (110), into (106) and integrating from time $t = 0$, at time t the target will have moved from its initial position by

$$\mathbf{r} = \begin{bmatrix} v_x^l t \\ \frac{v_M}{\omega}(1 - \cos(\omega t)) \\ \frac{v_M}{\omega} \sin(\omega t) \end{bmatrix}. \tag{111}$$

The magnitude of the y and z components, which are the only components affected by $\mathbf{v}_s$, is a constant v_M/ω and the rotational velocity is a constant $\|\mathbf{v}_s\| = v_M$. Consequently, the ratio of the rotational velocity to the turn rate determines the radius of the spiral. As for setting realistic bounds on the values so that the G-forces tolerable by a pilot will not be exceeded, the values in Appendix C-B are relevant given ω and the magnitude of $\mathbf{v}_s$, since those are the only velocity components that experience acceleration in the model.

Unlike with the weaving model of Appendix D, it is fairly simple to have the simplified flat-Earth spiraling target model go a desired distance in the xy plane: If $\mathbf{v}_l$ is in the xy plane and ω times the traveling time T is a multiple of 2π, then the target will be located at $\mathbf{v}_l T$ from its original position.

REFERENCES

[1] J. Alexander
Loxodromes: A rhumb way to go,
Mathematics Magazine, vol. 77, no. 5, pp. 349–350, Dec. 2004.

[2] "AIAA guide to reference and standard atmosphere models,"
American Institute of Aeronautics and Astronautics, Reston, VA, Tech. Rep. G-003B-2004, 17 Nov. 2004.

[3] E. L. Andreas
"Handbook of physical constants and functions for use in atmospheric boundary layer studies,"
U.S. Army Corps of Engineers, Engineer Research and Development Center, Cold Regions Research and Engineering Laboratory, Hanover, NH, Tech. Rep. ERDC/CRREL M-05-1, Oct. 2005.

[4] D. P. Bertsekas
Nonlinear Programming,
2nd ed. Belmont, MA: Athena Scientific, 2003.

[5] R. L. Bishop
There is more than one way to frame a curve,
The American Mathematical monthly, vol. 82, no. 3, pp. 246–251, Mar. 1975.

[6] R. L. Burden and J. D. Faires
Numerical Analysis,
9th ed. Boston, MA: Brooks/Cole, 2011.

[7] K. Carlton-Wippern
"Surface navigation and geodesy a parametric approach,"
Air Force Space Command, Tech. Rep., 1 Mar. 1988. [Online]. Available: http://www.dtic.mil/docs/citations/ADA227907.

[8] D. Cressey
Arctic melt opens northwest passage,
Nature, vol. 449, 20 Sep. 2007.

[9] D. F. Crouse
Basic tracking using nonlinear continuous-time dynamic models,
IEEE Aerospace and Electronic Systems Magazine, accepted, December 2013.

[10] ———
Basic tracking using nonlinear 3D monostatic and bistatic measurements,
IEEE Aerospace and Electronic Systems Magazine, vol. 29, no. 8, Part II, pp. 4–53, Aug. 2014.

[11] ———
Simulating targets near a curved earth,
in *Proceedings of the 17th International Conference on Information Fusion*, Salamanca, Spain, 7–10 Jul. 2014.

[12] R. E. Deakin and M. N. Hunter
"Geodesics on an ellipsoid–Bessel's method,"
RMIT University, Melbourne, Australia, Tech. Rep., Oct. 2009. [Online]. Available: http://user.gs.rmit.edu.au/rod/files/publications/Geodesics%20-%20Bessel's%20method.pdf.

[13] Department of Defense
"Department of Defense world geodetic system 1984: Its definition and relationships with local geodetic systems,"
National Imagery and Mapping Agency, Tech. Rep. NIMA TR8350.2, Jun. 2004, third Edition, Amendment 2. [Online]. Available: http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf.

[14] "In the matter of the petition of Airbus SAS,"
Department of Transportation Federal Aviation Administration, 24 Mar. 2006, Regulatory Docket No. FAA-2005-20139. [Online]. Available: http://rgl.faa.gov/Regulatory_and_Guidance_Library/rgEX.nsf/0/ 9929ce16709cad0f8625713f00551e74/$FILE/8695.doc.

[15] M. P. Do Carmo
Differential Geometry of Curves and Surfaces.
Upper Saddle River: Prentice-Hall, Inc., 1976.

[16] C. Dougan
The parallel transport frame,
in *Game Programming Gems 2*, M. DeLoura, Ed. Charles River Media, 2001, ch. 2.5.

[17] Eclipse Aerospace, Inc. (2013, 12 Sep.)
The Eclipse 550 twin-engine jet.
[Online]. Available: http://www.eclipse.aero/550.php.

[18] M. E. Eshelby
"Pressure based separation in the upper airspace,"
College of Aeronautics, Cranfield Institute of technology, Cranfield, England, Tech. Rep. 8614, Feb. 1986. [Online]. Available: http://repository.tudelft.nl/assets/uuid:851bb984-e88c-4fc1-90c1-a98a11b040c1/Cranfield_College_of_Aeronautics_Report_No._8614-1986.pdf.

[19] G. T. Fairley and S. M. McGovern
Mathematical formulation of a fast-time geometric heading navigation model,
in *Proceedings of the 29th IEEE/AIAA Digital Avionics Systems Conference*, Salt Lake City, UT, 3–7 Oct. 2010, pp. 4.E.2-1–4.E.2-10.

[20] W. E. Featherstone and S. J. Claessens
Closed-form transformation between geodetic and ellipsoidal coordinates,
Studia Geophysica et Geodaetica, vol. 52, no. 1, pp. 1–18, Jan. 2008.

[21] ———
"Air traffic control,"
Federal Aviation Administration (United States), 9 Feb. 2012. [Online]. Available: http://www.faa.gov/documentLibrary/media/Order/ATC.pdf.

[22] M. L. Foucault
Sur les phénomènes d'orientation des corps tournants entraˆnés par un axe fixe à la surface de la terre.—nouveaux signes sensibles du mouvement diurne,
Comptes rendus hebdomadaires des séances de l'Académie des Sciences, pp. 424–427, 5 Jul. 1852. [Online]. Available: http://gallica.bnf.fr/ark:/12148/bpt6k2992n/f428.image.

[23] G. Golub and W. Kahan
Calculating the singular values and pseudo-inverse of a matrix,
Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis, vol. 2, no. 2, pp. 205–224, 1965.

[24] G. E. Golub and C. F. van Loan
Matrix Computations,
4th ed. Baltimore, MD: The Johns Hopkins University Press, 2013.

[25] Ø. Grøn. and A. Næss
Einstein's Theory: A Rigorous Introduction for the Mathematically Untrained.
New York: Springer, 2011.

[26] K. T. Guthrie
"Linear parameter-varying path following control of a small fixed wing unmanned aerial vehicle,"
Ph.D. dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA, 31 Jul. 2013.

[27] J. E. Hammett
"Evaluation of a proposed INS Kalman filter in a dynamic flight environment,"
Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, Dec. 1974.

[28] A. J. Hanson and H. Ma
"Parallel transport approach to curve framing,"
Department of Computer Science, Indiana University, Bloomington, IN, Tech. Rep. TR425, 11 Jan. 1995. [Online]. Available: ftp://cgi.soic.indiana.edu/pub/techreports/TR425.pdf.

[29] B. Hofmann-Wellenhof and H. Moritz
Physical Geodesy,
2nd ed. SpringerWienNewYork, 2006.

[30] IHS, Inc. (2013)
Ihs janes'.
[Online]. Available: https://janes.ihs.com.

[31] G. H. Kaplan
Determining the position and motion of a vessel from celestial observations,
Navigation, vol. 42, no. 4, pp. 633–650, Winter 1995–1996. [Online]. Available: http://aa.usno.navy.mil/publications/reports/GK_posmo.pdf.

[32] ———
A navigation solution involving changes to course and speed,
Navigation, vol. 43, no. 4, pp. 469–482, 1996. [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix =html&identifier=ADA434100.

[33] F. Karmali and M. Shelhamer
The dynamics of parabolic flight: Flight characteristics and passenger percepts,
Acta Astronaut, vol. 63, no. 5–6, pp. 594–602, Sep. 2008.

[34] C. F. F. Karney. (2013, 31 Aug.)
Addenda and errata for papers on geodesics.
[Online]. Available: http://geographiclib.sourceforge.net/geod-addenda.html.

[35] ———
Algorithms for geodesics,
Journal of Geodesy, vol. 87, no. 1, pp. 43–45, Jan. 2013.

[36] ———. (2013, 2 Jul.)
Geodesics on an ellipsoid of revolution.
Matlab Central. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/39108.

[37] ———. (2013, 30 Jun.)
GeographicLib.
[Online]. Available: http://geographiclib.sourceforge.net.

[38] L. A. Kivioja
Computation of geodetic direct and indirect problems by computers accumulating increments from geodetic line elements,
Bulletin Géodésique, vol. 99, no. 1, pp. 55–63, Mar. 1971.

[39] P. F. Krause and K. L. Flood
"Weather and climate extremes,"
U.S. Army Corps of Engineers, Topographic Engineering Center, Alexandria, VA, Tech. Rep. TEC-0099, Sep. 1997. [Online]. Available: http://www.dtic.mil/docs/citations/ADA346058.

[40] X. R. Li and V. P. Jilkov
Survey of maneuvering target tracking. part I: Dynamic models,
IEEE Transactions on Aerospace and Electronic Systems, vol. 39, no. 4, pp. 1333–1364, Oct. 2003.

[41] ———
Survey of maneuvering target tracking. part II: Motion models of ballistic and space targets,
IEEE Transactions on Aerospace and Electronic Systems, vol. 46, no. 1, pp. 96–119, Jan. 2010.

[42] X. Li and H.-J. Götze
Ellipsoids, geoid, gravity, geodesy, and geophysics,
Geophysics, vol. 66, no. 6, pp. 1660–1668, Nov.–Dec. 2001, an errata wash published in Volume 67, Number 3.

[43] L. Lin, T. Kirubarajan, and Y. Bar-Shalom
3-D track initiation in clutter using 2-D radar measurements,
IEEE Transactions on Aerospace and Electronic Systems, vol. 38, no. 4, pp. 1434–1441, Oct. 2002.

[44] B. Luong. (2012, 18 Oct.)
Plot Earth. Matlab Central.
[Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/25048-plot-earth.

[45] W. A. Mair and D. L. Birdsall
Aircraft Performance,
ser. Cambridge Aerospace Series 5. Cambridge, UK: Cambridge University Press, 1992.

[46] P. S. Maybeck
"Wander azimuth implementation algorithm for a strapdown inertial system,"
Air Force Avionics Laboratory, Wright-Patterson AFB, OH, Tech. Rep. AFFDL-TR-73-80, Oct. 1973.

[47] R. A. Minzer
The 1976 standard atmosphere and its relationship to earlier standards,
Reviews of Geophysics and Space Physics, vol. 15, no. 3, pp. 375–384, Aug. 1977.

[48] P. J. Mohr, B. N. Taylor, and D. B. Newell
CODATA recommended values of the fundamental physical constants: 2010,
Reviews of Modern Physics, vol. 84, no. 4, pp. 1527–1605, Oct.–Dec. 2012. [Online]. Available: http://physics.nist.gov/cuu/Constants/Preprints/lsa2010.pdf.

[49] D. J. Mook and I.-M. Shyu
Nonlinear aircraft tracking filter utilizing control variable estimation,
Journal of Guidance, Control, and Dynamics, vol. 15, no. 1, pp. 228–237, Jan.–Feb. 1992.

[50] S. H. Musick
"PROFGEN: A computer program for generating flight profiles,"
Air Force Avionics Laboratory, Wright-Patterson AFB, OH, Tech. Rep. AFAL-TR-76-247, Nov. 1976. [Online]. Available: http://www.dtic.mil/docs/citations/ADA034993.

[51] National Geospatial Intelligence Agency. (2012, 1 Jul.)
WGS 84 G1647 geodetic control network upgrade for areas of white sands missile range and Holloman AFB, NM.
[Online]. Available: http://www.wsmr.army.mil/pdf/G1674-Upgrade-01Jul2012.pdf.

[52] ———. (2013, 29 Apr.)
Office of geomatics: World geodetic system 1984 (WGS 84).
[Online]. Available: http://earth-info.nga.mil/GandG/wgs84/.

[53] NGA Office of Geomatics. (2014, 16 May)
WGS 84 and the web mercator projection.
National Geospatial Intelligence Agency. A slide presentation on the linked page. [Online]. Available: http://earth-info.nga.mil/GandG/wgs84/web_mercator/index.html.

[54] N. Oceanic and A. Administration
"U.S. standard atmosphere 1976,"
Washington, D.C., Oct. 1976. [Online]. Available: http://www.dtic.mil/docs/citations/ADA035728.

[55] N. A. Office
Astronomical Almanac for the Year 2013 and its Companion, the Astronomical Almanac Online.
United States Defense Department, Navy, Nautical Almanac Office, 2013.

[56] E. J. Ohlmeyer
Root-mean-square miss distance of proportional navigation missile against sinusoidal target,
Journal of Guidance, Control, and Dynamics, vol. 19, no. 3, pp. 563–568, May–Jun. 1996.

[57] G. Panou, D. Delikaraoglou, and R. Korakitis
Solving the geodesics on the ellipsoid as a boundary value problem,
Journal of Geodetic Science, vol. 3, no. 1, pp. 40–47, Mar. 2013.

[58] G. Petit and B. Luzum
IERS Conventions (2010),
International Earth Rotation and Reference Systems Service Std. 36, 2010.

[59] D. H. Platus
Ballistic re-entry vehicle flight dynamics,
Journal of Guidance, Control, and Dynamics, vol. 5, no. 1, pp. 4–16, Jan.–Feb. 1982.

[60] P. Pokorny
Geodesics revisited,
Chaotic Modeling and Simulation International Journal of Nonlinear Science, pp. 281–298, Jan. 2012. [Online]. Available: http://www.cmsim.eu/papers_pdf/january_2012_papers/25_CMSIM_2012_Pokorny_1_281-298.pdf.

[61] R. H. Rapp
"Geometric geodesy, part I,"
Ohio State University Department of Geodetic Science and Surveying, Tech. Rep., Apr. 1991. [Online]. Available: http://hdl.handle.net/1811/24333.

[62] M. Sankowski
Reference model of aircraft movements in geodetic coordinates,
in *Proceedings of the International radar Symposium*, Leipzig, Germany, 7–9 Sep. 2011, pp. 874–879.

[63] ———
Target course estimation in tracking filters using local Cartesian coordinates,
in *Proceedings of the 19th International Radar Symposium*, Warsaw, Poland, 23–25 May 2012, pp. 515–520.

[64] L. E. Sjöberg and M. Shirazian
Solving the direct and inverse geodetic problems on the ellipsoid by numerical integration,
Journal of Surveying Engineering, vol. 138, no. 1, pp. 9–16, Feb. 2012.

[65] I. Sofair
Improved method for calculating exact geodetic latitude and altitude,
Journal of Guidance, Control, and Dynamics, vol. 20, no. 4, pp. 824–826, Jul.–Aug. 1997.

[66] ———
Improved method for calculating exact geodetic latitude and altitude revisited,
Journal of Guidance, Control, and Dynamics, vol. 23, no. 2, p. 369, Mar. 2000.

[67] J. Souchay and R. Gaume. (2013, 3 Sep.)
ICRS product center.
[Online]. Available: http://hpiers.obspm.fr/icrs-pc/.

[68] R. Stöckli, E. Vermote, N. Saleous, R. Simmon, and D. Herring. (2005, 22 Aug.)
The blue marble next generation—a true color earth dataset including seasonal dynamics from MODIS.
[Online]. Available: http://eoimages.gsfc.nasa.gov/images/imagerecords/73000/73751/readme.pdf.

[69] K. R. Symon
Mechanics,
3rd ed. Reading, MA: Addison-Wesley, 1971.

[70] D. Teagle and B. Ildefonse
Journey to the mantle of the earth,
Nature, vol. 471, pp. 437–439, 24 Mar. 2011.

[71] The Visible Earth. (2013, 11 Sep.)
The blue marble: Land surface, ocean color, sea ice and clouds.
NASA Goddard Space Flight Center. [Online]. Available: http://visibleearth.nasa.gov/view.php?id=57735.

[72] ———. (2013, 11 Sep.)
NASA visible Earth: July, blue marble next generation w/ topography and bathymetry.
NASA Goddard Space Flight Center. [Online]. Available: http://visibleearth.nasa.gov/view.php?id=73751.

[73] J. A. Thelander
"Aircraft motion analysis,"
Air Force Flight Dynamics Laboratory, Tech. Rep. FLD-TDR-64-70, Mar. 1965. [Online]. Available: http://www.dtic.mil/docs/citations/AD0617354.

[74] C. M. Thomas and W. E. Featherstone
Validation of Vincenty's formulas for the geodesic using a new fourth-order extension of Kivioja's formula,
Journal of Surveying Engineering, pp. 20–26, Feb. 2005.

[75] United States Air Force. (2007, 8 Oct.)
F-16 fighting falcon.
[Online]. Available: http://www.af.mil/AboutUs/FactSheets/Display/tabid/224/Article/104505/f-16-fighting-falcon.aspx.

[76] United States Geological Survey. (2006, 2 Feb.)
Mauna Loa Volcano, Hawaii.
[Online]. Available: http://hvo.wr.usgs.gov/maunaloa/.

[77] T. Vincenty
Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations,
Survey Review, vol. 23, no. 176, pp. 88–93, Apr. 1975. [Online]. Available: http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf.

[78] M. Vinnins
"PROFGEN: An aircraft flight profile generation program,"
Research and Development Branch, Department of national Defence, Ottawa, Canada, Tech. Rep. 78-14, Sep. 1978.

[79] R. Williams
The great ellipse on the surface of the spheroid,
Journal of Navigation, vol. 49, no. 2, pp. 229–234, May 1996.

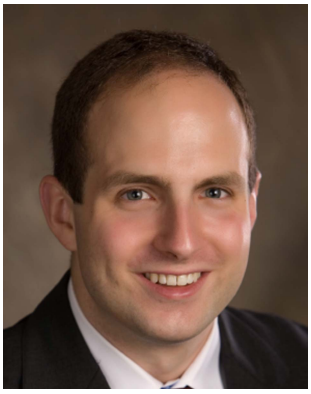
[80] Wolfram, Inc. (2013, 10 Sep.)
Numerical solution of partial differential equations—Mathematica 9 documentation.
[Online]. Available: http://reference.wolfram.com/mathematica/tutorial/NDSolvePDE.html.

[81] M. Yeddanapudi, Y. Bar-Shalom, and K. Pattipati
IMM estimation for multitarget-multisensor air traffic surveillance,
Proceedings of the IEEE, vol. 85, no. 1, pp. 80–94, Jan. 1997.

[82] P. Zarchan
Tactical and Strategic Missile Guidance,
5th ed. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2007.

[83] P. Zarchan and H. Musoff
Fundamentals of Kalman Filtering: A Practical Approach,
3rd ed. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2009.

[84] P. H. Zipfel
Modeling and Simulation of Aerospace Vehicle Dynamics,
2nd ed. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2007.



David Frederic Crouse received B.S., M.S., and Ph.D. degrees in Electrical Engineering in 2005, 2008, and 2011 from the University of Connecticut (UCONN). He also received a B.A. degree in German from UCONN for which he spent a year at the Ruprecht-Karls Universität in Heidelberg, Germany.

He is currently employed at the Naval Research Laboratory in Washington, D.C. and serves as an associate editor at the IEEE Aerospace and Electronic Systems Magazine. His interests lie in the areas of stochastic signal processing and tracking.

Road Network Identification by means of the Hough Transform with Uncertainty Analysis

ERIC SALERNO
NAGAVENKAT ADURTHI
TARUNRAJ SINGH
PUNEET SINGLA
ADNAN BUBALO
MARIA CORNACCHIA
MARK ALFORD
ERIC JONES

The focus of this paper is on the use of ground target kinematics to estimate the underlying road network on which the vehicles are assumed to be travelling. Assuming that the road network can be represented as an amalgamation of straight line segments, a Hough transform approach is used to identify portion of road which correspond to straight line segments. Since multiple tracks can be associated with one segment of the road and since the track estimates are inherently uncertain, an iterative approach is presented to identify a parametric representation of the line segments of the roads using the total least squares cost function. Cramér-Rao bounds are identified to characterize the bounds on the uncertainty associated with the proposed approach. A complex dataset which include multiple tracks is used to illustrate the ability of the proposed algorithm to identify the underlying road network and characterize the uncertainty associated with the parametric estimate of the road.

Manuscript received December 11, 2013; revised November 4, 2014 and February 24, 2015; released for publication March 1, 2015.

Refereeing of this contribution was handled by Robert Lynch.

Authors' addresses: E. Salerno, N. Adurthi, T. Singh, P. Singla, Dept. of Mech. & Aero. Eng., University of Buffalo, Buffalo, NY 14260 (e-mail: esalerno@buffalo.edu, nagavenk@buffalo.edu, tsingh@buffalo.edu, psingla@buffalo.edu). A. Bubalo, M. Cornacchia, M. Alford, E. Jones, Information Directorate, Air Force Research Laboratory, Rome, NY 13441 (e-mail: Adnan.Bubalo@rl.af.mil).

1557-6418/15/$17.00

1 INTRODUCTION

Automatic cartographic feature extraction has been the goal for road network identification given the tremendous growth in automobiles with navigation systems, and the interest in driverless cars. In post hurricane disaster scenarios when bridges and roads might be washed out, there is a need to rapidly update road network databases for logistics. In regions of conflict or deserts, there is a need to develop road maps to identify safe travel routes. Precise road networks can also be used to enhance the performance of ground vehicle trackers by restricting the motion of the target to the road network.

Synthetic Aperture Radar (SAR) and Ground Moving Target Indicator (GMTI) data is often processed and analyzed to produce such networks. SAR produces images of varying intensity which can be processed to separate buildings, roads, and terrain. However, SAR is only able to detect prominent existing features [1]. For example, SAR will only detect a road if there is a distinct outline of such a path. GMTI on the other hand tracks moving targets and relays the latitude and longitude coordinates as well as the kinematic information. The disadvantage of GMTI, however, is the necessity of a moving target. Should the target stop or be obstructed in any way from the sensors, the tracker will lose the target for the duration of the obstructions [2]. Many of the currently available algorithms rely on information from pre-existing road maps, however, in many scenarios the availability of this a priori information is limited and inaccurate. In some situations there are no existing road maps, such as in times of conflict in desert regions. Therefore the need for an algorithm which can accurately estimate road networks in a timely manner is of great importance. Furthermore, there is a lack of a real quantifiable measure of the accuracy of the extracted road estimates. Several available algorithms use a "completeness" and "correctness" measure, which is a comparison of the extracted road network and the actual network [3, 4, 5, 6]. However, as previously stated in many situations there are no available true networks (i.e., desert regions) so the metrics used to evaluate the performance of algorithms are not relevant.

Hu, Razdan, Femiani, Cui, and Wonka use a spoke and wheel method in order to determine the footprints [5]. These footprints or polygons, are road segments which span in any direction and terminate when the intensity of the spoke or line segment falls below a threshold. Then a toe-finding algorithm is used to determine the number of branches in the footprint. In this portion of the algorithm, if the angle between two branches is less than 45 degrees they are merged, in some cases this will eliminate Y-shaped intersections and parallel roads. Instead of using centerlines to approximate the road network they utilize inscribed lines to define the road structure. Finally the road network

is trimmed to eliminate noisy data which might contribute to false roads, and also possible gaps between roads due to obstructions. This algorithm relies heavily on pre-existing information pertaining to roads such as the possible shapes of intersections, road widths, and angle of roads at these intersections.

Tupin, Maitre, Mangin, Nicolas, and Pechersky first identify linear features in the data and then separate the true segments by using a Markov random field (MRF) [1]. Two different line detectors are used to identify candidate road segments and then the results of these two detectors are fused together. With the identification of the candidates, the MRF-based model fills in large gaps and removes the false detections. This MRF-based model relies on a priori information of the road network being developed. The assumption is made that all roads lead to an initial starting point which can limit the accuracy of the overall algorithm.

Shackelford and Davis use a pixel-based fuzzy classifier and an object based classification approach to identify the road networks [6]. Skeletonization is an approach in which the image is thinned or eroded away until only the essential lines remain. This method produces a large amount of false positives in the road network. The second approach utilized is an iterative approach in which the longest roads are initially identified and then shorter and shorter segments are added throughout the algorithm. The second algorithm proves to be much better than skeletonization, however, the "completeness" measure of the road network has decreased in both the urban and suburban scenarios for the second algorithm when compared with the first.

Sklarz, Novoselsky, and Dorfan focus on the fusion of linear segments and curves based on a unified entity approach rather than a single pixel based approach [7]. The road network can either begin as a blank slate or already contain roads. As new tracks become available the curve is associated with an existing curve if one exists otherwise a new segment of road is added to the existing network. Should a track already exist, the new track is cropped into segments to match with the relative endpoints of the pre-existing road segments. The optimization of the curve fusion's computational complexity increases drastically as the curves are discretized into more finite segments thus limiting the potential of the algorithm.

Koch, Koller, and Ulmke utilize a Multiple Hypothesis Tracking (MHT) algorithm, which consists of target track extraction, prediction, filtering, track maintenance, and retrodiction [2]. It is assumed that the posterior probability density function is Gaussian, thus the Kalman filter is utilized since the algorithm breaks the road up into linear segments. The pruning removes any segments, which have a weight smaller than a threshold, depending on the threshold this could cause some issues with removing actual tracks. The merging depends on segments having similar state vectors and covariances.

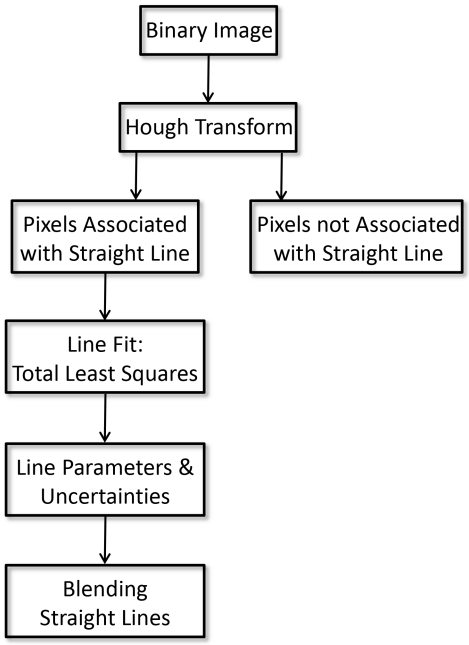


Fig. 1. Algorithm flow chart.

In this paper, a method for developing road networks and characterizing the uncertainty in these estimates is developed. It is assumed that road networks can be broken down into piecewise linear segments. Figure 1 illustrates the sequence of processing of data to generate a parameterization of the road network with the associated uncertainties. The initial processing of the data is done by creating a binary image of the track data and extracting possible line segments using the Hough transform. This is followed by the clustering of the data associated with the identified straight lines. Since the Hough transform does not provide a measure of uncertainty, the Total Least Squares approach is implemented and the Cramér Rao lower bounds is derived from this maximum likelihood estimate. The Total Least Squares solution allows for an iterative estimate, which is updated in time as additional measurements become available. The Least Squares solution will be used as an initial estimate for the recursive Total Least Squares algorithm. Once the data collection has been terminated the individual line segments can be merged, extended or trimmed, and blended to produce a more complete road network.

Section 2 details the derivation of the Hough transform for identifying straight line edges in an image. The recursive Total Least Squares solution is presented in Section 3 along with the corresponding uncertainty analysis and derivations. In Section 4 the results of the algorithms outlined are applied to a data set and we conclude the paper with suggestions for future research in Section 5.

2 HOUGH TRANSFORM

The Hough transform is a well studied feature extraction technique that has been used extensively in

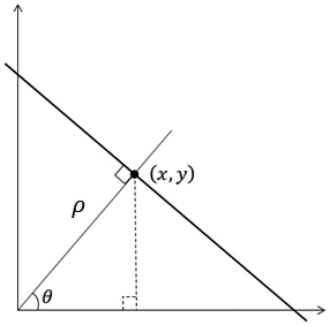


Fig. 2. Parameter identification.

image analysis [9]. This transform requires the image to be binary in nature, where white pixels correspond to ones and black pixels correspond to zeros. The derivation of the Hough transform requires basic trigonometric identities. Suppose we have a line oriented as shown in Figure 2 then by defining the parameters ρ and θ we can derive the Hough transform. The perpendicular distance from the origin to a line is denoted by ρ. The angle that this distance vector makes with the x-axis is θ. We note the definitions of the cosine and sine functions:

$$\cos\theta = \frac{x}{\rho} \qquad \sin\theta = \frac{y}{\rho}. \tag{1}$$

Algebraic substitution of a single cosine and sine term in the Pythagorean trigonometric identity leads to the equation:

$$\frac{x}{\rho}\cos\theta + \frac{y}{\rho}\sin\theta = 1. \tag{2}$$

It is now a simple matter to rearrange Equation (2) to obtain the conventional form of the Hough transform as given by the equation:

$$\rho = x\cos\theta + y\sin\theta. \tag{3}$$

Now if Equation (3) is rearranged into a slope-intercept form

$$y = -\frac{\cos\theta}{\sin\theta}x + \frac{\rho}{\sin\theta} \tag{4}$$

we can infer that when θ approaches zero degrees, corresponding to a vertical line, the slope tends to infinity, resulting in an poor parameterization of the line. However, using Equation (3), we avoid this problem.

The principle concept of the Hough transform in the line identification algorithm can be stated as the following: **if two points are collinear then they share a pair (ρ,θ) of commonality in the Hough space.** However, in order to determine this common pair, a Hough matrix must be constructed, this is done by iterating over a θ range of -90 to 90 degrees for each white pixel, which corresponds to the (x,y) coordinates, in the image. Recall that θ is the angle the ρ vector makes with the x-axis. We can imagine that an arbitrary number of lines (black solid lines) pass through each coordinate shown by the solid black lines in Figure 3, which are associated with a (ρ,θ) given by the normal lines passing through the origin shown by the dashed lines. The solid blue line is the line of interest and Figure 3 illustrates two points which lie on this line. Note that both these points illustrated by the solid circle have a coincident (ρ,θ) pair, which parameterize the dashed blue line.

For each white pixel of the binary image with a coordinate x_i, y_i determine the parameter

$$\rho(\theta) = x_i\cos(\theta) + y_i\sin(\theta) \tag{5}$$

which corresponds to a sinusoid in the (ρ,θ) space. Determine the (ρ,θ) pair for every white pixel of the binary images, round it to the closest discretized value of (ρ,θ), and augment the appropriate indices of an array called the accumulator. Since every point on a line will share a unique (ρ,θ) pair which corresponds to a line normal to the line of interest passing through the origin, the accumulator bin with the highest count will correspond to parameters of a straight line. Figure 4 illustrates the mapping of the Hough accumulator with the white pixels indicating a higher count compared to the black pixels. The point highlighted by the square corresponds to the (ρ,θ) combination associated with a straight line.

Matlab's image processing toolbox is used to determine the Hough transform, identify the peaks which corresponds to the straight lines, identify the points on the images corresponding to the identified line segments which are subsequently used to characterize the uncertainty of the identified lines.

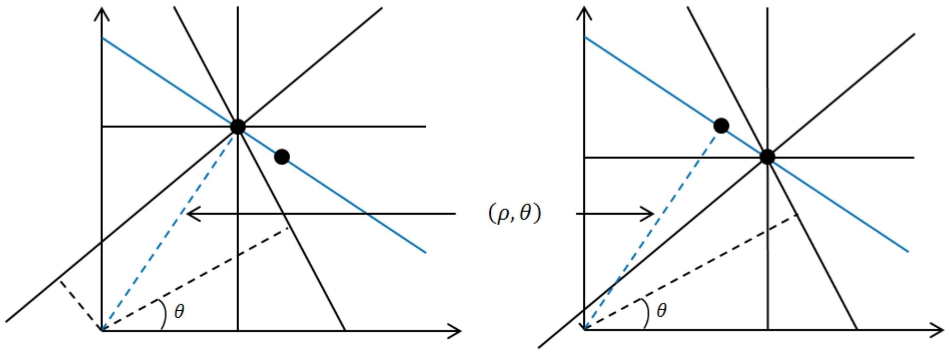


Fig. 3. Collinear points.

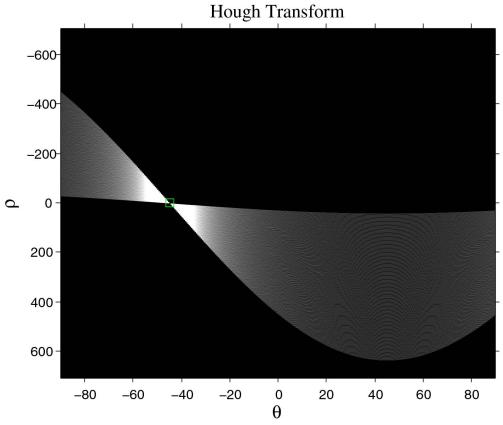


Fig. 4. Hough transform.

3 MAXIMUM LIKELIHOOD ESTIMATORS

In this section we will derive the solutions for the Total Least Squares (TLS) to estimate the parameters of a straight line. In this formulation, the measurement noise covariance matrix is allowed to be populated, i.e., the dependent and independent variables are noisy and can be correlated. The Hough transform presented in Section 2 will be utilized to identify groups of nearly collinear points in an image. A best fit line can be obtained using the TLS solution since there is noise in both the x and y directions. The TLS solution, which will be derived, does not have a closed form solution and therefore requires an initial estimate to initiate the solver which minimizes the normal distance of the measurements from the line. This initial estimate can be given by a transformed version of the Hough coefficients (i.e., transform from ρ and θ to slope, m and intercept b), however if θ is zero degrees then we would obtain an infinite slope, which forces the TLS solution to diverge. Therefore, rather than use the Hough transform coefficients transformed to the appropriate slope and intercept we will simply utilize a Least Squares estimate as the initial guess for the TLS algorithm.

3.1 Total Least Squares

Consider the problem of a straight line fit where the true model is given by the equation:

$$y_i = mx_i + c \tag{6}$$

where y_i and x_i are the true dependent and independent coordinates. m and c correspond to the slope and ordinate intercept of the line. Assuming that the measurement of both y_i and x_i are noisy, the measurement equations and the corresponding pdf of the noise are given by the equations:

$$\tilde{x}_i = x_i + \nu_x \tag{7}$$

$$\tilde{y}_i = y_i + \nu_y \tag{8}$$

$$p(\nu_1,\nu_2) = \mathcal{N}\left(\begin{bmatrix}\nu_1\\ \nu_2\end{bmatrix} : \begin{bmatrix}0\\ 0\end{bmatrix}, \begin{bmatrix}\sigma_{xx}^2 & \sigma_{xy}\\ \sigma_{xy} & \sigma_{yy}^2\end{bmatrix}\right) \tag{9}$$

permitting the measurement noise in x_i and y_i to be correlated. Note that σ_{xx}^2 and σ_{yy}^2 denote the variance of ν_x and ν_y respectively and σ_{xy} represent the cross-covariance of ν_x and ν_y. The notation $\mathcal{N}(\nu : \mu, \Sigma)$ is a Gaussian probability density function(pdf) for the random vector ν with mean μ and covariance Σ. Identifying the parameters of a line when provided with n measurements, the likelihood function for the measurements is given by the equation:

$$p = \prod_{i=1}^{n} p_i \tag{10}$$

$$p_i = p\left(\begin{bmatrix}\tilde{x}_i\\ \tilde{y}_i\end{bmatrix} \mid m, c, x_1, x_2, \ldots, x_n\right)$$

$$= \mathcal{N}\left(\begin{bmatrix}\tilde{x}_i - x_i\\ \tilde{y}_i - y_i\end{bmatrix} : \begin{bmatrix}0\\ 0\end{bmatrix}, \begin{bmatrix}\sigma_{xx}^2 & \sigma_{xy}\\ \sigma_{xy} & \sigma_{yy}^2\end{bmatrix}\right) \tag{11}$$

$$= \mathcal{N}\left(\begin{bmatrix}\tilde{x}_i - x_i\\ \tilde{y}_i - mx_i - c\end{bmatrix} : \begin{bmatrix}0\\ 0\end{bmatrix}, \begin{bmatrix}\sigma_{xx}^2 & \sigma_{xy}\\ \sigma_{xy} & \sigma_{yy}^2\end{bmatrix}\right) \tag{12}$$

$$= \mathcal{N}\left(\begin{bmatrix}\tilde{x}_i\\ \tilde{y}_i\end{bmatrix} : \begin{bmatrix}x_i\\ mx_i + c\end{bmatrix}, \begin{bmatrix}\sigma_{xx}^2 & \sigma_{xy}\\ \sigma_{xy} & \sigma_{yy}^2\end{bmatrix}\right). \tag{13}$$

The Log-Likelihood to be maximized with respect to the free variables m, c and x_i where $i = 1, 2, \ldots, n$ is

$$\ln(p) = -n\ln\left(2\pi\sqrt{\sigma_{xx}^2\sigma_{yy}^2 - \sigma_{xy}^2}\right) - \frac{1}{2(\sigma_{xx}^2\sigma_{yy}^2 - \sigma_{xy}^2)}\sum_{i=1}^{n}(\sigma_{yy}^2(\tilde{x}_i - x_i)^2 - 2\sigma_{xy}(\tilde{x}_i - x_i)(\tilde{y}_i - mx_i - c) + \sigma_{xx}^2(\tilde{y}_i - mx_i - c)^2). \tag{14}$$

There are $n + 2$ variables: m, c, and all the n abscissas x_i. Differentiate the log-likelihood with respect to the variables and solve the $n + 2$ equations:

$$\frac{\partial \ln(p)}{\partial m} = -\frac{1}{2(\sigma_{xx}^2\sigma_{yy}^2 - \sigma_{xy}^2)} \times \sum_{i=1}^{n}(2\sigma_{xy}(\tilde{x}_i - x_i)x_i - 2\sigma_{xx}^2(\tilde{y}_i - mx_i - c)x_i) = 0 \tag{15}$$

$$\frac{\partial \ln(p)}{\partial c} = -\frac{1}{2(\sigma_{xx}^2\sigma_{yy}^2 - \sigma_{xy}^2)} \times \sum_{i=1}^{n}(2\sigma_{xy}(\tilde{x}_i - x_i) - 2\sigma_{xx}^2(\tilde{y}_i - mx_i - c)) = 0 \tag{16}$$

$$\frac{\partial \ln(p)}{\partial x_i} = -\frac{1}{2(\sigma_{xx}^2\sigma_{yy}^2 - \sigma_{xy}^2)} \times (-2\sigma_{yy}^2(\tilde{x}_i - x_i) + 2\sigma_{xy}(\tilde{y}_i - mx_i - c) + 2\sigma_{xy}(\tilde{x}_i - x_i)m - 2\sigma_{xx}^2(\tilde{y}_i - mx_i - c)m) = 0. \tag{17}$$

Solving for the x_i from equation (17) leads to the equation:

$$x_i = \frac{c(-m\sigma_{xx}^2 + \sigma_{xy}) - m\sigma_{xy}\tilde{x}_i + \sigma_{yy}^2\tilde{x}_i + m\sigma_{xx}^2\tilde{y}_i - \sigma_{xy}\tilde{y}_i}{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2\sigma_{xy}m}. \tag{18}$$

Substituting the expression for x_i back into the log likelihood function $\ln(p)$ of equation (14) and on simplification, the final cost function in terms of m and c is:

$$\ln(p) = -n\ln\left(2\pi\sqrt{\sigma_{xx}^2\sigma_{yy}^2 - \sigma_{xy}^2}\right) - \frac{1}{2}\sum_{i=1}^{n}\frac{(\tilde{y}_i - m\tilde{x}_i - c)^2}{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2\sigma_{xy}m} \tag{19}$$

or the problem can be stated as

$$\min_{m,c}\sum_{i=1}^{n}\frac{(\tilde{y}_i - m\tilde{x}_i - c)^2}{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2\sigma_{xy}m}. \tag{20}$$

Additionally, the noise covariance parameters can be specific to each measurement, i.e., σ_{xx}, σ_{yy}, and σ_{xy} can be $\sigma_{xx}^{(i)}, \sigma_{yy}^{(i)}$ and $\sigma_{xy}^{(i)}$ respectively corresponding to the measurement $(\tilde{x}_i, \tilde{y}_i)$. It can be observed that, when the noise in the variables is uncorrelated ($\sigma_{xy} = 0$) and of equal variance($\sigma_{xx} = \sigma_{yy} = \sigma$), the standard Total least square problem is recovered, which can be identified as the perpendicular distance of $(\tilde{x}_i, \tilde{y}_i)$ to the line $y = mx + c$, and

$$\min_{m,c}\sum_{i=1}^{n}\frac{(\tilde{y}_i - m\tilde{x}_i - c)^2}{1 + m^2}$$

is the resulting cost function to be minimized.

3.2 Alternate Equivalent Formulation

The algebraic manipulation involved in arriving at the cost function of (20) can be avoided altogether. Consider the truth model where we represent the truth (x_i, y_i) in terms of the noisy measurement $(\tilde{x}_i, \tilde{y}_i)$ as:

$$y_i = mx_i + c \tag{21}$$

$$x_i = \tilde{x}_i - \nu_x \tag{22}$$

$$y_i = \tilde{y}_i - \nu_y. \tag{23}$$

Substituting (22) and (23) into (21) results in the equation:

$$\tilde{y}_i - m\tilde{x}_i - c = \nu_y - m\nu_x. \tag{24}$$

Define ν' as:

$$\nu' = \frac{\nu_y - m\nu_x}{\sqrt{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2m\sigma_{xy}}} \tag{25}$$

whose mean and variance of random variable ν' conditioned on m are given by the equations:

$$E[\nu'] = E\left[\frac{\nu_y - m\nu_x}{\sqrt{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2m\sigma_{xy}}}\right] = \frac{E[\nu_y] - mE[\nu_x]}{\sqrt{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2m\sigma_{xy}}} = 0 \tag{26}$$

$$E[\nu'^2] = E\left[\frac{(\nu_y - m\nu_x)^2}{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2m\sigma_{xy}}\right] = \frac{E[\nu_y^2] + m^2E[\nu_x^2] - 2mE[\nu_x\nu_y]}{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2m\sigma_{xy}} = 1. \tag{27}$$

The random variable ν' has a Gaussian distribution given by the equation:

$$p(\nu' \mid m) = \mathcal{N}(\nu' : 0, 1). \tag{28}$$

But from equations (24) and (25), one has

$$\nu' = \frac{\tilde{y}_i - m\tilde{x}_i - c}{\sqrt{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2m\sigma_{xy}}}.$$

Hence, for a set of measurements $(\tilde{x}_i, \tilde{y}_i)$, the Maximum Likelihood Estimator (MLE) requires maximizing the function given by Equation (10), which can be rewritten as:

$$p_i = \mathcal{N}\left(\frac{\tilde{y}_i - m\tilde{x}_i - c}{\sqrt{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2m\sigma_{xy}}} : 0, 1\right). \tag{29}$$

The negative of the Log-Likelihood to be minimized with respect to the free variables m and c is

$$J = -\ln(p) = \sum_{i=1}^{n}\frac{(\tilde{y}_i - m\tilde{x}_i - c)^2}{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2m\sigma_{xy}}. \tag{30}$$

Notice that there are no x_i variables in the cost function. Differentiating the cost function J only with the variables m and c, we arrive at the gradient constraint equations:

$$\frac{\partial J}{\partial m} = \sum_{i=1}^{n}\left(\frac{2(\tilde{y}_i - m\tilde{x}_i - c)(-\tilde{x}_i)}{\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2m\sigma_{xy}} - \frac{(\tilde{y}_i - m\tilde{x}_i - c)^2(2\sigma_{xx}^2m - 2\sigma_{xy})}{(\sigma_{yy}^2 + \sigma_{xx}^2m^2 - 2m\sigma_{xy})^2}\right) = 0 \tag{31}$$

$$\frac{\partial J}{\partial c} = \sum_{i=1}^{n}(-2(\tilde{y}_i - m\tilde{x}_i - c)) = 0 \tag{32}$$

which is nonlinear in m and c and can be solved numerically. One can use the solution of the least squares problem to initialize the nonlinear solver. One can also

estimate the most likely value of the variable x_i using the equation:

$$x_i = \frac{c(-m\sigma_{xx}^2 + \sigma_{xy}) - m\sigma_{xy}\tilde{x}_i + \sigma_{yy}^2\tilde{x}_i + m\sigma_{xx}^2\tilde{y}_i - \sigma_{xy}\tilde{y}_i}{\sigma_{yy}^2 + \sigma_{xx}^2 m^2 - 2\sigma_{xy}m} \tag{33}$$

from the resulting solution for the slope m and intercept c.

3.3 Geometric Interpretation

Consider the case of estimating the parameters of a straight line where the $\tilde{x}_i$ and $\tilde{y}_i$ measurements are contaminated by isotropic noise. Figure 5 illustrates the true x_i and y_i and the corresponding contaminated $\tilde{x}_i$ and $\tilde{y}_i$ data. Assuming the parameters $\sigma_{xx} = \sigma_{yy} = 1$ and σ_{xy}=0, the MLE cost function given by Equation (30) reduces to:

$$J = \sum_{i=1}^{n} \frac{(\tilde{y}_i - m\tilde{x}_i - c)^2}{1 + m^2} \tag{34}$$

which is referred to as geometric distance which is a gradient weighted algebraic distance. This error referred to as the Sampson error is the first order approximation of the geometric distance of the point from the curve.

In Figure 5, one can easily illustrate the line connecting the points $(\tilde{x}_i, \tilde{y}_i)$ along the normal to the true line has a length given by equation $(\tilde{y}_i - m\tilde{x}_i - c)^2/(1 + m^2)$, illustrating that the total least squares cost function corresponds to minimizing the geometric distance. Note that the least squares problem minimizes the distance $(\tilde{y}_i - m\tilde{x}_i - c)^2$ which is referred to as the algebraic distance and corresponds to the distance along an oblique projection.

3.4 Least Squares

The least squares problem is a special case of the total least squares where the independent variable x_i is not random, i.e., $\sigma_{xx} = 0$. This results in the cost function given by Equation (30) reducing to:

$$J = -\ln(p) = \sum_{i=1}^{n} \frac{(\tilde{y}_i - mx_i - c)^2}{\sigma_{yy}^2} \tag{35}$$

which has a closed form solution for m and c:

$$\begin{Bmatrix} m \\ c \end{Bmatrix} = \begin{Bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{Bmatrix}^{-1} \begin{Bmatrix} \sum_{i=1}^{n} x_i\tilde{y}_i \\ \sum_{i=1}^{n} \tilde{y}_i \end{Bmatrix}. \tag{36}$$

3.5 Uncertainty Analysis

In this section, we will present the derivation for the Cramér Rao lower bounds which provides a measure of uncertainty in the coefficients of the linear fit. The derivation is based on a fully populated covariance matrix and the bounds are estimated by the inverse of the Fisher Information matrix. The derivations are accompanied by a Monte Carlo simulation to verify the convergence properties of the solutions. For the straight

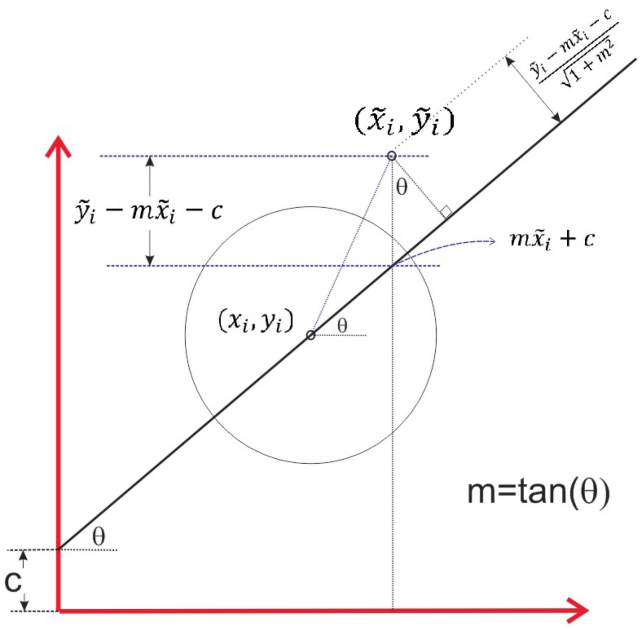


Fig. 5. Total least squares minimization.

line we have a total of 2+n unknown parameters, the slope m, the intercept c, and the true values of $x : x_t$.

After solving for the optimal estimates for the slope $\hat{m}$ and y-intercept $\hat{c}$, the optimal estimate for $\hat{x}_i$ is given by Equation (33). Assuming the parameters $\sigma_{xx} = \sigma_{yy} = 1$ and σ_{xy}=0, Equation (33) reduces to:

$$\hat{x}_i = \frac{-\hat{c}\hat{m} + \tilde{x}_i + \hat{m}\tilde{y}_i}{1 + \hat{m}^2} \tag{37}$$

$$\Rightarrow \hat{x}_i = \frac{-\hat{c}\hat{m} + \tilde{x}_i + \tilde{x}_i\hat{m}^2 - \tilde{x}_i\hat{m}^2 + \hat{m}\tilde{y}_i}{1 + \hat{m}^2} \tag{38}$$

$$\Rightarrow \hat{x}_i = \tilde{x}_i + \frac{\hat{m}e_i}{1 + \hat{m}^2} \tag{39}$$

where

$$e_i = \tilde{y}_i - \hat{m}\tilde{x}_i - \hat{c} \tag{40}$$

is the error associated with the measurement $\tilde{y}_i$ and the estimate $\hat{y}_i$ at the measured x coordinate $\tilde{x}_i$. Figure 6 illustrates the optimal estimates $(\hat{x}_i, \hat{y}_i)$, which results in the equation:

$$\hat{y}_i = \tilde{y}_i - e_i - \hat{m}(\tilde{x}_i - \hat{x}_i) \tag{41}$$

$$\hat{y}_i = \tilde{y}_i - \frac{e_i}{1 + \hat{m}^2}. \tag{42}$$

It can be shown that the generalized solutions for the estimated values for x_{t_i} and y_{t_i} are:

$$\hat{x}_i^2 = \tilde{x}_i + \frac{(\hat{m}\sigma_{x_ix_i}^2 - \sigma_{x_iy_i})e_i}{\hat{m}^2\sigma_{x_ix_i}^2 - 2\hat{m}\sigma_{x_iy_i} + \sigma_{y_iy_i}^2} \tag{43}$$

$$\hat{y}_i^2 = \tilde{y}_i + \frac{(\hat{m}\sigma_{x_iy_i} - \sigma_{y_iy_i}^2)e_i}{\hat{m}^2\sigma_{x_ix_i}^2 - 2\hat{m}\sigma_{x_iy_i} + \sigma_{y_iy_i}^2} \tag{44}$$

based on Equation (33).

Recall that e_i is the error in the measurement with the appropriate estimates $e_i = -\hat{m}\tilde{x}_i - \hat{c} + \tilde{y}_i$. Note that in

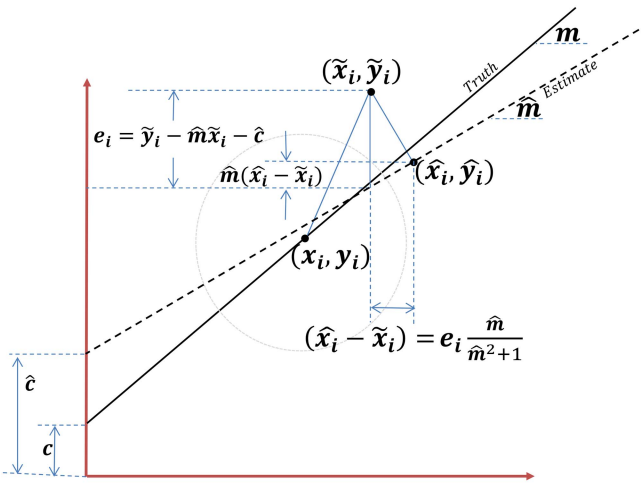


Fig. 6. Optimal estimates of coordinates.

Equations (43) and (44), the estimate for the slope and intercept must be known in order to calculate $\hat{x}_i^2$ and $\hat{y}_i^2$. In our scenario these initial estimates are given by the basic Least Squares solution rather than the solution of the Hough transform. This is to preclude singular estimates for the slope and intercept:

$$\hat{m} = -\frac{\cos\theta}{\sin\theta} \tag{45}$$

$$\hat{c} = \frac{\rho}{\sin\theta} \tag{46}$$

if the Hough transform returns a line estimate with θ exactly zero. Note that the Hough transform is used to identify the measurements which are associated with the line which is being estimated.

With the initial estimates for the slope and intercept, one can update the estimate of the true value $\hat{x}_i$ using Equation (43). In addition to the estimated true value of x, the line's coefficients are updated using the Total Least Squares solution. The next step is to determine the uncertainty in these estimates. The probability density function for (x_i, y_i), given the estimates $[\hat{m}, \hat{c}, \hat{x}_i]$ is:

$$\begin{aligned} &p(\tilde{x}_i, \tilde{y}_i \mid \hat{m}, \hat{c}, \hat{x}_i) \\ &\quad = \frac{1}{2\pi\sqrt{|\mathbf{Q}_i|}} \exp\left[-\frac{1}{2}\left(\begin{bmatrix}\tilde{y}_i \\ \tilde{x}_i\end{bmatrix} - \begin{bmatrix}\hat{m}\hat{x}_i + \hat{c} \\ \hat{x}_i\end{bmatrix}\right)^T \right. \\ &\qquad \left. \times \mathbf{Q}_i^{-1}\left(\begin{bmatrix}\tilde{y}_i \\ \tilde{x}_i\end{bmatrix} - \begin{bmatrix}\hat{m}\hat{x}_i + \hat{c} \\ \hat{x}_i\end{bmatrix}\right)\right]. \end{aligned} \tag{47}$$

The inverse of the covariance matrix $\mathbf{Q}_i$ is easily defined for a 2×2 matrix:

$$\mathbf{Q}_i^{-1} = \frac{1}{\sigma^2_{\tilde{x}_i\tilde{x}_i}\sigma^2_{\tilde{y}_i\tilde{y}_i} - \sigma^2_{\tilde{x}_i\tilde{y}_i}} \begin{bmatrix} \sigma^2_{\tilde{y}_i\tilde{y}_i} & -\sigma_{\tilde{x}_i\tilde{y}_i} \\ -\sigma_{\tilde{x}_i\tilde{y}_i} & \sigma^2_{\tilde{x}_i\tilde{x}_i} \end{bmatrix}. \tag{48}$$

We expand the exponential term in Equation (47) and write it as a product of two variables α_i and K_i which are defined in Equations (49) and (50), respectively:

$$\alpha_i = -\frac{1}{2(\sigma^2_{\tilde{x}_i\tilde{x}_i}\sigma^2_{\tilde{y}_i\tilde{y}_i} - \sigma^2_{\tilde{x}_i\tilde{y}_i})} \tag{49}$$

$$\begin{aligned} K_i = {} & \sigma^2_{\tilde{x}_i\tilde{x}_i}(\tilde{y}_i - \hat{m}\hat{x}_i - \hat{c})^2 - 2\sigma_{\tilde{x}_i\tilde{y}_i}(\tilde{y}_i - \hat{m}\hat{x}_i - \hat{c})(\tilde{x}_i - \hat{x}_i) \\ & + \sigma^2_{\tilde{y}_i\tilde{y}_i}(\tilde{x}_i - \hat{x}_i)^2. \end{aligned} \tag{50}$$

With α_i and K_i defined for each measurement we can then rewrite the likelihood function in a more contracted form as:

$$p(\tilde{x}_i, \tilde{y}_i \mid \hat{m}, \hat{c}, \hat{x}_i) = \frac{1}{2\pi\sqrt{|\mathbf{Q}_i|}} e^{\alpha_i K_i}. \tag{51}$$

For each measurement, there is a corresponding (x_i, y_i), independent of one another. Therefore if we assume there are a total of M measurements, the probability density function for the matrix of measurements, $[\tilde{\mathbf{x}}, \tilde{\mathbf{y}}]$, given the parameter estimates, $[\hat{m}, \hat{c}, \hat{\mathbf{x}}]$, where $\hat{\mathbf{x}}$ is now a vector of estimated x values, is given by the product of each measurement's probability density function:

$$p(\tilde{\mathbf{x}}, \tilde{\mathbf{y}} \mid \hat{m}, \hat{c}, \hat{\mathbf{x}}) = \prod_{i=1}^{M} \frac{1}{2\pi\sqrt{(\sigma^2_{\tilde{x}_i\tilde{x}_i}\sigma^2_{\tilde{y}_i\tilde{y}_i} - \sigma^2_{\tilde{x}_i\tilde{y}_i})}} e^{\alpha_i K_i} \tag{52}$$

The Fisher Information matrix is defined as the negative expected value of the Hessian of the log-likelihood function with respect to the estimated parameters. We define the log-likelihood function as, $f = \ln[p(\tilde{\mathbf{x}}, \tilde{\mathbf{y}} \mid \hat{m}, \hat{c}, \hat{\mathbf{x}})]$, therefore the Fisher Information matrix is defined as:

$$F = -E\begin{bmatrix} \frac{\partial f}{\partial\hat{m}\partial\hat{m}} & \frac{\partial f}{\partial\hat{m}\partial\hat{c}} & \left(\frac{\partial f}{\partial\hat{m}\partial\hat{\mathbf{x}}}\right)^T \\ \frac{\partial f}{\partial\hat{c}\partial\hat{m}} & \frac{\partial f}{\partial\hat{c}\partial\hat{c}} & \left(\frac{\partial f}{\partial\hat{c}\partial\hat{\mathbf{x}}}\right)^T \\ \frac{\partial f}{\partial\hat{\mathbf{x}}\partial\hat{m}} & \frac{\partial f}{\partial\hat{\mathbf{x}}\partial\hat{c}} & \frac{\partial f}{\partial\hat{\mathbf{x}}\partial\hat{\mathbf{x}}} \end{bmatrix} \tag{53}$$

where

$$\frac{\partial f}{\partial\hat{m}\partial\hat{\mathbf{x}}} = \frac{\partial f}{\partial\hat{\mathbf{x}}\partial\hat{m}} = \left[\frac{\partial f}{\partial\hat{m}\partial\hat{x}_1}, \frac{\partial f}{\partial\hat{m}\partial\hat{x}_2}, \ldots, \frac{\partial f}{\partial\hat{m}\partial\hat{x}_M}\right]^T$$

and similarly $\partial f/\partial\hat{c}\partial\hat{\mathbf{x}} = \partial f/\partial\hat{\mathbf{x}}\partial\hat{c}$. The sub-block $\partial f/\partial\hat{\mathbf{x}}\partial\hat{\mathbf{x}}$ of the Fisher Information matrix is a $M \times M$ diagonal matrix with diagonal elements:

$$\frac{\partial f}{\partial\hat{\mathbf{x}}\partial\hat{\mathbf{x}}} = \mathrm{Diag}\left(\left[\frac{\partial f}{\partial\hat{x}_1\partial\hat{x}_1}, \frac{\partial f}{\partial\hat{x}_2\partial\hat{x}_2}, \ldots, \frac{\partial f}{\partial\hat{x}_M\partial\hat{x}_M}\right]\right).$$

First the partial derivatives of f are taken with respect to the estimated parameters. Recall that the probability density function is a product of the individual measurement's and with the properties that the log of

the product becomes a summation over the M measurements. Thus the partial derivatives are:

$$\frac{\partial f}{\partial \hat{m}} = \sum_{i=1}^{M} \alpha_i [-2\sigma_{\tilde{x}_i\tilde{y}_i}(-\tilde{x}_i\hat{x}_i + \hat{x}_i^2) + \sigma^2_{\tilde{x}_i\tilde{x}_i}(-2\tilde{y}_i\hat{x}_i^{+}2\hat{m}\hat{x}_i^2 + 2\hat{x}_i\hat{c})] \quad (54)$$

$$\frac{\partial f}{\partial \hat{c}} = \sum_{i=1}^{M} \alpha_i [-2\sigma_{\tilde{x}_i\tilde{y}_i}(-\tilde{x}_i + \hat{x}_i) + \sigma^2_{\tilde{x}_i\tilde{x}_i}(-2\tilde{y}_i + 2\hat{m}\hat{x}_i + 2\hat{c})] \quad (55)$$

$$\frac{\partial f}{\partial \hat{x}_{t_i}} = \alpha_i [\sigma^2_{\tilde{y}_i\tilde{y}_i}(-2\tilde{x}_i + 2\hat{x}_i) - 2\sigma_{\tilde{x}_i\tilde{y}_i}(-\tilde{y}_i - \tilde{x}_i\hat{m} + 2\hat{m}\hat{x}_i + \hat{c}) + \sigma^2_{\tilde{x}_i\tilde{x}_i}(-2\tilde{y}_i\hat{m} + 2\hat{m}^2\hat{x}_i + 2\hat{m}\hat{c})]. \quad (56)$$

Then the second derivatives, which compose the Fisher Information matrix, can easily be determined using the equations:

$$E\left[\frac{\partial^2 f}{\partial \hat{m}\partial \hat{m}}\right] = \sum_{i=1}^{M} \alpha_i (2\sigma^2_{x_i x_i}\hat{x}_i^2) \quad (57)$$

$$E\left[\frac{\partial^2 f}{\partial \hat{m}\partial \hat{c}}\right] = \sum_{i=1}^{M} \alpha_i (2\sigma^2_{x_i x_i}\hat{x}_i) \quad (58)$$

$$E\left[\frac{\partial^2 f}{\partial \hat{m}\partial \hat{x}_{t_i}}\right] = E\left[\alpha_i(-2\sigma_{x_i y_i}(-x_i + 2\hat{x}_i) + \sigma^2_{x_i x_i}(-2y_i + 4\hat{m}\hat{x}_i + 2\hat{c}))\right] = \alpha_i(-2\sigma_{x_i y_i}\hat{x}_i + 2\sigma^2_{x_i x_i}\hat{m}\hat{x}_i) \quad (59)$$

$$E\left[\frac{\partial^2 f}{\partial \hat{c}\partial \hat{c}}\right] = \sum_{i=1}^{M} 2\alpha_i\sigma_{x_i x_i} \quad (60)$$

$$E\left[\frac{\partial^2 f}{\partial \hat{x}_{t_i}\partial \hat{c}}\right] = \alpha_i(-2\sigma_{x_i y_i} + 2\sigma^2_{x_i x_i}\hat{m}), \quad i = 1,2,\ldots,M \quad (61)$$

$$E\left[\frac{\partial^2 f}{\partial \hat{x}_{t_i}\partial \hat{x}_{t_i}}\right] = \alpha_i(2\sigma^2_{y_i y_i} - 4\sigma_{x_i y_i}\hat{m} + 2\sigma^2_{x_i x_i}\hat{m}^2), \quad i = 1,2,\ldots,M \quad (62)$$

$$E\left[\frac{\partial^2 f}{\partial \hat{x}_{t_i}\partial \hat{x}_{t_j}}\right] = E\left[\frac{\partial^2 f}{\partial \hat{x}_{t_j}\partial \hat{x}_{t_i}}\right] = 0, \quad i \neq j. \quad (63)$$

These equations then give us an estimate of the uncertainty in the estimated parameters. The next step is to perform a Monte Carlo simulation to show the convergence characteristics of this estimate. We begin the simulation by choosing a slope, intercept, and range of x values. These will be the true simulation parameters and are specified as:

$$m = -0.4326 \quad c = -1.6656 \quad x = -3 : 0.1 : 3. \quad (64)$$

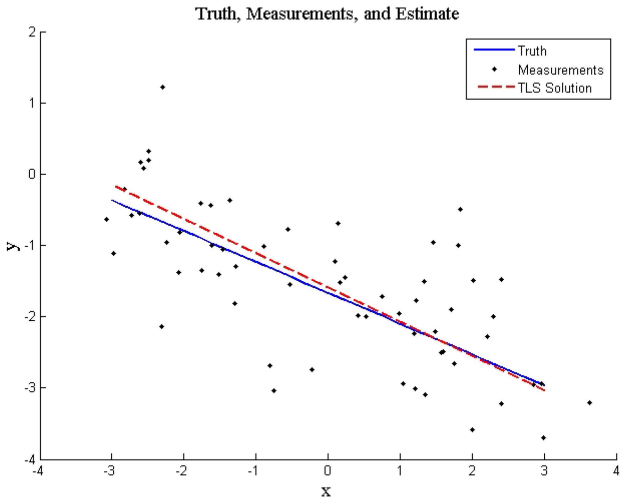


Fig. 7. Truth, measurements, TLS line fit.

Furthermore the covariance matrix, $\mathbf{Q}$ for all measurements is equal and given to be:

$$\mathbf{Q} = \begin{bmatrix} \sigma^2_{yy} & \sigma_{xy} \\ \sigma_{yx} & \sigma^2_{xx} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.0 \\ 0.0 & 0.5 \end{bmatrix}. \quad (65)$$

Since this simulation is to determine the convergence characteristics and not the capabilities of the Hough transform, we will use the Least Squares solution from Equation (36) where the covariance matrix $\mathbf{R}$ is $\sigma^2_{yy}\mathbf{I}_{M\times M}$ and M is the number of measurements, which in this simulation is 61. Since the x truth was already established the y truth can be calculated using the given values for the true slope and intercept. Gaussian white noise is then added to the truth, which was specified in $\mathbf{Q}$ and finally the estimated values of x and y can be obtained via Equations (43) and (44) respectively.

A single simulation's results are shown in Figure 7. Here the truth is shown by the solid blue line, the measurements (truth with added noise) are black dots, and the Total Least Squares line fit is represented by the dashed red line. The estimated values of the slope and intercept from the Total Least Squares algorithm, for this single simulation run are:

$$\hat{m} = -0.3775 \quad \hat{c} = -1.5871. \quad (66)$$

We perform 10,000 simulations to determine the convergence characteristics of the Fisher Information matrix. The measure used for convergence is the determinant of the difference of the Monte Carlo covariance and the inverse of the Fisher Information matrix. The Monte Carlo covariance is calculated as a difference of the truth and the averaged estimates. We will denote this covariance as MC_{cov} and is calculated as:

$$MC_{cov} = \frac{1}{N}\sum_{i=1}^{N}\left(\begin{bmatrix} m \\ c \end{bmatrix} - \begin{bmatrix} \hat{m}_i \\ \hat{c}_i \end{bmatrix}\right)\left(\begin{bmatrix} m \\ c \end{bmatrix} - \begin{bmatrix} \hat{m}_i \\ \hat{c}_i \end{bmatrix}\right)^T \quad (67)$$

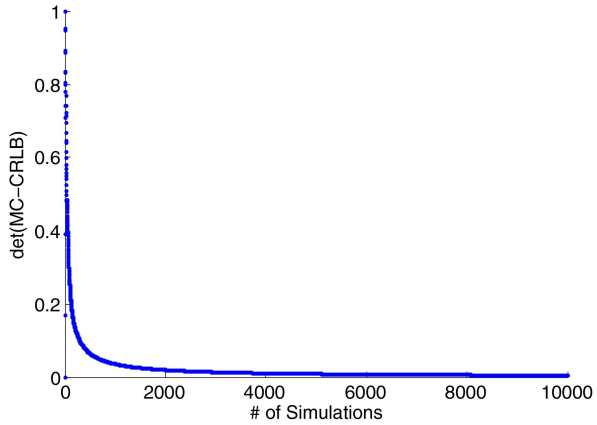


Fig. 8. Monte Carlo simulation.

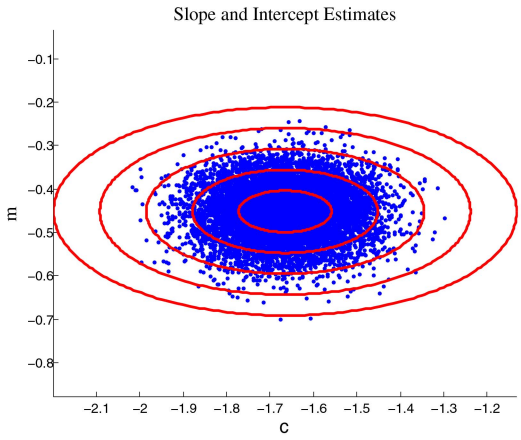


Fig. 9. Sigma ellipses.

where N is the number of Monte Carlo runs and the estimated values m_i, c_i, are used to estimate the covariance. Then the convergence measure is given by the equation:

$$convergence = |MC_{cov} - F^{-1}|. \tag{68}$$

The Fisher Information matrix is also averaged over each simulation. Figure 8 shows the value of this *convergence* measure after each simulation.

Figure 8 shows only a portion of the total number of simulations and in addition the value of the convergence parameter has been normalized. The estimated bounds on the parameters, the inverse of the negative of the Fisher Information matrix after 10,000 simulations is given to be:

$$F^{-1} = 1e^{-4}\begin{bmatrix} 0.14577 & 0.00065 \\ 0.00065 & 0.59065 \end{bmatrix}.$$

Next we examine the estimates and their statistical properties. Each of the estimated values of m and c are plotted in Figure 9 along with the sigma ellipses.

Finally we select all combinations of $\hat{m}$ and $\hat{c}$ which fall within one sigma of the average values of the respective estimates and plot them, where the range is

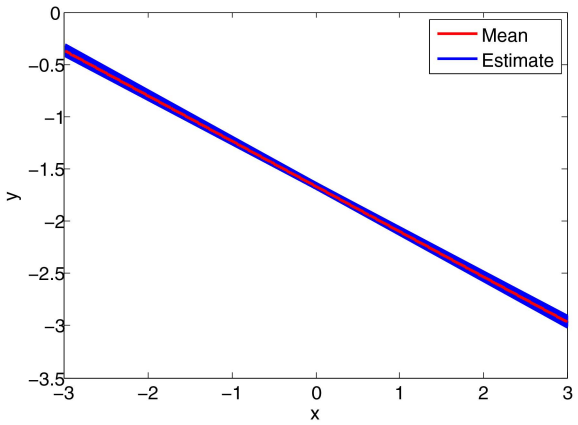


Fig. 10. One sigma slopes/intercepts.

given as:

$$\hat{m} = -0.4326 \pm 0.00382 \tag{69}$$

$$\hat{c} = -1.6648 \pm 0.0077. \tag{70}$$

The lines defined by all such coefficients are plotted, in blue, along with the true fit of the line, in red, in Figure 10 and we can see the uncertainty in the estimates. As we diverge from the relative midpoint of the data range the uncertainty grows. This is to be expected since the variance in the y direction depends on the variance of the slope, intercept, and estimated x value. To prove that the variance in the y direction is dependent on the variance of the slope, intercept, and estimated x value we can transform the Fisher Information matrix from the model parameters into a variance in terms of x and y. This can be done using the Jacobian transformation. In this transformation the Fisher Information matrix is left and right multiplied by the Jacobian of the measurement functions with respect to the estimated parameters. The Jacobian takes the form of:

$$A = \begin{bmatrix} \frac{dy}{d\hat{m}} & \frac{dy}{d\hat{c}} & \frac{dy}{d\hat{x}_i^2} \\ \frac{dx}{d\hat{m}} & \frac{dx}{d\hat{c}} & \frac{dx}{d\hat{x}} \end{bmatrix} = \begin{bmatrix} \hat{x}_i & 1 & \hat{m} \\ 0 & 0 & 1 \end{bmatrix}. \tag{71}$$

Therefore we perform the matrix multiplication to understand the growing variance in the y direction:

$$\begin{aligned} &\begin{bmatrix} \sigma^2_{yy_i} & \sigma_{xy_i} \\ \sigma_{xy_i} & \sigma^2_{xx_i} \end{bmatrix} \\ &\quad = A_i F^{-1} A_i^T \\ &\quad = \begin{bmatrix} \hat{x}_i & 1 & \hat{m} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma^2_{mm} & \sigma_{mc} & \sigma_{mx_t} \\ \sigma_{cm} & \sigma^2_{cc} & \sigma_{cx_t} \\ \sigma_{x_t m} & \sigma_{x_t c} & \sigma^2_{x_t x_t} \end{bmatrix} \begin{bmatrix} \hat{x}_i & 0 \\ 1 & 0 \\ \hat{m} & 0 \end{bmatrix}. \end{aligned} \tag{72}$$

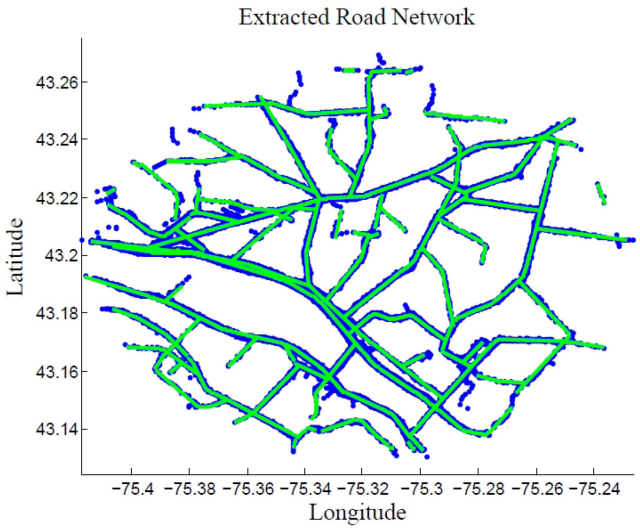


Fig. 11. Data set #2 original measurements and extracted road network.

This results in the covariances taking the form of Equations (73) through (76)

$$\sigma^2_{y y_i} = \hat{x}_i^2 \sigma^2_{mm} + 2\hat{x}_i \sigma_{mc} + 2\hat{m}\hat{x}_i \sigma_{mx} + \sigma^2_{cc} + 2\hat{m}\sigma_{xc} + \hat{m}^2 \sigma^2_{xx} \tag{73}$$

$$\sigma_{x y_i} = \hat{x}_i \sigma_{mx} + \sigma_{cx} + \hat{m}\sigma^2_{xx} \tag{74}$$

$$\sigma_{y x_i} = \sigma_{x y_i} \tag{75}$$

$$\sigma^2_{x x_i} = \sigma^2_{xx}. \tag{76}$$

From the above equations we can see that the variance in the y direction depends on the varying value of $\hat{x}_i$, so therefore as we diverge from $\hat{x}_i = 0$ in either direction the variance in y grows.

4 RESULTS

A Graphical User Interface (GUI) was developed, which allows the user to easily manipulate the processed data. The automated phase of this process consists of the Hough transform and the extraction of the linear features. Several additional operations are allowed by manual interaction, which include merging, trimming, extending, blending, removal, and ellipse identification. The merging option allows the user to select two lines which identify the same road segment. This is possible due to the threshold placed on the distance from a point to the line segment. The trimming and extending features allow a user to smooth out the road network and connect each and every segment with another. A 3rd order polynomial is used as the blending function between two lines which are selected by the user. This updates the road network with curves which are defined by the last xx% of each of the two line segments selected (percentage decided by the user). The final feature, the ellipse identification, is left up to the user due to the computational complexity correlated with automatic identification of an ellipse in an image. The ellipse identification only requires the user to define the number of line segments that are associated with the ellipse and then select each of these segments. This allows for a fast and simple identification of an ellipse.

The data which has been supplied consists of simulated GMTI tracks generated by the Air Force Research Labs in Rome, NY. Each track varies in the number of measurements it contains, however, the structure of each track is consistent. The data structure is broken down as follows:

- *tracks*—main field of the structure.
  - —*tracks(#).loc*—N × 2 matrix of latitude and longitude coordinates.
  - —*tracks(#).cov*—4 × 4 × N covariance matrices corresponding to each measurement of the form.

$$\begin{bmatrix} \sigma^2_{x_i x_i} & \sigma_{x_i y_i} & \sigma_{x_i \dot{x}_i} & \sigma_{x_i \dot{y}_i} \\ \sigma_{y_i x_i} & \sigma^2_{y_i y_i} & \sigma_{y_i \dot{x}_i} & \sigma_{y_i \dot{y}_i} \\ \sigma_{\dot{x}_i x_i} & \sigma_{\dot{x}_i y_i} & \sigma^2_{\dot{x}_i \dot{x}_i} & \sigma_{\dot{x}_i \dot{y}_i} \\ \sigma_{\dot{y}_i x_i} & \sigma_{\dot{y}_i y_i} & \sigma_{\dot{y}_i \dot{x}_i} & \sigma^2_{\dot{y}_i \dot{y}_i} \end{bmatrix}$$

  - —*tracks(#).vel*—N × 2 matrix of component velocities at the same time the measurement is taken.
  - —*tracks(#).update*—unix time representation of measurement time (seconds since January 1, 1970).

The data set that was used to test the proposed algorithm includes 1,675 tracks, where every track had differing numbers of kinematic data.

The data set was completely processed without manual intervention (i.e., merging or ellipse finding). This data set contained several areas of interest which may create issues which include sectors with no track data. Primarily we expect there to be a significant increase in the number of extracted segments.

The line extraction portion of the algorithm took approximately 25 minutes to complete. The extracted line segment data structure was stored before any additional functions were implemented. Prior to any removal or merging of line segments, the data structure contains 150 individual line segments. The merging, blending, and trimming/extending was complete in approximately 3 hours. This data set consisted of 1,675 tracks with a total of 88,685 measurements.

The extracted network was converted back to the latitude longitude coordinate system and the results of the extracted network plotted on top of the original measurements is shown in Figure 11. Although it appears that there is a low association due to the lack of identified segments in certain regions, there are actually very few data points in these areas, 99.8% of the data has been associated with geometric features in the image (88,508/88,685). The final data structure consists of 0 ellipses, 72 third order polynomials, and 131 line segments. We note that some areas are lacking extracted features which is due to either the lack of data or the user removed a feature due to inaccuracy.

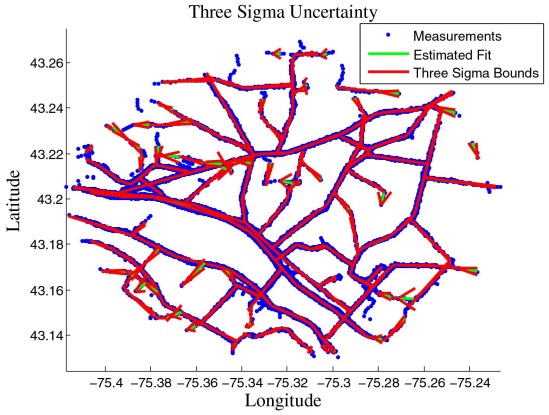


Fig. 12. Data set #2 line CRLB.

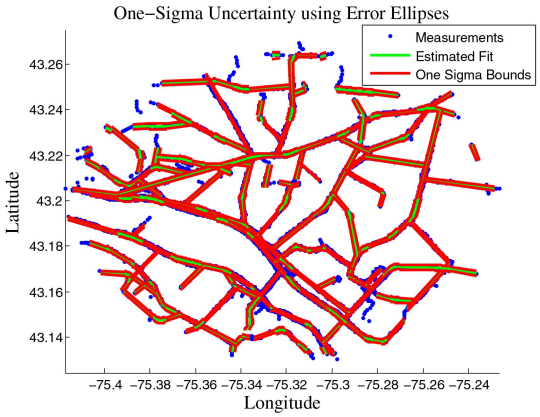


Fig. 13. Data set #2 error ellipses one sigma.

First we present the results for the estimated CRLB matrix corresponding to the line estimate's parameters. Figure 12 shows the three sigma bounded region for the lines, which again due to the number of measurements associated with each of the lines, converges to small values for most lines. For a few straight line segments which have very few associated measurements, the bounds are very lax.

We supplement Figure 12 with an example of a line's CRLB estimate. Equation (77) refers to a line which has 1,034 measurements associated with it:

$$\begin{bmatrix} \sigma^2_{mm} & \sigma_{mc} & \sigma_{mx} \\ \sigma_{cm} & \sigma^2_{cc} & \sigma_{cx} \\ \sigma_{xm} & \sigma_{xc} & \sigma^2_{xx} \end{bmatrix} = \begin{bmatrix} 2.2541e-05 & 3.5129e-09 & -1.7855e-09 \\ 3.5129e-09 & 3.6176e-10 & -1.2573e-10 \\ -1.7855e-09 & -1.2573e-10 & 1.4242e-10 \end{bmatrix} \tag{77}$$

which illustrates a high degree of confidence in the estimated model parameters. We must then use the Jacobian transformation to obtain the covariance in terms of x and y. A single measurement's covariance matrix is given by the equation:

$$\begin{bmatrix} \sigma^2_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma^2_{yy} \end{bmatrix} = 1.0e-06 \times \begin{bmatrix} 0.3850 & -0.4209 \\ -0.4209 & 0.8059 \end{bmatrix}. \tag{78}$$

The Jacobian transformed CRLB is then added to the measurement covariance defined in Equation (78) to produce the final covariance in the measurement including the covariance in the estimate given by Equation (79):

$$\begin{bmatrix} \sigma^2_{xx} & \sigma_{xy} \\ \sigma_{yx} & \sigma^2_{yy} \end{bmatrix} = 1.0e-06 \times \begin{bmatrix} 0.3852 & -0.4209 \\ -0.4209 & 0.8089 \end{bmatrix}. \tag{79}$$

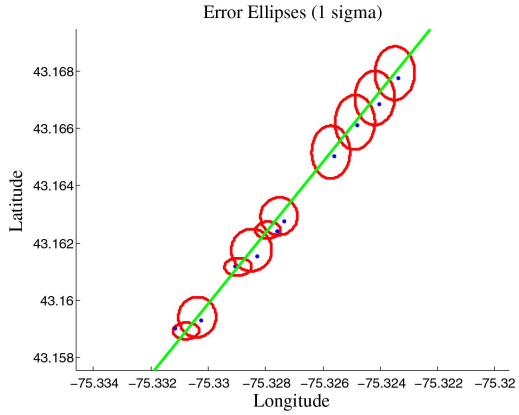


Fig. 14. Data set #2 mean line uncertainty.

Figure 13 shows a few measurement's one sigma ellipses with the line estimate.

Using the error ellipses we once again compute the mean covariance in x and y. This covariance is then used to represent a mean error in the line using one sigma we can see from Figure 14 that this encompasses the majority of the data associated with each of the line estimates.

There are no ellipses identified in this data set but there are a significant number of polynomial blends. A polynomial blend in this data set typical contains from 400 to 1,000 measurements. We present the results for a polynomial, which contains 1,032 associated measurements in Equation (80):

$$\begin{bmatrix} \sigma^2_{AA} & \sigma_{AB} & \sigma_{AC} & \sigma_{AD} \\ \sigma_{BA} & \sigma^2_{BB} & \sigma_{BC} & \sigma_{BD} \\ \sigma_{CA} & \sigma_{CB} & \sigma^2_{CC} & \sigma_{CD} \\ \sigma_{DA} & \sigma_{DB} & \sigma_{DC} & \sigma^2_{DD} \end{bmatrix} = \begin{bmatrix} 3.9162e-12 & 2.9909e-10 & -2.0009e-08 & -1.5304e-06 \\ 2.9909e-10 & 2.2842e-08 & -1.5281e-06 & -0.0001 \\ -2.0009e-08 & -1.5281e-06 & 0.0001 & 0.0078 \\ -1.5304e-06 & -0.0001 & 0.0078 & 0.5980 \end{bmatrix}. \tag{80}$$

5 CONCLUSION

This paper presents a systematic approach for the estimation of the road network from ground moving target data assuming the vehicles to be road based vehicles. The position locations of the multiple ground vehicles are used to generate a binary images, which permits using the Hough transform to identify straight line segments. Having identified the Hough parameters associated with various straight line segments, the position data associated with each of these straight line segments is identified and used by a least squares algorithm to identify the parameters of a straight line, i.e., slope and intercept. Since the two dimensional position data is contaminated with noise in both the dimensions, the total least squares is deemed the appropriate approach for the estimation of the slope and intercept of the straight line segments. A detailed exposition of the total least squares approach for the estimation of the model parameters is followed by the determination of the Cramér-Rao bounds on the estimated parameters. A multi road network with numerous intersections is used as a test case to illustrate the potential of the proposed approach to estimate the road network and characterize the associated uncertainty. Currently the proposed approach is being extended to estimating curved section of roads which are assumed to be representable by arcs of ellipses. This will permit a parsimonious representation of the road-network by eliminating numerous straight line segments which are necessary to represent curved sections of roads. A graphical user interface was also developed to permit a seamless estimation of road-network from ground vehicle kinematic data.

ACKNOWLEDGEMENT

This material is based upon work supported by the AFRL under Cooperative Agreement No. FA 8750-11-2-0082.

REFERENCES

[1] Tupin, Florence, Henri Maitre, Jean-Francois Mangin, Jean-Marie Nicolas, and Eugene Pechersky
Detection of Linear Features in SAR Images: Application to Road Network Extraction.
IEEE Transactions on Geoscience and Remote Sensing 36.2 (1998).

[2] Koch, W., J. Koller, and M. Ulmke
Ground Target Tracking and Road Map Extraction.
ISPRS Journal of Photogrammetry and Remote Sensing 61.3–4 (2006): 197–208.

[3] P. Gamba, F. Dell'Acqua, and G. Lisini
Improving urban road extraction in high-resolution images exploiting directional filtering, perceptual grouping, and simple topological concepts,
IEEE Geosci. Remote Sens. Lett., vol. 3, no. 3, pp. 387–391, Jul. 2006.

[4] V. Amberg, M. Coulon, P. Marthon, and M. Spigai
Structure extraction from high resolution SAR data on urban areas,
in *Proc. IEEE Geosci. Remote Sens. Symp.*, 2004, vol. 3, pp. 1784–1787.

[5] Hu, Jiuxiang, Anshuman Razdan, John C. Femiani, Ming Cui, and Peter Wonka
Road Network Extraction and Intersection Detection From Aerial Images by Tracking Road Footprints.
IEEE Transactions on Geoscience and Remote Sensing 45.12 (2007): 4144–4157.

[6] Aaron, Shackelford K., and Curt H. Davis
Urban Road Network Extraction from High-Resolution Multispectral Data.
2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas, 2003, p. 142–146.

[7] Sklarz, S. E., Novoselsky A., and Dorfan M.
Incremental Fusion of GMTI Tracks for Road Map Estimation.
2008 11th International Conference on Information Fusion, Cologne, Germany, pp. 1–7.

[8] Halir, R. and Flusser, J.
Numerically stable direct least squares fitting of ellipses.
The Sixth International Conference in Central Europe. 1998.

[9] Duda, R. O. and P. E. Hart
Use of the Hough Transformation to Detect Lines and Curves in Pictures,
Comm. ACM, Vol. 15, pp. 11–15 (January, 1972).

[10] Guil, N., and E. L. Zapata
Lower Order Circle and Ellipse Hough Transform.
Pattern Recognition 30.10 (1997): 1729–1744.

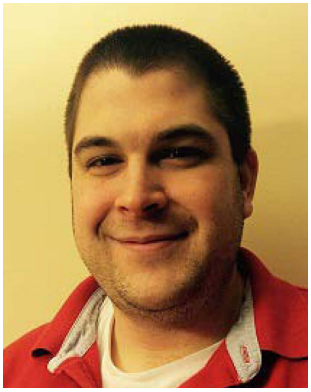
[11] Tsuji, Saburo, and Fumio Matsumoto
Detection of Ellipses by a Modified Hough Transformation.
IEEE Transactions on Computers C-27.8 (1978): 777–781.

[12] Aguado, A. S., and M. S. Nixon
"A New Hough Transform Mapping for Ellipse Detection."
Technical Report, University of Southampton, 1995.

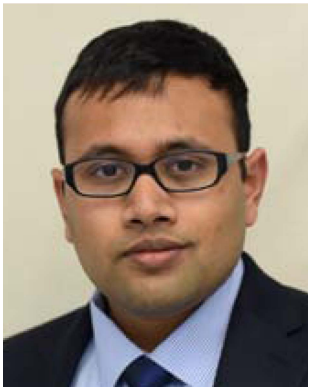
[13] Crassidis, John L., and Yang Cheng
Error-Covariance Analysis of the Total Least Square Problem.
Proceedings of the AIAA Guidance, Navigation and Control Conference, Portland, Oregon, August 8–11, 2011.

[14] Fitzgibbon, A. W. and Fischer, R. B.
A buyer's guide to conic fitting.
Proc. of the British Machine Vision Conference. pp. 265–271. 1995.

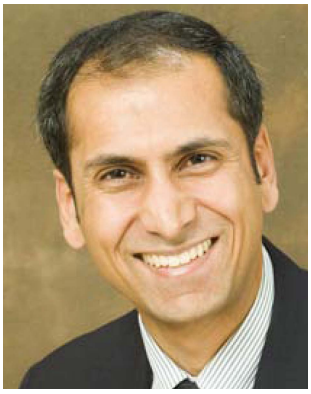
[15] Weisstein, Eric W.
"Ellipse."
From MathWorld—A Wolfram Web Resource.
http://mathworld.wolfram.com/Ellipse.html, Web. 18 Nov. 2014.



Eric Salerno received his Masters of Science in Mechanical Engineering from the State University of New York at Buffalo (Buffalo, New York) in 2013 where he focused his studies on estimation and control systems. He completed two internships with the Air Force Research Laboratory in Rome, New York where he worked on test bedding Natural Language Processing algorithms and developing a Bayesian model for the estimation of remaining useable life for cellular devices in combat zones. He is currently pursuing a second Master's of Science in Computer Science from the State University of New York Polytechnic Institute (Utica, New York) as well as his Certified Information Systems Security Professional certification and currently works at the Department of Defense.



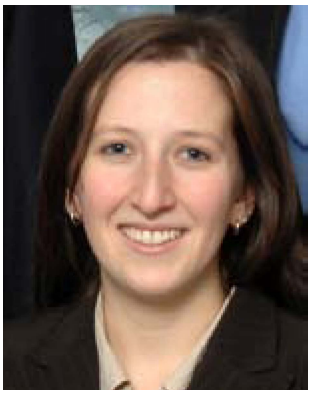
Nagavenkat Adurthi is a graduate student at the University at Buffalo, pursuing a Ph.D. degree. He has received his Bachelors from the Indian Institute of Technology at Guwahati in 2010 and a Master's degree from the University at Buffalo in 2013. He was recognized with the NAGS Distinguished Master's Thesis Award for the work on "The Conjugate Unscented Transform—An Approach to Evaluate Multidimensional Integrals." His research interests include the fields of Uncertainty Quantification, Information Theory and State Estimation.



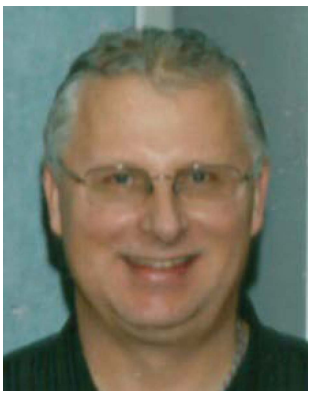
Tarunraj Singh is a fellow of ASME and AAAS. He received his B.E. degree from Bangalore University, Bangalore, India, a M.E. degree from the Indian Institute of Science, Bangalore, and his Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, all in mechanical engineering. He was a Postdoctoral Fellow with the Aerospace Engineering Department, Texas A&M University, College Station. Since 1993, he has been with the University at Buffalo, Buffalo, NY, where he is currently a Professor with the Department of Mechanical and Aerospace Engineering. He was a von Humboldt Fellow and spent his sabbatical at the IBM Almaden Research Center in 2000{2001 and at RWTH Aachen in 2011. He was a National Aeronautics and Space Administration Summer Faculty Fellow with the Goddard Space Flight Center in 2003. His research has been supported by the National Science Foundation, Air Force Office of Scientific Research, National Security Agency, Office of Naval Research, and various industries, including MOOG Inc. Praxair and Delphi Thermal Systems. He has published more than 200 refereed journal and conference papers and has presented over 50 invited seminars at various universities and research laboratories. His research interests are in robust vibration control, estimation, and intelligent transportation.

Puneet Singla (Ph.D., 2006, Texas A&M University) is an Associate Professor and director of graduate studies of Mechanical & Aerospace engineering at the University at Buffalo. He has developed a vigorous research program in the general area of estimation and control with specific focus on uncertainty quantification and characterization at University at Buffalo. He has authored over 100 papers to-date including one book and 27 journal articles covering a wide array of problems. He has received competitive NSF CAREER award for his work on Uncertainty Propagation and Data Assimilation for Toxic Cloud Prediction and the AFOSR Young Investigator Award for his work on Information Collection and Fusion for Space Situational Awareness. He is a senior member of the American Institute of Aeronautics and Astronautics (AIAA), and member of the American Astronautical Society (AAS), the American Society for Mechanical Engineers (ASME), the Institute of Electrical and Electronics Engineers (IEEE) and the Society for Industrial and Applied Mathematics (SIAM).

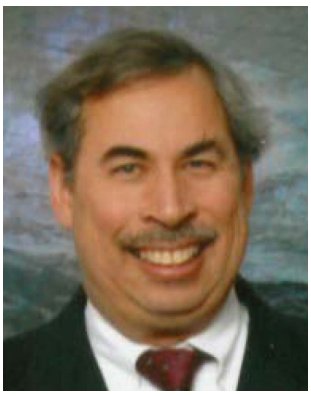
Adnan Bubalo received his M.S. degree in Computer and Information Science from the State University of New York Institute of Technology in 2009. He is presently a Computer Scientist with the Air Force Research Laboratory (AFRL) Information Directorate in Rome, NY. Adnan is the Deputy Lead for the AFRL Processing and Exploitation (PEX) Core Technical Competency (CTC). His research interests include target tracking, information fusion, and clustering.



Maria Cornacchia received her M.S. degree in Computer and Information Science from Syracuse University in 2009. Mrs. Cornacchia is also currently pursuing a Ph.D. in Computer Information Science and Engineering from Syracuse University. She is a Computer Scientist with the Air Force Research Laboratory (AFRL) Information Directorate in Rome, NY. Her research interests include target tracking, information fusion, data mining, and computer vision.



Mark Alford received his M.S. degree in Electrical and Computer Engineering from Syracuse University in 1988. He is presently an Electronics Engineer with the Air Force Research Laboratory (AFRL) Information Directorate in Rome, NY. Mark is the primary in-house program manager for the Activity Based Analysis branch. He has a strong background in fusion and tracking and his research interests include both quantitative and qualitative intelligence analysis and reporting: primarily from chat, video and images.



Eric Jones received his M.S. degree in Electrical Engineering from Syracuse University in 1989. He is presently an Electronics Engineer with the Air Force Research Laboratory (AFRL) Information Directorate in Rome, NY. His research interests include multiple target tracking, nonlinear filtering, and image analysis.

Efficient GPU-Accelerated Implementation of Particle and Particle Flow Filters for Target Tracking

VESSELIN P. JILKOV
JIANDE WU

Particle filtering is a very popular method for nonlinear/non-Gaussian state estimation, however, implementation of particle filters (PFs) with a high state dimension in real-time is a very challenging practical task because the computation is prohibitive. Parallel & distributed (P&D) computing is a natural way to deal with the computational challenges of PF methods in order to make them practical for large scale problems, such as multitarget multisensor tracking. This paper presents results on development, implementation and performance evaluation of computationally efficient parallel algorithms for particle and particle flow filters (PFFs) utilizing a Graphics Processing Unit (GPU) as a parallel computing environment. Proposed are state-of-the-art parallel PF and PFF implementations which are optimized for GPU architecture and capabilities. The proposed algorithms are applied and tested, via simulation, for tracking multiple targets using a pixelized sensor, and for a high-dimensional nonlinear density estimation problem. It is demonstrated by the obtained simulation results that the proposed parallel GPU implementations can greatly accelerate the computation of both PFs and PFFs, and thereby bring them closer to practical applications.

Manuscript received on December 30, 2014; released for publication March 23, 2015.

Refereeing of this contribution was handled by Paolo Braca.

Supported in part by Louisiana BoR through grant LEQSF(2009–12)-RD-A-25, ARO through grant W911NF-08-1-0409, ONR-DEPSCoR through grant N00014-09-1-1169, and NASA/LEQSF (2013–15) through grant NNX13A29A.

Authors'address: Department of Electrical Engineering, University of New Orleans, New Orleans, LA 70148, USA (e-mail: vjilkov@uno.edu).

1557-6418/15/$17.00

I. INTRODUCTION

For more than two decades now, particle filtering [1] has become the most popular approach for nonlinear/non-Gaussian state estimation problems. Particle filters (PFs) have found numerous applications in areas that involve nonlinear filtering of noisy signals (data) [2], most notably in target tracking, e.g., [3]. However, while the literature reporting capabilities of PFs to solve hard nonlinear estimation problems from different application areas is abundant, most practical implementations are limited to small-scale problems with low dimensional state vectors, such as single target tracking. Implementation of high dimensional PFs in real-time for large-scale problems is still quite challenging. Parallel & distributed (P&D) computing is a natural way to overcome/mitigate this limitation. Development of P&D algorithms and architectures that fully exploit the spatial and temporal concurrency of the computations is a great potential to make PFs (and density-based nonlinear filtering, in general) practical for large scale problems, such as multitarget multisensor tracking. Considerable research effort has been going on along this direction, e.g., [4]–[15].

In recent years a new class of nonlinear filters has been gaining momentum—the particle flow filters (PFFs), proposed by Daum & Huang [16]–[22], which overcome the well known problem of particle degeneracy of the PFs (and other, e.g., deterministic sampling methods for density-based nonlinear filtering [23]) caused by the pointwise multiplication of the Bayes rule. The method for samples' update is deterministic and is conceptually based on a natural homotopy relating the prior and posterior filter densities which induces a flow of the samples from the prior density towards a set of samples from the posterior. A flow of each particle from prior to posterior is governed by a flow equation—an ordinary differential equation (ODE)—which in general satisfies a linear partial DE (PDE) with constant coefficients. This PDE is central in this approach. While a numerical integration of the PDE is prohibitive for real-time computation, the good news is that for some special distributions, e.g. Gaussian and exponential families it is analytically tractable. The explicit solution for (unnormalized) Gaussian prior and likelihood, referred to as the exact PFF [19] is straightforward to implement and fast for computation. Plenty of simulation results have been reported by the authors of the PF method, e.g., [18], [21], [22], showing that PFFs can outperform other nonlinear filters in computation and accuracy for difficult nonlinear problems. Another nice property of the PFFs is that they are essentially "embarassingly" parallel—(almost) all computations are independent and can be conducted in a parallel/distrubuted manner as opposed to PFs where resampling is a bottleneck. This makes PFFs even more attractive and promising for P&D computing. It also motivated us to pursue parallel implementations of PFF

TABLE I
Generic SIS/R PF Algorithm [26]

- *Importance Sampling* (IS)
  - — For $i = 1,\ldots,\bar{N}$
    Draw a sample (particle): $\bar{x}_k^i \sim \pi(x_k \mid x_{k-1}^i, z^k)$
    Evaluate importance weights
    $\bar{w}_k^i = w_{k-1}^i \dfrac{p(z_k \mid x_k^i) p(x_k^i \mid x_{k-1}^i)}{\pi(x_k^i \mid x_{k-1}^i, z^k)}$
  - — For $i = 1,\ldots,\bar{N}$
    Normalize importance weights: $w_k^i = \dfrac{\bar{w}_k^i}{\sum_{j=1}^{\bar{N}} \bar{w}_k^j}$
- *Resampling* (R)
  - — Effective sample size estimation: $\hat{N}_{eff} = \dfrac{1}{\sum_{j=1}^{N} (w_k^j)^2}$
  - — If $\hat{N}_{eff} < N_{th}$
    Sample from $\{\bar{x}_k^j, w_k^j\}_{j=1}^{N}$ to obtain
    a new sample set $\left\{ x_k^i = \bar{x}_k^{j_i}, w_k^i = \dfrac{1}{N} \right\}_{i=1}^{N}$

and make a quantitative performance evaluation and comparison with parallel PF algorithms studied by us before [11]–[13].

At present, there are many types of parallel hardware available such as multicore processors, field-programmable gate arrays (FPGAs), computer clusters, and graphics processing units (GPUs). A GPU is a single instruction multiple data (SIMD) parallel processor intended originally to meet the demands of computationally intensive tasks for real-time high-resolution 3D-graphics. Nowadays, GPUs are highly parallel multicore systems that can process very efficiently large blocks of data in parallel [24]. For highly parallelizable algorithms GPUs are becoming more efficient than the sequential central processing unit (CPU) [10]. On the other hand, GPUs are easily accessible and inexpensive—most new personal computers have GPU card. Hence, GPUs offer an attractive opportunity for speeding up PFs and PFFs for real-time applications.

In this paper we present results on development, implementation and performance evaluation of computationally efficient parallel algorithms for particle and particle flow filters by utilizing a Graphics Processing Unit (GPU) as a parallel computing environment. Proposed are state-of-the-art parallel PF and PFF implementations which are optimized for GPU architecture and capabilities. The proposed algorithms are applied and tested, via simulation, for tracking multiple targets using a pixelized sensor (up to 20 targets), and for a high-dimensional nonlinear density estimation problem (up to 40-dimensional state vector). Comprehensive simulation results are presented which illustrate that the proposed parallel GPU implementations can greatly accelerate the computation of both PF and PFF, and thereby bring them closer to practical applications.

The paper is organized as follows. Sect. II provides background information on basic particle & particle flow filtering, and GPU concepts, needed for the rest of the paper. Sect. III proposes parallel PF and PFF algorithms, optimized for GPU implementation. Sect. IV presents application of the parallel GPU PFs for tracking multiple ground targets with a pixelized sensor by a Joint Multitarget Probability Density (JMPD) algorithm and analyzes their performance based on the obtained simulation results. Sect. V presents implementation and performance evaluation/comparison of parallel PF vs. PFF over a high dimensional density estimation problem. Sect. VI provides a summary and conclusions.

This paper is an outgrowth of our previous conference papers [13] and [14], wherein some preliminary parts of this paper have been presented. However the presentation for this journal paper has been unified, improved, and extended with significant new results, e.g., the enhanced parallel PF presented in Sect. III-B and its performance evaluation in Sect IV-B.2.

II. BACKGROUND

This section outlines very briefly the computational procedures of the generic Particle Filter (PF) and the Exact Particle Flow Filter (PFF), and provides some basic GPU computing concepts, needed for the rest of the paper. Parallelization of the filtering algorithms and their GPU implementation are discussed in Section III.

A. Bayesian Recursive Filter

Let $\{x_k\}_{k=1,2,\ldots}$ be a vector valued discrete-time Markov process with state transition probability density function (pdf) $p(x_k \mid x_{k-1})$, and $\{z_k\}_{k=1,2,\ldots}$ be another process, stochastically related to $\{x_k\}_{k=1,2,\ldots}$ through the likelihood $p(z_k \mid x_k)$. x_k and z_k are the state and the measurement, respectively, and $p(x_k \mid x_{k-1})$ and $p(z_k \mid x_k)$ are the state and measurement models. The exact Bayesian recursive filter (BRF) provides the posterior density $p(x_k \mid z^k)$ via the following prediction-update scheme [25]:

$$p(x_{k-1} \mid z^{k-1}) \rightarrow p(x_k \mid z^{k-1}) \rightarrow p(x_k \mid z^k) \qquad (1)$$

given $p(x_0)$ and measurements $z^k = \{z_1,\ldots,z_k\}$.

B. Generic Particle Filter

In particle filtering the pdfs are represented approximately through a set of random samples (particles) and the BRF (1) is performed directly on these samples. Most PFs are based on two principal components: sequential importance sampling (SIS) and resampling (R) as given in Table I [26].

The importance distribution $\pi(\cdot)$ must contain the support of the posterior and is subject to design. One possibility is to choose $\pi = p(x_k \mid x_{k-1}^i)$ [1] which is often referred to as the sampling importance resampling (SIR) PF, or the bootstrap PF. Many other choices are possible [2]. Resampling is in effect discarding of samples that have small probability and concentrating on samples that are more probable. The resampling step is critical in every implementation of PF because

TABLE II
Gaussian Exact Particle Flow Algorithm

- *Particle Prediction*
  — For $i = 1,\ldots,\bar{N}$
  Draw a predicted particle: $\bar{x}_k^i \sim p(x_k \mid x_{k-1}^i)$
- *Particle Update*
  — Compute predicted estimate $\bar{x}_k$, its error covariance P, and linearized measurement model matrix H
  — Compute parameters of flow velocity (exact computation):
  $A(\lambda) = -\frac{1}{2}PH'(\lambda HPH' + R)^{-1}H$
  $b(\lambda) = (I + 2\lambda A)[(I + \lambda A)PH'R^{-1}z_k + A\bar{x}_k]$
  — For $i = 1,\ldots,N$
  Solve the particle flow ODE
  $\frac{dx}{d\lambda} = f(x,\lambda) = A(\lambda)x + b(\lambda)$
  for $\lambda \in [0\ \ 1]$ with initial condition $x(0) = \bar{x}_k^i$;
  Let $x_k^i := x(1)$.

without it the variance of the particles weights quickly increases leading to inference degradation.

C. Gaussian Exact Particle Flow Filter

In this paper we limit our consideration to the Gaussian Exact PFF [19]. We summarize one time-step, $k-1 \to k$, of the algorithm in Table II.

First, note that the prior and posterior at each time-step k are also represented through samples: $\{\bar{x}_k^i\}_{i=1}^{\bar{N}}$ and $\{x_k^i\}_{i=1}^{N}$, respectively. In Table II we give the prediction step in terms of random sampling (as in the bootstrap PF) which amounts to passing each particle from the posterior through the stochastic state dynamic model. However the prediction sampling need not be random in general—the prior can be approximated by deterministic samples as well, e.g., based on optimal Dirac mixture approximations [27]. In our GPU implementation we use random sampling but deterministic sampling for prediction is of further interest for future implementation because generating random numbers by GPU is not straightforward and may introduce some latency.

Second, the computation of the state prediction estimate $\bar{x}_k$, error covariance matrix $\bar{P}_k$, and linearized measurement model H_k is not explicitly given because it can be done in different ways, e.g., $\bar{x}_k$ and $\bar{P}_k$ can be computed as the sample mean an covariance of the particles and then H_k can be obtained via linearization of the measurement model about $\bar{x}_k$, or the computation can be done via EKF/UKF equations.

Third, the flow velocity parameters $A(\lambda)$, $b(\lambda)$ are common for all particles and, therefore, their computation is given outside the `for`-loop for solving the ODE in Table II. However, in a parallel computer implementation this is not necessarily the fastest arrangement—Assuming each "processor" is dedicated to one particle or a groups of particles (as in our GPU-implementation, discussed in Sect. III), it could be better to compute $A(\lambda)$, $b(\lambda)$ for each particle (as if within the particles' `for`-loop) or group of particles in parallel, and thus eliminate the time for transferring data among "processors."

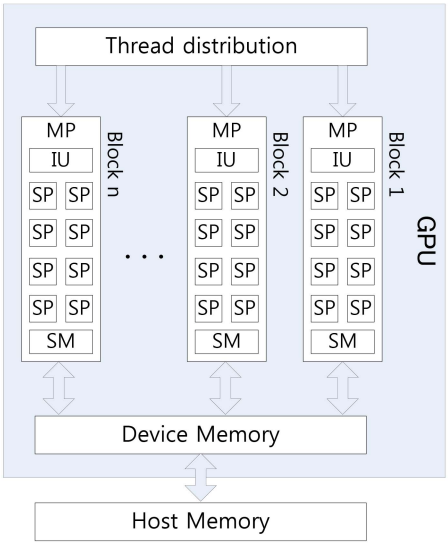


Fig. 1. Nvidia GPU Architecture [28]

D. GPU & CUDA Computing

A typical GPU is a collection of multiprocessors (MPs) where each of them has several scalar processors (SP), also referred to as cores [28]. At any given clock cycle, each SP executes the same program (one or a set of instructions) on different data by following the single instruction/program multiple data (SIMD/SPMD) models. Each SP has access to different memory levels. Fig. 1 gives a simplified illustration of the architecture of Nvidia GPU, where SM stands for shared memory, and IU—for instruction unit. This type of architecture is ideally suited to data-parallel computation since large quantities of data can be loaded into shared memory for the cores to process in parallel.

Compute Unified Device Architecture (CUDA) is a parallel computing architecture developed by Nvidia as a computing engine in GPUs. It allows access to the virtual instruction set and memory of a GPU's parallel computational elements. By using CUDA the GPUs can be used for computation like CPUs.

A detailed description of CUDA is given in [29]. The MP model used in CUDA is called single-instruction multiple-thread (SIMT). In SIMT, the MP maps each thread to one SP core, and each thread executes independently with its own instruction address and register state. The MP creates, manages, and executes concurrent threads in hardware with no scheduling overhead. The threads are logically organized in blocks and grids. A block is a set of threads, while a grid is a set of blocks. The sizes of the blocks and grids can be programmatically controlled. To optimize an execution configuration the first parameters to choose are the grid size (number of blocks per grid) and block size (number of threads per block). As a general guideline, provided by [29], the primary concern is keeping the entire GPU busy—the

number of blocks in a grid should be larger than the number of multiprocessors so that all multiprocessors have at least one block to execute, and there should be multiple active blocks per multiprocessor so that blocks which are not waiting for thread synchronization can keep the hardware busy.

CUDA devices use several memory spaces, which have different characteristics that reflect their distinct usages in CUDA applications. These memory spaces include global, local, shared, texture, constant, and registers. There is a 16 KB per thread limit on local memory, a total of 64 KB of constant memory, and a limit of 16 KB of shared memory, and either 8,192 or 16,384 32-bit registers per multiprocessor. Global, local, and texture memory have the greatest access latency, followed by constant memory, registers, and shared memory. Memory optimizations are key for performance. The best way to maximize bandwidth is to use as much fast-access memory and as little slow-access memory as possible.

III. PARALLEL PF AND PFF IMPLEMENTATION ON GPU

The general idea of implementing both particle algorithms (PF and PFF) on GPU is to map (dedicate) a thread to a particle and organize the algorithms in terms of groups of particles, mapped correspondingly to GPU blocks, in order to take advantage of the GPU architecture. The key is to use the shared memory for fast particle data communication within each block and avoid/reduce communications between different blocks, as much as possible. Optimizing any particular implementation is crucially dependent on the capabilities of the available GPU hardware, as well as, on the filtering problem dimension (size). For our application studies this issue is discussed further in Sect. IV.

Other important issues arising in the GPU implementation for both filters are the cumulative summation (needed for normalization and sample mean computation) and random number generation. The cumulative sum can be implemented using a multipass scheme similar to that of [10]. This multipass scheme is a standard method for parallelizing seemingly sequential algorithms based on the scatter and gather principles. The CURAND library [30] provides facilities to efficiently generate high-quality pseudorandom and quasirandom numbers. It is used to create several generators at the same time. Each generator has a separate state and is independent of all other generators.

A. Parallel PF

The computation for importance sampling is independent across particles and can be executed in parallel without any data communication among particle-dedicated processors (or threads). Parallelization of the resampling part, however, is quite nontrivial because generating a single resampled particle requires information from all particles of the sample set. Thus, resampling becomes a bottleneck in parallel implementations. Several parallel/distributed resampling schemes have been already proposed and studied in the literature, e.g., [4], [7]. In this paper we implement distributed resampling with nonproportional allocation (RNA) [7]. Briefly, the idea is as follows (more details, including some high level pseudocode, can be found in [7] and our previous papers [11]–[13]).

Distributed RNA is a modification of the distributed resampling with proportional allocation (RPA) which is essentially based on stratified sampling [2]. The sample space is partitioned into several strata (groups) and each stratum corresponds to a processing node (PN)—mapped to a GPU block of threads in our GPU implementation. Proportional allocation among strata is used, which means that more samples are drawn from the strata with larger weights. After the weights of the strata are known, the number of particles that each stratum replicates is calculated at a head processing node (HN)—a dedicated block of threads in a GPU implementation—using residual systematic resampling, and this process is referred to as inter-resampling since it treats the PNs as single particles. Finally, resampling is performed inside the strata (at each PN, in parallel) which is referred to as intra-resampling. Therefore, the resampling algorithm is accelerated by having an inner loop that can run in parallel on the PNs (intra-resampling) with small centralized processing (inter-resampling) at HN. RPA requires a complicated scheme for particle routing. In the parallel RNA particle routing is deterministic and planned in advance by the designer. The number of particles within a group after resampling is fixed and equal to the number of particles per group as opposed to the RPA where this number is random (proportional to the weight of the stratum). This introduces an extra approximation in the resampling (in statistical sense) but allows to execute resampling in parallel by each group (GPU block).

In our implementation each group of particles, as defined in RNA, is naturally mapped into a GPU block as each particle of the group corresponds to a thread of the block. Thus, in terms of the GPU architecture, the importance sampling is thread-parallel while the intra-resampling part is only block-parallel, and the inter-resampling and weight normalization are centralized (Table III). The centralized parts can be computed at the CPU or at some of the blocks of the GPU (designated as a head-node). Both options are implemented in our simulation study, presented in Sect. IV.

B. Enhanced Parallel PF

A significantly enhanced alternative to the presented, RNA-based, PPF is proposed based on the following two reasons. First, each thread can access all particles'

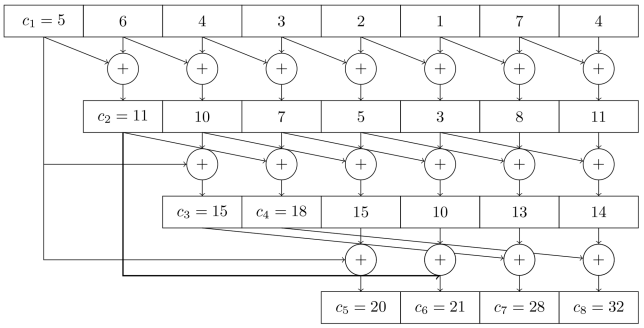


Fig. 2. Parallel Cumulative Sum

TABLE III
Parallel PF Algorithm for GPU

- *Importance Sampling* (IS)
  - — For $i = 1,\ldots,N$ (**Thread Parallel**)
    Sample: $\bar{x}_k^i \sim \pi(x_k \mid x_{k-1}^i, z^k)$
    Evaluate importance weights $\bar{w}_k^i$
  - — For $i = 1,\ldots,N$ (**Centralized**)
    Normalize importance weights: $w_k^i = \dfrac{\bar{w}_k^i}{\sum_{j=1}^{N} \bar{w}_k^j}$
- *Resampling* (R)
  $\{\bar{x}_k^j, w_k^j\}_{j=1}^N \Rightarrow \left\{ x_k^i = \bar{x}_k^{j_i}, w_k^i = \dfrac{1}{N} \right\}_{i=1}^N$
  - — *Inter Resampling* (**Centralized**)
  - — *Intra Resampling* (**Block Parallel**)

information concurrently through the shared and device memory and, thus, replicating particles could be parallelized in all threads. Second, the cumulative sum of weights, which is key for resampling, can be also executed in parallel in contrast to the generic (RNA-based) PPF where it is done in a centralized manner.

When implementing a cumulative weight sum on a single processor, the time complexity is $O(n)$ addition operations for an array of length n. This is the minimum number of additions required. A parallel algorithm is presented in [31]. Fig. 2 shows the operation. The time complexity is $O(\log n)$ if enough processors are available. This is fine for small arrays. In our application, particle filter algorithms use large number of particles, so large arrays are used.

We modify the PPF by using a parallel cumulative weighted sum algorithm. The enhanced PPF algorithm is shown in Table IV.

Additional illustration of the operation of the EPPF is given in Fig. 3. Note that Step 3 can be done thread parallel because all threads can access all cumulative sums concurrently.

After optimizing the particle filter algorithm, the main bottleneck left is the global memory access. To better cover the global memory access latency and improve overall efficiency, we should let each thread do more computation. So we could let each thread process two or four particles instead of one particle. Each thread performs a sequential access of four particles and stores

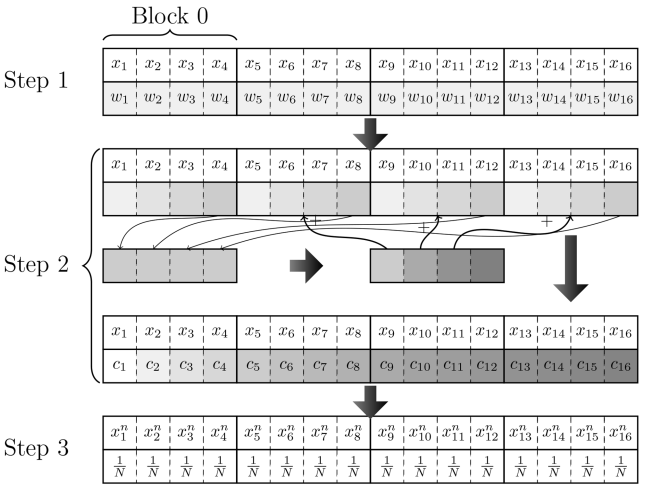


Fig. 3. Illustration of the Enhanced GPU PF

TABLE IV
Enhanced PPF Algorithm for GPU

- *Importance Sampling* (IS)
  - — For $i = 1,\ldots,N$ (**Thread Parallel**)
    Sample: $\bar{x}_k^i \sim \pi(x_k \mid x_{k-1}^i, z^k)$
    Evaluate importance weights $\bar{w}_k^i$
- *Unnormalized Cumulative Sum of Weights* (UCSW)
  - — For $j = 1,\ldots,N$ (**Block Parallel**)
    $\bar{c}_k^j = \bar{w}_k^1 + \cdots + \bar{w}_k^j$
    Execute UCSW for each block
    Execute UCSW for all blocks
    Get all UCSWs
- *Resampling* (R)
  - — For $i = 1,\ldots,N$ (**Thread Parallel**)
    Normalize UCSW: $c_k^i = \dfrac{\bar{c}_k^i}{\sum_{j=1}^{N} \bar{c}_k^j}$
    Sample from $\{x_k^j\}_{j=1}^N$ with $\{c_j\}_{j=1}^N$ to obtain a new sample set $\left\{ x_k^i = x_k^{j_i}, w_k^i = \dfrac{1}{N} \right\}_{i=1}^N$

them in registers. This method is more than twice as fast as the code which only processes one particle at each thread.

C. Parallel PFF

GPU parallelization of PFF is much easier than that of the PF. It is apparent from Table II that the prediction part and solving the ODE for each particle is independent across particles and can be executed completely in parallel by mapping a particle to a GPU thread. For computing $\bar{x}_k$, P_k, H_k, and the flow velocity parameters $A(\lambda)$, $b(\lambda)$ there are different options because they are common for all particles. One way is to have them computed by the CPU ("externally" to the GPU) by a point estimator like EKF/UKF and sent to each particle. Another way is to collect all predicted particles in the CPU, compute sample mean $\bar{x}_k$, and based on it, compute P_k, H_k, $A(\lambda)$, $b(\lambda)$ and send the results to each particle. Involvement of the CPU imposes a communication time overhead (due to the need

to access the slow memory), and does not take advantage of the GPU block architecture. That is why, as in the GPU-PF, we divide all particles in groups and map each group to a GPU block wherein each particle corresponds to a thread. A predicted estimate $\bar{x}_{k,b}$ is computed for each block b, as the sample mean of its particles, and based on it, flow velocity parameters for this block are computed. Thus, even though some extra approximation is incurred, any communication with the CPU and among different block is avoided. An outline of the Parallel GPU PFF is shown in Table V.

For computing a block $\bar{x}_{k,b}$ we use the sample mean of the particles within the block. Then $P_{k,b}$, $H_{k,b}$ are computed using the EKF equations, based on $\bar{x}_{k,b}$.

For numerical computation of the ith particle's flow ODE we use the finite difference forward Euler method (as given in Table V), where $L > 1$ is an integer defining the discretization step $\Delta\lambda = 1/L$ of the interval [0 1].

In accordance with the limitation of shared memory (our device is 48K bytes/block), the number of particles we include in each block is given in Table VIII. We also use the same block sizes for the GPU-PF implementation.

Note that, except for the small block sample mean part (which is block parallel), this GPU-PFF implementation is thread parallel as opposed to the GPU-PF implementation (Table III) wherein a significant part (the resampling) is only partially (block) parallel. This clearly gives an explanation to the fact that the implemented GPU-PFF is much faster than the GPU-PF for the same number of particles, as seen in the simulation results presented in Sect. V.

TABLE V
Gaussian Exact Parallel PFF Algorithm

- *Particle Prediction*
  - — For $i = 1,\ldots,N$ (**Thread Parallel**)
    Sample: $\bar{x}_k^i \sim p(x_k \mid x_{k-1}^i)$
- *Particle Update*
  - — For $b = 1,\ldots,N_b$ (**Block Parallel**)
    Compute $\bar{x}_{k,b}$, $P_{k,b}$, $H_{k,b}$, $f_{k,b}$
  - — For $i = 1,\ldots,N$ (**Thread Parallel**)
    Compute particle flow
    $x_{[0]} = \bar{x}_k^i$
    $x_{[l+1]} = x_{[l]} + \Delta\lambda f_b(x_{[l]}, l\Delta\lambda)$, $l = 0,\ldots,L-1$
    $x_k^i = x_{[L]}$

IV. SIMULATION STUDY I: GROUND MULTITARGET TRACKING USING IMAGE SENSOR

The main purpose of this simulation is to evaluate the computational feasibility and performance of the GPU PPFs (given in Tables III & IV) for a realistic target tracking scenario of high dimension. The parallel algorithms developed and implemented are based on the Joint Multitarget Probability Density (JMPD) algorithm for multiple target tracking [32]–[37].

A. JMPD Tracking

1) Problem Formulation:

In the JMPD approach to multiple target tracking the uncertainty about the number of targets present in a surveillance region as well as their individual states is represented by a single composite pdf. That is, the state of all targets is described by a meta-target state vector $X_k = (x_k^1, x_k^2, \ldots, x_k^T)$ where x_k^i is the state of target $i = 1,\ldots,T$. The number of targets at time k, $T_k \in [0,1,2,\ldots,\infty)$, is also assumed random. The posterior distribution of interest[1] is

$$p(X_k, T_k \mid Z^k) = p(X_k \mid T_k, Z^k) p(T_k \mid Z^k)$$

where $Z^k = \{Z_1, Z_2, \ldots, Z_k\}$ is the cumulative measurement set of the surveillance region up to time k, and $Z_l, l = 1,\ldots,k$ is the measurement set at time l. The model of target state and number evolution over time is given by $p(X_k, T_k \mid X_{k-1}, T_{k-1})$ and is referred to as the kinematic prior. It includes models of target motion, target birth and death, and any additional prior information on kinematics that may be available, e.g., terrain and road maps. The measurement model over the surveillance region is given by the likelihood $P(Z_k \mid X_k, T_k)$.

For the purpose of our implementation and performance study we adopt the kinematic prior and measurement models of [34].

2) Multitarget Model:

Each target $i = 1,\ldots,T$ is assumed to follow a nearly constant velocity motion model

$$x_k^i = F x_{k-1}^i + w_k^i \tag{2}$$

where $x = (\mathrm{x}, \dot{\mathrm{x}}, \mathrm{y}, \dot{\mathrm{y}})'$ is the state vector, $w \sim \mathcal{N}(0, Q)$ is white process noise with given covariance, and

$$F = \mathrm{diag}\left\{ \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} \right\}$$

where Δ is the sampling interval. To account for maneuvers a mode variable can be also added [36]. The number of target in this paper is considered constant and known. Unknown number of targets using the transitional model of [36] is of interest for future work.

3) Sensor Model:

It is assumed that a pixelized sensor provides raw (unthresholded) measurements data from the surveillance region according to the following association-free model [34] used often for track-before-detect (TBD) problems. A sensor scan at time k consists of the

[1]Note that in this formulation the so-called "mixed labeling" [38] is not addressed. It is assumed that no track extraction is needed and, consequently, the ordering of x^i within X is irrelevant as far as only the density is of interest. In [34] this assumption is referred to as a symmetry under permutation.

outputs of M pixels (cells of the region), i.e., $Z = \{z[1],\ldots,z[M]\}$[2] where $z[i]$ is the output of pixel i. The likelihood $P(Z \mid X,T)$ is given by

$$P(Z \mid X,T) \propto \prod_{i \in i_X} \frac{p_{n_{[i]}(X)}(z[i])}{p_0(z[i])} \quad (3)$$

where i_X is the set of all pixels that couple to X, $n_{[i]}(X)$ is the occupation number of pixel i (number of targets from X that lie in i). The output z of each pixel is assumed to follow the Rayleigh model

$$p_n(z) = \frac{z}{1+n\lambda} \exp\left(-\frac{z^2}{2(1+n\lambda)}\right), \quad n = 0,1,2,\ldots \quad (4)$$

where λ denotes the signal-to-noise ratio (SNR).

4) Particle Filter:

With the definition of the transitional density $p(X_k,T_k \mid X_{k-1},T_{k-1})$ and likelihood $P(Z_k \mid X_k,T_k)$ the solution to the multitarget tracking problem formally boils down to the BRF (1) and the standard SIS/R PF given above can be applied. However, with large number of target the computation requirements become prohibitive. The first step to improve the efficiency of the multitarget PF is to choose an appropriate importance (proposal) distribution for sampling that takes into account the specifics of the multitarget problem. Along with the SIR algorithm's Kinematic Prior (KP) proposal $\pi = p(X_k,T_k \mid X^i_{k-1},T^i_{k-1})$, where $i-$ particle index, [34] suggested three more sophisticated schemes for choosing the importance distribution for multitarget PF, referred to as Independent Partition (IP), Coupled Partition (CP), and Adaptive Partition (AP) [34]. The second step is to parallelize as much as possible the resulting multitarget algorithms. Next we propose parallel implementation of these schemes and incorporate them in the parallel structures of the corresponding overall multitarget parallel algorithms.

5) JMPD Parallel PF:

In the JMPD SIR PF the proposal is just the kinematic prior and the IS step is completely decoupled with respect to particles $\{X^i\}$. Consequently, both versions of the above generic parallel PF (Tables III & IV) work without any modification. The significantly more efficient proposal schemes IP, CP, and AP of JMPD PF have intrinsic coupling among particles introduced by the dependence of the proposal on the current measurement data. By more careful inspection, however, IP and CP can be parallelized as given next.

Each particle i for T_i targets is $X^i = (x^{i,1},x^{i,2},\ldots,x^{i,T^i_k})$ and $x^{i,j}$ is referred to as a partition j of particle i.

The IS step of the JMPD with IP can be done as follows:

A) For each partition $j = 1,\ldots,T^i_k$ (in parallel)

$$x^{ij}_k, w^{ij}_k = \mathsf{IP}[\{x^{ij}_{k-1}, w^{ij}_{k-1}\}^N_{i=1}, Z_k]$$

[2]Time index k is omitted here to lighten notation.

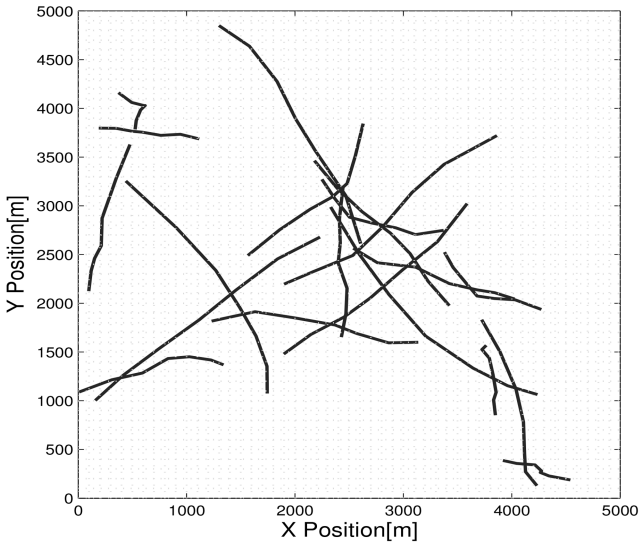


Fig. 4. Scenario with 20 Targets

B) For each particle $1 = 1,\ldots,N$

$$\text{Importance weights } \bar{w}^i_k = w^i_{k-1} \frac{p(Z_k \mid X^i_k)}{\Pi^T_{\theta=1} w^{ij}_k}$$

where IP denotes the IP subroutine of [34] which practically implements the SIR algorithm for each partition. The IS step of the JMPD with CP can be done similarly, except for IP being replaced by a subroutine CP which practically implements the known auxiliary SIR particle filter [2] for each partition but only outputs one resampled partition.

The local importance weights $w^{i,j}_k$ are data dependent and their inclusion in the calculation of importance weights $\bar{w}^i_k$ amounts to improving the proposal π, i.e., bias the proposal towards the optimal importance density $\pi(\cdot,Z_k)$.

Part A) is integrated easily with the RNA resampling scheme used in the PPF (Table III). Part B) can also be computed at the head-node (at the CPU or at the GPU) at the expense of an extra communication between head-node and all threads, or it can be computed locally at each thread but this incurs extra pairwise (thread-to-thread) communication between all threads through the device memory. The latter option is parallel but not necessarily faster due to the communication overhead. In our GPU implementation we use the former option—compute B) at the head-node (implemented alternatively at the GPU or at the GPU).

B. Simulation Experiments & Results

Two simulation experiments were performed using two different hardware computing platforms and parallel algorithms tailored to each one of them, respectively.

Scenarios: For both experiments, the same tracking scenarios were simulated. The parameters of these scenarios are as follows. The targets move in a 5000 m × 5000 m surveillance area. They have a nearly constant

TABLE VI
Hardware Configuration 1

CPU	Model:	Intel Core(TM)2 Duo
	Clock Rate:	1.40 GHz
	Memory:	2.0 G
	Operating System:	Windows 7
GPU	Model:	NVIDIA GeForce 8400M GS
	CUDA Driver:	3.20
	Clock Rate:	0.80 GHz
	Cores:	2 (MP) x 8 (Cores/MP) = 16 (Cores)
	Global Memory:	115M bytes
	Constant Memory:	64K bytes
	Shared Memory:	16K bytes/block

velocity motion, according to the state model (2) with $Q = \text{diag}\{20, 0.2, 20, 0.2\}$. The initial position and velocity of each target, for each Cartesian coordinate x and y are generated randomly from the uniform distributions $\mathcal{U}(0, 5000)$ and $\mathcal{U}(-10, 10)$, respectively. The sensor scans a fixed rectangular region of 50×50 pixels, where each pixel represents a 100 m $\times$ 100 m area on the ground plane. The sensor returns Rayleigh-distributed measurements in each pixel, depending on the number of targets that occupy the pixel according to the measurement model (3). The sensor sampling interval $\Delta = 1s$ and SNR $\lambda = 15$. Scenarios with different number of targets T were simulated. Fig. 4 shows a realization of a scenario with $T = 20$.

1) Experiment 1:

a) Algorithms: Three PFs (CPU-CR, GPU-CR, and GPU-DR)[3] were implemented with different number of particles for each scenario, as follows (see also Sect. III-A).

- *CPU-CR:* CPU performs prediction (draw predicted samples and calculate importance weights) and resampling (using the residual systematic resampling method [2]). This algorithm does not use GPU. It is implemented just for comparison.
- *GPU-CR:* GPU (in parallel) performs importance sampling and CPU performs centralized resampling.
- *GPU-DR:* Both IS & R are performed in parallel on the GPU. DR is as described in Sect. III-A, Table III. After the weights of the blocks are known, the number of particles that each block replicates is calculated at CPU using residual systematic resampling (inter resampling). Finally, intra-resampling is performed inside the block in parallel.

b) Computing Platform: The hardware used in this experiment is presented in Table VI.

Note that there are 8 parallel pipelines and the number of particles chosen $\gg$ number of pipelines. Also, the GPUs has a maximal texture size that limits the number of particles that can be resampled as a single unit (block).

[3]CR stands for Centralized Resampling and DR—for Distributed Resampling

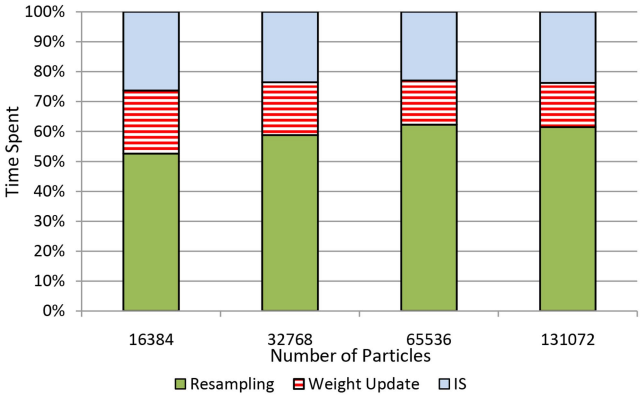


Fig. 5. Relative Times Spent in the Different Steps Using GPU-DR

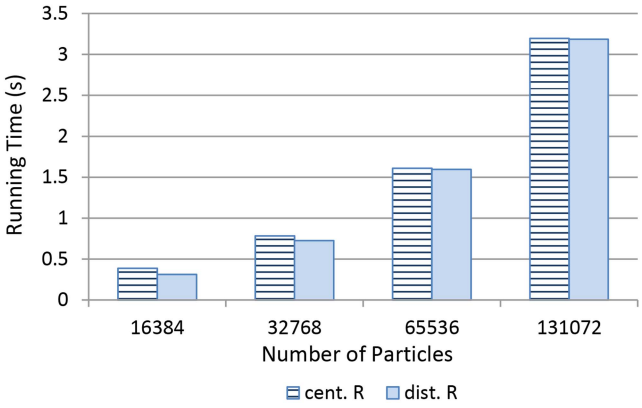


Fig. 6. Computation Times of CR & DR

The code for all implementations is written in C++ and compiled using Visual C++ 2008.

c) Results & Comparison: Due to space limitation, only the most representative results are reported.

First, Fig. 5 shows the computation times spent on different parts of the PF algorithm in the GPU-DR implementation. The resampling step incurs the highest computational cost. Quantitatively, resampling is from two to four times more costly (depending on the number of particles) than importance sampling, and about 2.5 times than generating the estimates.

Second, Fig. 6 shows a comparison between the times for resampling only (CR and DR) with different number of particles. Even though DR is in parallel, the improvement over CR is not significant because the clock rate of GPU is much lower (almost two times) than that of the CPU. Also, as the number of particles increases the efficiency of DR decreases due to the limited pipelines of GPU.

Next, Fig. 7 and Fig. 8 show the position *time-average root-mean square errors* (TARMSE) and execution times of the three PF algorithms for 3 targets, respectively. Fig. 9 and Fig. 10 are for 20 targets. Fig. 7 indicates that, for 3 targets, using slightly more than 60K particles (in all filters) is a reasonable choice for practical purposes. For 20 targets, Fig. 9 indicates that more than 130k particles are needed. These figures also

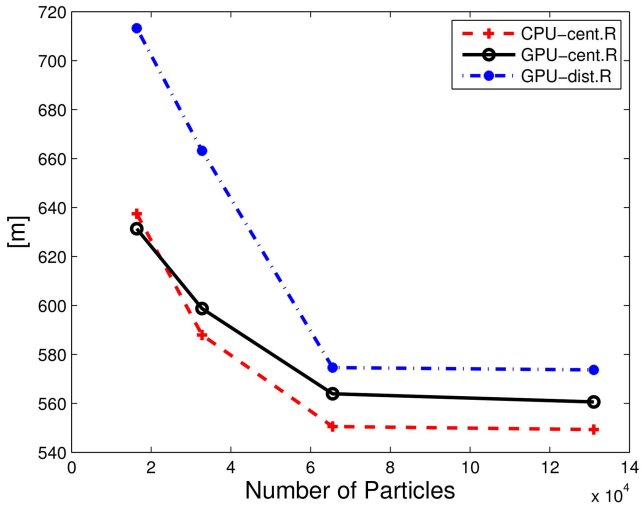


Fig. 7. Position TARMSE; 3 Targets; 100 MC Runs

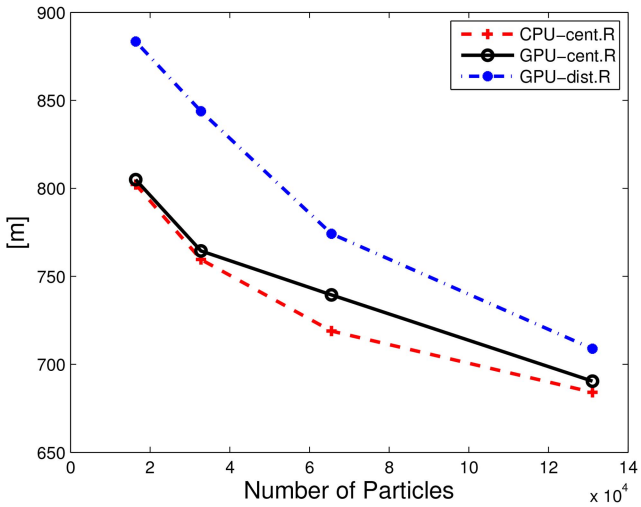


Fig. 9. Position TARMSE; 20 Targets; 100 MC Runs

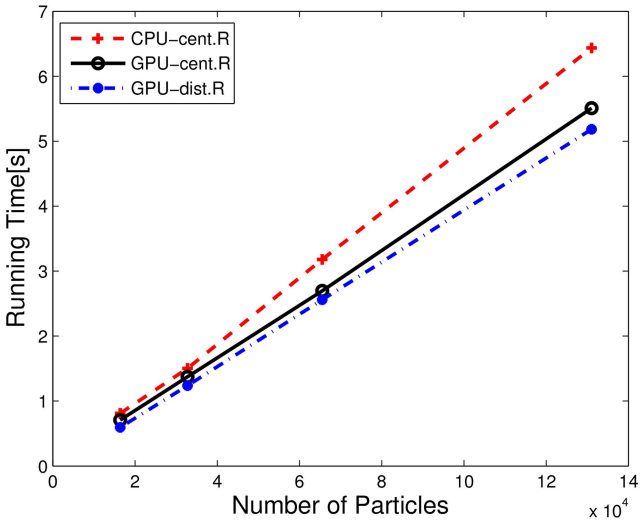


Fig. 8. Average Execution Time; 3 Targets; 100 MC Runs

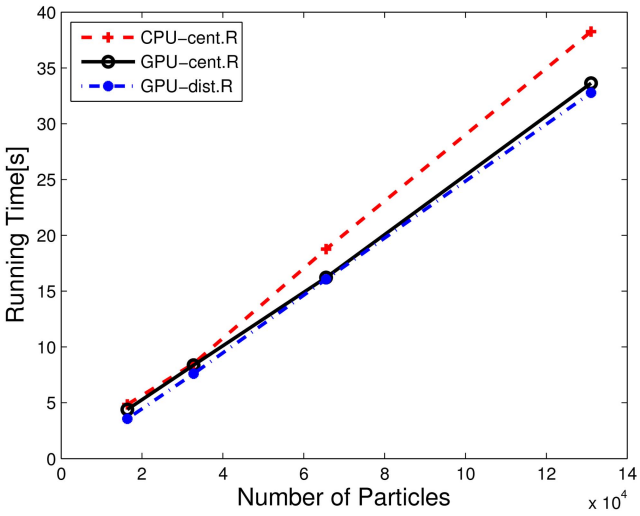


Fig. 10. Average Execution Time; 20 Targets; 100 MC Runs

illustrate that CPU-CR and GPU-CR are better than GPU-DR in terms of accuracy (for the same number of particles). This is clear because CR has better utilization of particles than DR—the former implements the resampling exactly as opposed to the latter which is approximate.

On the other hand, Fig. 8 and Fig. 10 show the execution times for one computational cycle of the tracking filter. Now the order of performance is reversed with CPU-CR being considerably slower than both GPU-CR and GPU-DR (which are close in computation times). Quantitatively (based on all simulations with 130K particles), CPU-CR is about 30% slower than GPU-DR.

The fully centralized algorithm (CPU-CR) is the best in terms of accuracy at a given number of particles but its computation time is worst. The fully distributed algorithm (GPU-DR) has shown the best running time but its accuracy is the worst. The partially distributed algorithm (GPU-CR), for the considered scenarios, has shown a computation time close to that of the fully distributed (GPU-DR) and accuracy not much worse than that of the fully centralized (CPU-CR). It appears that, for the considered hardware configuration, the partially distributed implementation may provide a reasonable tradeoff between filter accuracy and computation time as compared to the other two implementations.

2) Experiment 2:

a) Algorithms: Three PFs (CPU-CR, GPU-DR, and GPU-DR (new)) were implemented with different number of particles for each scenario. CPU-CR and GPU-DR were the same as described in Experiment 1. GPU-DR (new) implemented the novel enhanced PPF proposed in Sect. III-B

- *GPU-DR (new):* Both IS & R are performed in parallel on the GPU. DR is as described for the enhanced PPF in Table IV.

b) Computing Platform: The hardware used in this experiment is presented in Table VII.

c) Results & Comparison: Here we evaluate the novel EPPF proposed in Sect. III-B, Table IV in comparison

TABLE VII
Hardware Configuration 2

CPU	Model:	Intel(R) Core(TM) i5-3210M
	Clock Rate:	2.50 GHz
	Memory:	4.0 G
	Operating System:	Windows 7
GPU	Model:	NVS 5400M
	CUDA Driver:	5.0
	Clock Rate:	0.95 GHz
	Cores:	2 (MP) x 48 (Cores/MP) = 96
	Registers:	32K bytes/block
	Constant Memory:	64K bytes
	Shared Memory:	48K bytes/block

with CPU-CR and GPU-DR, described in Sect. III-A, Table III.

Fig 11 shows the computation time of the CPU-centralized resampling (cent.R), GPU-distributed resampling with RNA (dist.R (RNA)) and our enhanced algorithm (dist.R (new)). It is seen that for different number of particles our updated algorithm has better performance than that of the generic algorithms. The new algorithm is up to 1.5 times faster than dist.R (RNA). Fig. 12 shows relative times spent in the different steps of three PF algorithms. The resampling step of dist.R (new) has almost the same computational cost as the step of importance sampling. It is a significant improvement of the resampling step as compared with cent.R and dist.R (RNA).

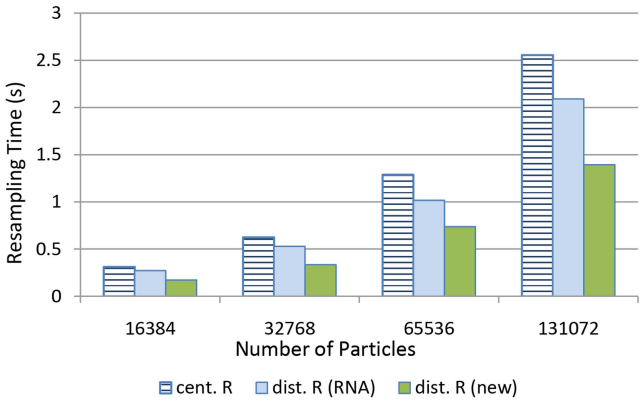


Fig. 11. Computation Times of CR & DR

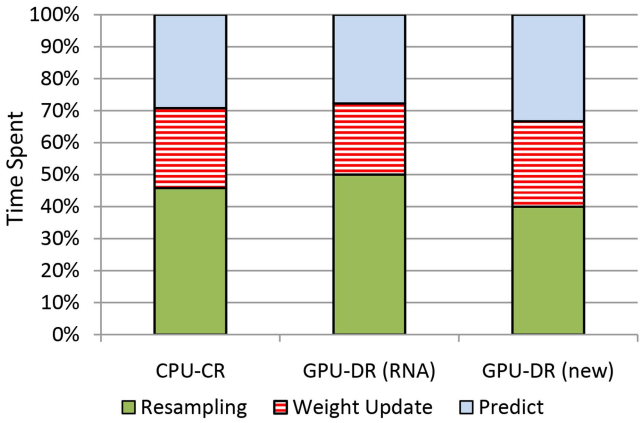


Fig. 12. Relative Times Spent in the Different Steps Using GPU-DR

V. SIMULATION STUDY II: GPU PPF VS. PFF FOR HIGH-DIMENSIONAL FILTERING PROBLEM

The purpose of this simulation study is to evaluate and compare the performances of the GPU-accelerated parallel PF (Table III) and PFF (Table V) for a high dimensional nonlinear filtering problem.

A. Model

We consider nonlinear filtering for the following model with cubic measurement nonlinearities[4]

$$x_{k+1} = \Phi x_k + w_k \tag{5}$$

$$z_k = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathsf{x}_{k,1}^3 \\ \mathsf{x}_{k,2}^3 \\ \mathsf{x}_{k,3}^3 \end{bmatrix} + v_k \tag{6}$$

where $k = 0, 1, \ldots$ is time index, $x_k = [\mathsf{x}_{k,1}\ \mathsf{x}_{k,2}\ \ldots, \mathsf{x}_{k,d}]'$ is the state vector of dimension $d \geq 3$, Φ is a positive definite transition matrix that is generated randomly for each scenario in the simulation, $w_k \sim \mathcal{N}(0, 0.04^2 I_d)$ and $v_k \sim \mathcal{N}(0, 1.0^2 I_3)$ are zero-mean Gaussian white process and measurement noises, respectively, and the initial state vector $x_0 \sim \mathcal{N}(0.8, 25000 I_d)$ is randomly generated.

Four scenarios with different state dimension (i.e., $d = 10, 20, 30, 40$), each with different number of particles as specified in Table VIII, are simulated.

B. Algorithms

Implemented are the following four filter algorithms: PFF-CPU, PFF-GPU, PF-CPU and PF-GPU where CPU stands for a sequential implementation (needed for the purposes of comparison). The hardware used in our experiments is presented in Table VII. The code for all implementations is written in C++ and compiled using Visual C++ 2008. The particular configuration parameters of our GPU implementations are given in Table VIII.

C. Performance Measures

In order to obtain statistically significant evaluation of the performance metrics, $R = 50$ Monte Carlo (MC) runs are performed for each scenario.

The filters' estimation accuracy at each time step k is measured in terms of the average dimension-free error

$$e_k = \frac{1}{R} \sum_{r=1}^{R} e_k^{(r)}, \tag{7}$$

[4]The target dynamics need not be nonlinear for the purpose of comparison because both nonlinear filters, PF and PFF, differ only in the measurement update part.

TABLE VIII
GPU Blocks' Specification

Num of Particles =		1536	12288	49152	98304	196608	393216	786432	1572864
XDim = 10	Threads/Block = 256	6	48	192	384	768	1536	3072	6144
XDim = 20	Threads/Block = 128	12	96	384	768	1536	3072	6144	12288
XDim = 30	Threads/Block = 96	16	128	512	1024	2048	4096	8192	16384
XDim = 40	Threads/Block = 64	24	192	768	1536	3072	6144	12288	24576

where

$$e_k^{(r)} = \frac{1}{d}(\hat{x}_k^{(r)} - x_k^{(r)})'(\hat{x}_k^{(r)} - x_k^{(r)}) \tag{8}$$

and $x_k^{(r)}$ and $\hat{x}_k^{(r)}$ are the true and estimated state vectors, respectively, in MC run $r = 1,2,\ldots,R$.

The filters' overall accuracy is measured in terms of the time-averaged error (TAE), defined as follows

$$\varepsilon_{[m,n]} = \frac{1}{n-m+1}\sum_{k=m}^{n} e_k \tag{9}$$

where $[m,n]$ is the time interval of averaging. TAE is sometimes used in target tracking as a single measure of "steady-state" filter accuracy. In the simulation $m = 21$, $n = 50$.

The computational performance of all four filters is measured in terms of average running time per one filter time-step. The computational performance of the parallel GPU filters is measured in terms of speedup with respect to the corresponding CPU sequential filter, i.e.,

$$Speedup = \frac{T_{CPU}}{T_{GPU}} \tag{10}$$

where T_{CPU} and T_{GPU} denote the average running time of the filter (PF or PFF) on CPU and GPU, respectively. The speedup characterizes the scalability of a parallel algorithm.

D. Results

Fig. 13 shows the dimension-free error plots of the filters[5] with 10 dimensional state vectors and 800K particles. It illustrates that PFF-CPU is the best in terms of accuracy (for the same number of particles), followed by PFF-GPU, PF-CPU and PF-GPU. Very similar results were obtained for different state vector dimensions (up to 40) and different number of particles. Based on all results, the GPU versions of both PFF and PF are apparently less accurate than their corresponding CPU versions. This is because the parallel GPU versions are actually approximations of the fully centralized CPU versions. For the PF this approximation is in the resampling step: it is global (uses all particles) in PF-CPU and local (uses only the particles within a block) in PF-GPU. A similar effect happens with the PFF: the computation of $\bar{x}_k$ in PFF-CPU is global (the sample mean of all particles), while the computation of $\bar{x}_{k,b}$ in PFF-GPU is local and consequently less accurate.

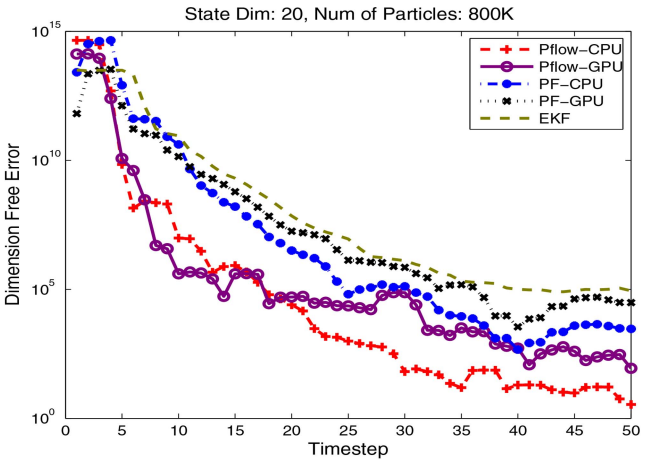


Fig. 13. Dimension-Free Errors

More detailed overall accuracy comparison can be made based on the plots in Fig. 14 (left) that show the time-averaged dimension-free errors of the four filters with different dimensions of the state vector versus the number of particles. The differences in accuracy are quite significant. In particular, PFF-GPU is several orders of magnitude more accurate than both PF-GPU and PF-CPU. Of course, PFF-GPU is less accurate than PFF-CPU (for reasons explained above), but this does not seem significant, given the fact (discussed next) that the former is much faster than the latter.

The plots in Fig. 14 (right) show the average running times—the "prices" paid to achieve the accuracies shown in the corresponding plots of Fig. 14 (left). With the same number of particles, PFF-CPU is about 1.5–2 times faster than PF-CPU (depending on the number of particles), and PFF-GPU is about 4–5 times faster than PF-GPU. Table IX is provided for more accurate comparison. Even though it might seem that PFF-CPU requires more computation time per particle than PF-CPU, it actually appears otherwise. PFF update includes computing $A(\lambda)$, $b(\lambda)$ and solving the ODE via the Euler scheme (Table V). Computing A, b is common for all particles and independent from the number of particles (except for $\bar{x}_k$). Therefore, for large number of particles it has insignificant contribution to the "per particle" computation time. The main portion of computation time per particle is spent on the ODE which is a fairly simple and fast computation for small L. In the simulation $L = 10$ and the time spent on it is up to twice

[5]EKF's error plot is shown as a baseline only and is excluded from further comparison.

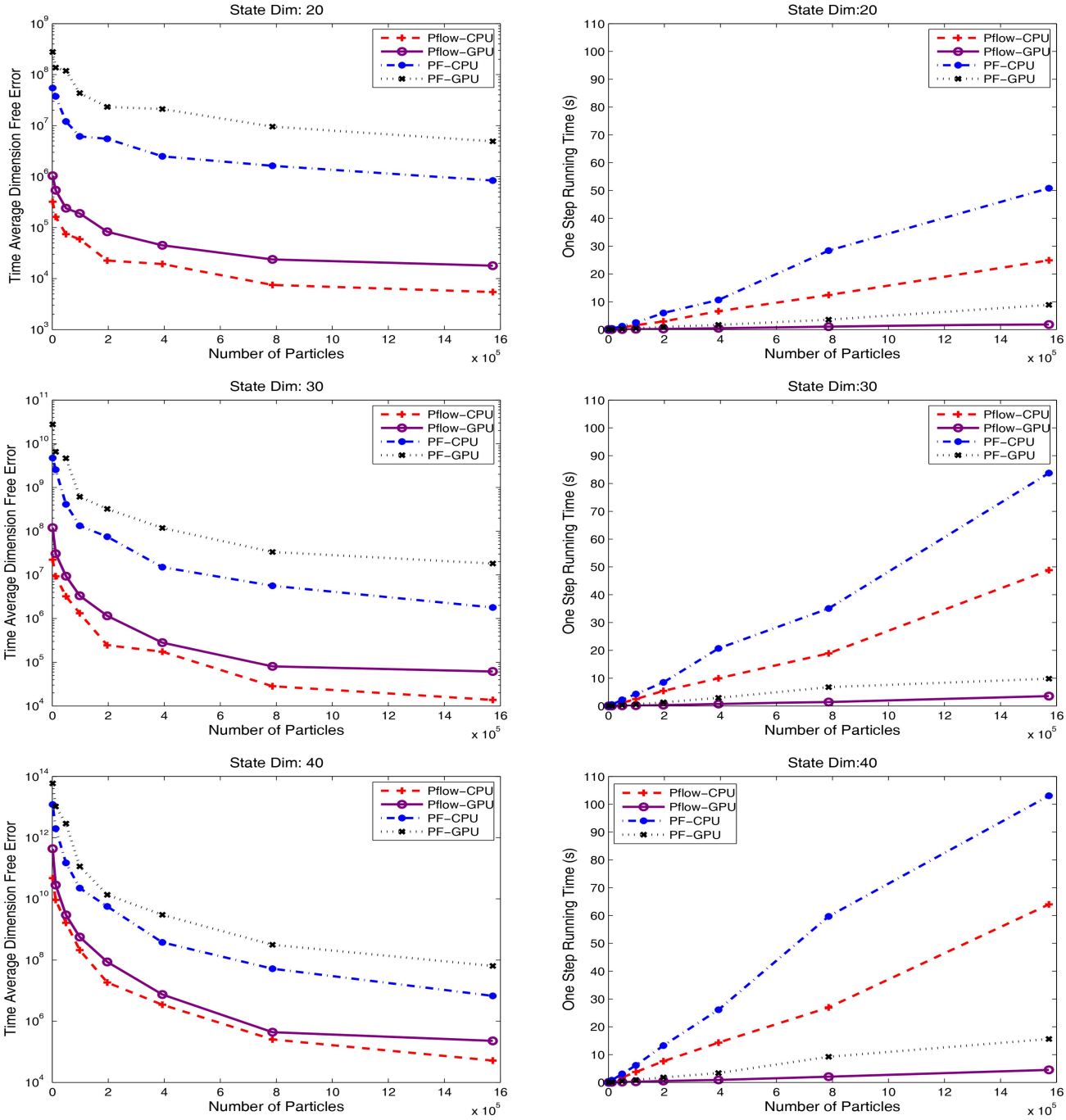


Fig. 14. Time-Averaged Dimension-Free Error (left) & Running Time (right)

smaller than the time PF spends to draw a resampled particle from a large set of particles using stratified sampling. PFF with $L = 5$ and 20 was also run, but $L = 10$ was chosen as the best tradeoff between accuracy and speed. For $L = 20$ the computation times of PFF-CPU and PF-CPU are closer but the advantage of PFF in accuracy increases. The time results regarding PFF-GPU vs. PF-GPU are not surprising given the fact that PFF-GPU is almost completely (thread) parallel while the resampling of PF-GPU is only partially (block) parallel.

Fig. 15 illustrates the effect of the state dimension on the running time with different number of particles of both parallel filters: PF-GPU (left) and PFF-GPU (right).

Fig. 16 shows the speedup of PF-GPU (left) and PFF-GPU (right) with different state dimension and different number of particles. It appears that for both GPU filters the speedup most often increases with the state dimension (for the same number of particles) which supports using GPU for highly dimensional problems. On the other hand, the speedup for both GPU filters does not seem to vary very significantly with the number of particles (for the same state dimension). Finally, Fig. 17 compares the speedup of PF-GPU and PFF-GPU for

TABLE IX
Running Time (s)

Num of Particles =		1536	12288	49152	98304	196608	393216	786432	1572864
XDim = 10	PFF-CPU	0.0099	0.0792	0.3403	0.6682	1.2630	3.1733	5.0851	14.4671
	PFF-GPU	0.0007	0.0054	0.0218	0.0439	0.1000	0.2187	0.4625	1.0493
	PF-CPU	0.0172	0.1418	0.5856	1.2602	2.5559	4.6397	9.1571	17.8682
	PF-GPU	0.0034	0.0281	0.1129	0.2438	0.5337	1.0696	1.7700	5.5668
XDim = 20	PFF-CPU	0.0218	0.1760	0.7338	1.5952	2.9421	6.6210	12.4409	24.9088
	PFF-GPU	0.0014	0.0117	0.0469	0.0989	0.2175	0.4928	1.0343	1.7933
	PF-CPU	0.0382	0.3139	1.2319	2.5249	5.9612	10.6548	28.3821	50.8195
	PF-GPU	0.0063	0.0513	0.2023	0.4362	0.9371	1.6812	3.5717	8.8979
XDim = 30	PFF-CPU	0.0359	0.2925	1.1410	2.4776	5.4436	9.9433	18.9128	48.8515
	PFF-GPU	0.0023	0.0181	0.0722	0.1464	0.2984	0.6978	1.4003	3.5066
	PF-CPU	0.0643	0.5213	2.2102	4.2976	8.4932	20.7041	35.0581	83.7303
	PF-GPU	0.0092	0.0743	0.3132	0.5864	1.3253	2.9221	6.7780	9.8334
XDim = 40	PFF-CPU	0.0520	0.4127	1.7799	3.7476	7.6032	14.2963	26.9597	64.0159
	PFF-GPU	0.0033	0.0272	0.1092	0.2148	0.5102	0.8640	2.0259	4.4173
	PF-CPU	0.0947	0.7756	3.0457	6.1521	13.2333	26.1284	59.6716	103.0515
	PF-GPU	0.0124	0.0996	0.4036	0.8578	1.7792	3.3625	9.2275	15.5961

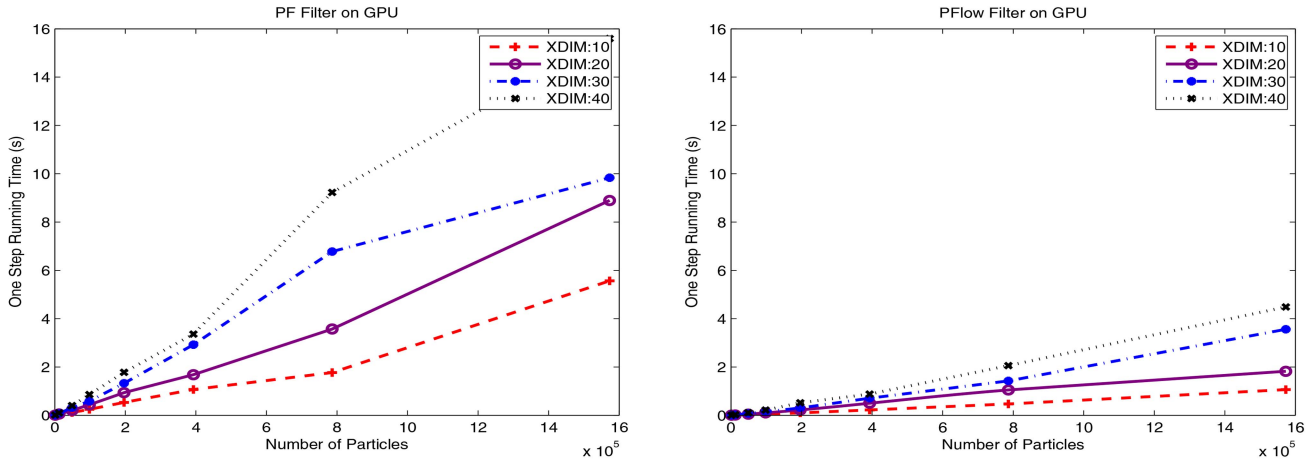


Fig. 15. One-Step Running Time

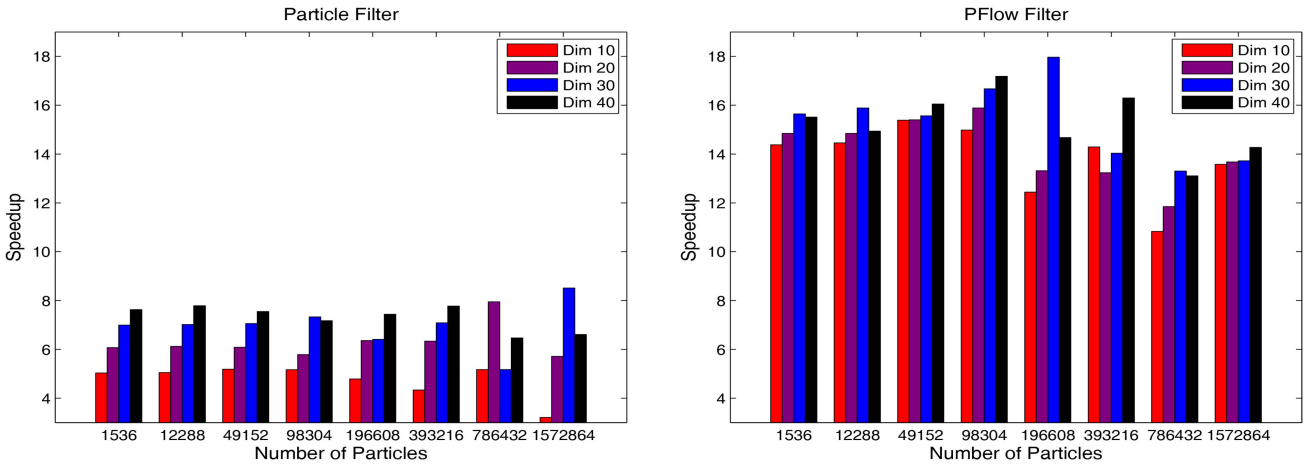


Fig. 16. Speedup of PF & PFF

different number of particles (for state dimension 40). PF-GPU provides a speedup of about six times with respect to PF-CPU and PFF-GPU provides a speedup of about fifteen times with respect to PF-CPU. PFF-GPU outperforms PF-GPU more than two times in terms of speedup due to its higher level of parallelization.

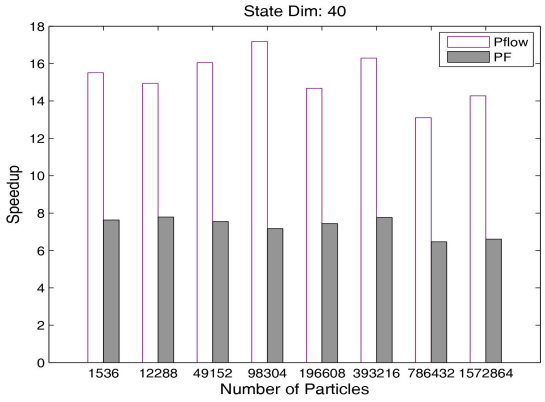


Fig. 17. Speedup of PF *vs.* PFF

VI. SUMMARY & CONCLUSIONS

Three efficient parallel particle and particle flow filter implementations, optimized for GPU architecture, have been proposed and studied. They have been applied and tested, via simulation, for tracking multiple targets using a pixelized sensor, and for a high-dimensional nonlinear density estimation problem.

Overall, the obtained simulation results have demonstrated that using GPU can significantly accelerate the computation of particle filters and particle flow filters through parallelization of the computational algorithms at a tolerable loss of accuracy, and thereby bring them closer to practical applications.

Specifically:

- For the multitarget target tracking problem, the newly proposed Enhanced PPF GPU implementation has shown superior computational performance and the same accuracy as compared to the previous (RNA-based) PPF implementation.
- For the high-dimensional nonlinear estimation problem, the parallel particle flow filter has shown superior performance in comparison with the parallel particle filter implementation in both estimation accuracy and computational efficiency.

ACKNOWLEDGMENT

The authors would like to thank Fred Daum and Jim Huang for providing the relevant papers on particle flow and keeping us updated on their latest results.

REFERENCES

[1] N. Gordon, D. Salmond, and A. Smith
"Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation,"
IEE Proceedings-F., vol. 140, no. 2, pp. 107–113, April 1993.

[2] A. Doucet, N. de Frietas, and N. Gordon, Eds.
Sequential Monte Carlo Methods in Practice,
ser. Statistics for Engineering and Information Science. New York: Springer-Verlag, 2001.

[3] B. Ristic, S. Arulampalam, and N. Gordon
Beyond the Kalman Filter. Particle Filters for Tracking Applications.
Artech House, 2004.

[4] A. S. Bashi, V. P. Jilkov, X. R. Li, and H. Chen
"Distributed Implementations of Particle Filters,"
in *Proc. 2003 International Conf. on Information Fusion*, Cairns, Australia, July 2003, pp. 1164–1171.

[5] V. Teulière and O. Brun
"Parallelisation of the Particle Filtering Technique and Application to Doppler-Bearing Tracking of Maneuvering Sources,"
Parallel Computing, vol. 29, no. 8, pp. 1069–1090, 2003.

[6] S. Sutharsan, A. Sinha, T. Kirubarajan, and M. Farooq
"An Optimization-Based Parallel Particle Filter for Multitarget Tracking,"
in *Proc. 2005 SPIE Conf. on Signal and Data Processing of Small Targets*, vol. 5913, San Diego, CA, Aug. 2005.

[7] M. Bolic, P. M. Djuric, and S. Hong
"Resampling Algorithms and Architectures for Distributed Particle Filters,"
IEEE Trans. Signal Processing, vol. 53, no. 7, pp. 2442–2450, Jul. 2005.

[8] G. Hendeby, J. D. Hol, R. Karlsson, and F. Gustafsson
"A Graphics Processing Unit Implementation of the Particle Filter,"
in *Proc. 15th European Signal Processing Conference*, 2007.

[9] A. Lee, C. Yau, M. B. Giles, A. Doucet, and C. C. Holmes
"On the Utility of Graphics Cards to Perform Massively Parallel Simulation of Advanced Monte Carlo Methods,"
Journal of Computational and Graphical Statistics, vol. 19, no. 4, pp. 769–789, 2010.

[10] G. Hendeby, R. Karlsson, and F. Gustafsson
"Particle Filtering: The Need for Speed,"
EURASIP Journal on Advances in Signal Processing, no. 181403, 2010.

[11] V. P. Jilkov, J. Wu, and H. Chen
"Parallel Particle Filter Implementation on Cluster Computer for Satellite Tracking Application,"
in *Proc. 4th International Conference on Neural, Parallel & Scientific Computations*, Atlanta, GA, USA, August 2010, pp. 179–186.

[12] V. P. Jilkov and J. Wu
"Implementation and Performance of a Parallel Multitarget Tracking Particle Filter,"
in *Proc. 2011 International Conf. on Information Fusion*, Chicago, IL, USA, July 2011, pp. 298–305.

[13] J. Wu and V. P. Jilkov
"Parallel Multitarget Tracking Particle Filters using Graphics Processing Unit,"
in *Proc. 44th IEEE Southeastern Symposium on System Theory*, Jacksonville, FL, USA, March 2012, pp. 151–155.

[14] V. P. Jilkov, J. Wu, and H. Chen
"Performance Comparison of GPU-Accelerated Particle Flow and Particle Filters,"
in *Proc. 2013 International Conf. on Information Fusion*, Istanbul, Turkey, July 2013.

[15] J. Wu
"Parallel Computing of Particle Filtering Algorithms for Target Tracking Applications,"
Ph.D. dissertation, University of New Orleans, 2014.

[16] F. Daum and J. Huang
"Nonlinear Filters with Log-Homotopy,"
in *Proc. 2007 SPIE Conf. on Signal and Data Processing of Small Targets*, vol. 6699, no. 669918, 2007.

[17] F. Daum, J. Huang, M. Krichman, and T. Kohen
"Seventeen Dubious Methods to Approximate the Gradient for Nonlinear Filters with Particle Flow,"
in *Proc. 2009 SPIE Conf. on Signal and Data Processing of Small Targets*, vol. 7445, no. 74450V, 2009.

[18] F. Daum and J. Huang
"Numerical Experiments for Nonlinear Filters with Exact Particle Flow Induced by Log-Homotopy,"
in *Proc. 2010 SPIE Conf. on Signal and Data Processing of Small Targets*, vol. 7698, no. 769816, 2010.

[19] ———
"Exact Particle Flow for Nonlinear Filters,"
in *Proc. SPIE Conf. on Signal Processing, Sensor Fusion, and Target Recognition XIX*, vol. 7697, no. 769704, 2010.

[20] ———
"Exact Particle Flow For Nonlinear Filters: Seventeen Dubious Solutions to a First Order Linear Underdetermined PDE,"
in *Proc. of the 44th Asilomar Conference on Signals, Systems and Computers*, November 2010, pp. 64–71.

[21] ———
"Particle Degeneracy: Root Cause and Solution,"
in *Proc. SPIE Conf. on Signal Processing, Sensor Fusion, and Target Recognition XX*, vol. 8050, no. 80500W, 2011.

[22] ———
"Hollywood Log-Homotopy: Movies of Particle Flow for Nonlinear Filters,"
in *Proc. SPIE Conf. on Signal Processing, Sensor Fusion, and Target Recognition XX*, vol. 8050, no. 80500X, 2011.

[23] X. R. Li and V. P. Jilkov
"A Survey of Maneuvering Target Tracking—Part VIb: Approximate Nonlinear Density Filtering in Discrete Time,"
in *Proc. 2012 SPIE Conf. on Signal and Data Processing of Small Targets*, vol. 8393, Baltimore, MD, USA, Apr. 2012.

[24] M. D. McCool
"Signal Processing and General-Purpose Computing and GPUs [Exploratory DSP],"
IEEE Signal Processing Magazine, vol. 24, no. 3, pp. 109–114, May 2007.

[25] X. R. Li and V. P. Jilkov
"A Survey of Maneuvering Target Tracking—Part VIa: Density-Based Exact Nonlinear Filtering,"
in *Proc. 2010 SPIE Conf. on Signal and Data Processing of Small Targets*, vol. 7698, Orlando, FL, USA, Apr. 2010.

[26] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp
"A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking,"
IEEE Trans. Signal Processing, vol. 50, no. 2, pp. 174–188, 2002.

[27] O. C. Schrempf and U. D. Hanebeck
"Recursive Prediction of Stochastic Nonlinear Systems Based on Optimal Dirac Mixture Approximations,"
in *Proc. American Control Conference*, 2007.

[28] S. Chikkagoudar, K. Wang, and M. Li
"GENIE: A Software Package for Gene-Gene Interaction Analysis in Genetic Association Studies Using Multiple GPU,"
BMC Research Notes, vol. 4:158, 2011.

[29] NVIDIA
CUDA C Programming Best Practices Guide,
2012. [Online]. Available: http://docs.nvidia.com/cuda.

[30] ———
CUDA CURAND Library,
2010. [Online]. Available: http://docs.nvidia.com/cuda.

[31] W. D. Hillis and G. L. Steele Jr
"Data Parallel Algorithms,"
Communications of the ACM, vol. 29, no. 12, pp. 1170–1183, 1986.

[32] C. Kreucher, K. Kastella, and A. Hero
"Tracking Multiple Targets using a Particle Filter Representation of the Joint Multitarget Probability Density,"
in *Proc. SPIE: Signal and Data Processing of Small Targets*, O. Drummond, Ed., no. 5204, 2003, pp. 258–269.

[33] C. Kreucher, M. Morelande, K. Kastella, and A. Hero
"Particle Filtering for Multitarget Detection and Tracking,"
in *2005 IEEE Aerospace Conference*, March 2005, pp. 2101–2116.

[34] C. Kreucher, K. Kastella, and A. O. Hero
"Multitarget Tracking Using the Joint Multitarget Probability Density,"
IEEE Trans. Aerospace and Electronic Systems, vol. 41, no. 4, pp. 1396–1414, Oct. 2005.

[35] M. Morelande, C. Kreucher, and K. Kastella
"A Study of Factors in Multiple Target Tracking with a Pixelized Sensor,"
in *Proc. SPIE: Signal and Data Processing of Small Targets*, O. Drummond, Ed., vol. 5913, 2005, pp. 155–167.

[36] ———
"A Bayesian Approach to Multiple Target Detection and Tracking,"
IEEE Trans. on Signal Processing, vol. 55, no. 5, pp. 1589–1604, 2007.

[37] C. Kreucher, A. Hero, K. Kastella, and M. Morelande
"An Information-Based Approach to Sensor Management in Large Dynamic Networks,"
Proceedings of the IEEE, vol. 95, no. 5, pp. 978–999, May 2007.

[38] Y. Boers, E. Sviestins, and H. Driessen
"Mixed Labelling in Multitarget Particle Filtering,"
IEEE Trans. on Aerospace and Electronic Systems, vol. 46, no. 2, pp. 792–802, 2010.

Vesselin P. Jilkov received the B.S./M.S. degree in Mathematics from the University of Sofia, Bulgaria in 1982, the Ph.D. degree in the Technical Sciences (Electrical Engineering) in 1988, and the academic rank Senior Research Fellow of the Bulgarian Academy of Sciences in 1997.

From 1982 to 1988, he was a research scientist with the R&D Institute of Special Electronics, Sofia, and, from 1988 to 1999, with the Institute for Parallel Processing, Bulgarian Academy of Sciences. Since 1999, he has been with the Department of Electrical Engineering, University of New Orleans, New Orleans, LA, where he is, at present, an Associate Professor and Riley/Ruby Parker Endowed Professor. His main research interests include stochastic systems, nonlinear filtering, applied decision, estimation & control, target tracking, information fusion, parallel processing.

Dr. Jilkov is author/coauthor of one book and over 100 journal articles and conference papers. He is a member of IEEE, ISIF, and SIAM.

Jiande Wu received the B.E. and M.S. degrees in Computer Science and Technology from the North China Electric Power University, Beijing, China in 1998 and 2005, respectively. He received the M.S. and Ph.D. degrees in Electrical Engineering from the University of New Orleans, New Orleans, LA, USA in 2012 and 2014, respectively.

His research interests include parallel computing, Monte Carlo methods, applied decision and estimation, and nonlinear filtering.

JAIF

Journal of Advances in Information Fusion

A semi-annual archival publication of the
International Society of Information Fusion

Volumes 1–9 Index

SUBJECT INDEX

INTERNATIONAL SOCIETY OF INFORMATION FUSION

ISIF Website: http://www.isif.org

2015 BOARD OF DIRECTORS*

2013–2015	2014–2016	2015–2017
Jean Dezert	Sten F. Andler	Joachim Biermann
Gee-Wah Ng	Murat Efe	Frederik Gustafsson
Anne-Laure Jousselme	Lyudmila Mihaylova	Jesús García

*Board of Directors are elected by the members of ISIF for a three year term.

PAST PRESIDENTS

Darin Dunham, 2014	Darko Musicki, 2008	Yaakov Bar-Shalom, 2002
Wolfgang Koch, 2013	Erik Blasch, 2007	Pramod Varshney, 2001
Roy Streit, 2012	Pierre Valin, 2006	Yaakov Bar-Shalom, 2000
Joachim Biermann, 2011	W. Dale Blair, 2005	Jim Llinas, 1999
Stefano Coraluppi, 2010	Chee Chong, 2004	Jim Llinas, 1998
Elisa Shahbazian, 2009	Xiao-Rong Li, 2003	

SOCIETY VISION

The International Society of Information Fusion (ISIF) is the premier professional society and global information resource for multidisciplinary approaches for theoretical and applied information fusion technologies.

SOCIETY MISSION

Advocate

To advance the profession of fusion technologies, propose approaches for solving real-world problems, recognize emerging technologies, and foster the transfer of information.

Serve

To serve its members and engineering, business, and scientific communities by providing high-quality information, educational products, and services.

Communicate

To create international communication forums and hold international conferences in countries that provide for interaction of members of fusion communities with each other, with those in other disciplines, and with those in industry and academia.

Educate

To promote undergraduate and graduate education related to information fusion technologies at universities around the world. Sponsor educational courses and tutorials at conferences.

Integrate

Integrate ideas from various approaches for information fusion, and look for common threads and themes–look for overall principles, rather than a multitude of point solutions. Serve as the central focus for coordinating the activities of world-wide information fusion related societies or organizations. Serve as a professional liaison to industry, academia, and government.

Disseminate

To propagate the ideas for integrated approaches to information fusion so that others can build on them in both industry and academia.

Call for Papers

The Journal of Advances in Information Fusion (JAIF) seeks original contributions in the technical areas of research related to information fusion. Authors are encouraged to submit their manuscripts for peer review http://isif.org/journal.

Call for Reviewers

The success of JAIF and its value to the research community is strongly dependent on the quality of its peer review process. Researchers in the technical areas related to information fusion are encouraged to register as a reviewer for JAIF at http://jaif.msubmit.net. Potential reviewers should notify via email the appropriate editors of their offer to serve as a reviewer.

Journal of Advances in Information Fusion Volume 10 Number 1 June 2015 ISSN 1557-6418