Performance Evaluation of Simultaneous Sensor Registration and Object Tracking Algorithm

Sofie Macdonald, Ian Proudler, Michael E. Davies, and James R. Hopgood

Abstract—Reliable object tracking with multiple sensors requires that sensors are registered correctly with respect to each other. When an environment is Global Navigation Satellite System (GNSS) denied or limited – such as underwater, or in hostile regions – this task is more challenging. This paper presents uncertainty quantification on a simultaneous tracking and registration algorithm for sensor networks that does not require access to a GNSS. The method uses a particle filter combined with a bank of augmented state extended Kalman filters (EKFs). The particles represent hypotheses of registration errors between sensors, with associated weights. The EKFs are responsible for the tracking procedure and for contributing to particle state and weight updates. This is achieved through the evaluation of a likelihood. Registration errors in this paper are spatial, orientation, and temporal biases: seven distinct sensor errors are estimated alongside the tracking procedure. Monte Carlo trials are conducted for the uncertainty quantification. Since performance of particle filters is dependent on initialisation, a comparison is made between more and less favourable particle (hypothesis) initialisation. The results demonstrate the importance of initialisation, and the method is shown to perform well in tracking a fast (marginally sub-sonic) object following a bow-like trajectory (mimicking a representative scenario). Final results show the algorithm is capable of achieving angular bias estimation error of 0.0034°, temporal bias estimation error of 0.0067 s, and spatial error of 0.021 m.

I. INTRODUCTION

A. Problem Overview

Sensor registration is an important capability for sensor networks [1]. Ensuring that the reference frames (RFs) of sensors are aligned – in space or time [2] – is key to a network’s reliability. If sensors can’t be accurately calibrated relative to each other then their gathered data can’t be usefully compared or combined. In the object tracking case, failure to calibrate sensors may lead to generation of false tracks or termination of true tracks [3]. In an emergency situation or hostile environment this could mean disaster. Calibrating sensors before deployment is not sufficient to ensure they remain calibrated for the duration of their commission. Effects of harsh weather, platform vibrations, incorrect initial calibration, as well as loss of Global Positioning System (GPS) access are all sources of potential error. Temporal bias can originate from: differences in sensors’ internal processing time, data collection rates, transmission time, or effects of CPU load. Therefore, a sensor network must be able to continuously self-localise, or perform ad-hoc self-localisation, to mitigate these errors, lest its data become meaningless. In this paper, three types of registration error (subsequently called bias) are considered: spatial, angular, and temporal. This means that there are seven distinct parameters to estimate per non-calibrated sensor: $x$, $y$, $z$, $\tau$ (time delay), $\alpha$ (yaw), $\beta$ (pitch), $\gamma$ (roll). Sensor registration is achieved with non-cooperative targets and the network operates in a centralised manner with a plot fusion architecture. The bias estimation procedure uses a hierarchical Bayesian model (HBM) [4] (suited to Bayesian inference problems) which manages the joint tracking and sensor registration – confirmed by extensive simulation. The HBM also ensures an adaptive setup. That is, the extended Kalman filter (EKF) used in the tracking procedure can be swapped out for any other filter.

B. State of the Art

The sensor registration problem has been considered previously [5]. A range of methods have been employed to approach the problem. In Sigalov et al. [6], the problem is also considered in the target tracking context. The authors present an algorithm for calibration of multiple sensors utilising targets of opportunity – i.e. targets which are observable to sensors but have no action scheduled. Their results show that in estimation of sensor rotational bias (yaw, pitch, roll) they come within 1.5 mRad (0.09°) of the true values. They also show that this degrades with increasing measurement noise. The team recently revised this work in [7] where they have stated that the updated version guarantees convergence in three dimensional scenarios. The maximum error on angular bias estimation of the newest algorithm is 0.013°: to be compared with a maximum error of 0.08° in their previous paper. Alignment of sensor reference frames in multi-platform, multi-sensor systems is also considered in [8]. They align sensors with what they call a global sensor which they assume is free of bias: the same approach taken in their previous paper. A method based on the unit quaternion is proposed for the estimation of rotation and spatial bias of the sensors in space. The mean error on the spatial bias estimation falls within 2 m and for angular bias estimation the mean error falls within 0.02° of the truth. Pu et al. [9] consider the estimation of sensor biases in multisensor systems given noisy and asynchronous measurements. They address this problem with a nonlinear least squares formulation and a reference target moving with unknown constant...
This section gives an overview of the hierarchical Bayesian model (HBM) used by the bias estimation algorithm in this paper. HBMs allow complex models to be broken down into series of simpler models. They have already been shown to be useful in joint tracking and registration [1]. In this paper, the HBM consists of a parent and an offspring process (Fig. 1). The chosen architecture for the centralised sensor network is plot fusion. In plot fusion, the sensors’ collected data is merged and processed by the same filter.

A. Parent Process: Parameter Estimation

The parent process carries out the bias parameter estimation. The registration errors are: \(x, y, z, \alpha, \beta, \gamma\). Here, \(x, y, z\) correspond to spatial offsets; \(\tau\) is a fixed time delay and \(\alpha, \beta, \gamma\) correspond to angles of yaw, pitch, and roll, respectively. These errors are all relative to a reference point: in this case, a reference sensor. With the particle filter approach, particles correspond to bias hypotheses. Therefore, for uncalibrated sensor \(s\) at timestep \(k\) the space of hypotheses is given by the following set of \(N\) particles: \(\Psi_{k,s} = \{\psi_{k,s}^1, \ldots, \psi_{k,s}^N\}\).

The particle state \(\psi_{k,s}\) is a vector of seven dimensions: \(\left[\bar{x}_{k,s}^1, \bar{y}_{k,s}^1, \bar{z}_{k,s}^1, \bar{\alpha}_{k,s}^1, \bar{\beta}_{k,s}^1, \bar{\gamma}_{k,s}^1\right]^T\) associated with weight \(w_{k,s}^1\). The particle weight is a reflection of the belief that the particle state (sensor registration error) is close to the truth. The set of hypotheses and their weights is updated and therefore vary in time. Following evaluation of effective sample size and comparison with a threshold, particles may be resampled and propagated. A multinomial resampling strategy has been selected for this work. The initial set of hypotheses are drawn from uniform distributions, \(U[0, b]\), where the lower and upper bounds (\(a\) and \(b\), respectively) are bias parameter dependent (Table I). It is assumed that all randomly drawn initial states are equally likely so the prior distribution is flat. Weights are predicted and updated recursively as in \([3]\) – see equations \((2a)\) and \((2b)\). The weights prediction can be understood as a convolution of prior weights with a kernel function. The kernel function for this work is the binomial distribution, \(B(n, p)\), where \(n\) and \(p\) are determined empirically and set to values of \(N\) and 0.5, respectively. The weights are updated with a likelihood function, \(\ell_k(\psi_{k,s}^i|Z_k)\), derived from the evaluation of the integral form of the Kalman likelihood conditioned on \(\psi_{k,s}^i\) – see Equation \((11)\) in \([16]\). Note that \(Z_k\) is the set of all sensor measurements up to time \(k\). In the case of the EKF:

\[
\ell_k(\psi_{k,s}^i|Z_k) = N(z_k|h(V_k^i), S_k^i)
\]

\(S_k^i\) is the EKF innovation covariance used to calculate the Kalman gain; \(h(\cdot)\) represents the Cartesian to spherical polar transformation; \(V_k^i\) is the object state prediction with bias hypotheses applied (see Section II-B.1 for further detail); \(z_k\) is the augmented measurement vector at timestep \(k\) – that is, it holds the measurements from all sensors at that time. The likelihood is an output of the offspring process. Following evaluation of the likelihood function, particle weights are updated according to Equation \((2b)\) in \([3]\).

C. Contributions

The algorithm investigated in this paper builds on that introduced in \([16]\). It extends the original algorithm by replacing the Kalman filter (KF) with an extended Kalman filter (EKF). The original grid-search technique for the bias estimation is replaced by a full particle filter method.

- Performance evaluation of the algorithm to estimate seven distinct sensor registration errors (spatial, temporal, angular) is performed through MC simulation.
- The accuracy of the algorithm is investigated in an artificial scenario that is generated to closely match a representative case: a fast-moving object following a bow-like trajectory.

Uncertainty quantification through MC trials allows assessment of the performance of the bias estimation algorithm and of the reliability of the corresponding tracking solution.
resulting in the observation matrix $H_k^i$ which has temporal and rotation bias hypothesis $i$ incorporated. Now the spatial bias hypothesis can be applied. Let the translation vector:

$$T_k^i = [\hat{x}_{k,1}^i, \hat{y}_{k,1}^i, \hat{z}_{k,1}^i, \cdots, \hat{x}_{k,S}^i, \hat{y}_{k,S}^i, \hat{z}_{k,S}^i]^T$$

be that associated with hypothesis $i$ at time $k$. This is applied to the object state prediction, $\hat{X}_k^i$, in the following manner:

$$V_k^i = H_k^i \hat{X}_k^i - T_k^i$$

We can calculate the Jacobian, $J_k^i$, for hypothesis $i$, time $k$:

$$J_k^i = \frac{\partial h(V_k^i)}{\partial X_k^i}$$

The evaluation of this derivative involves the chain rule:

$$\frac{\partial h(V_k^i)}{\partial V_k^i} \cdot \frac{\partial V_k^i}{\partial X_k^i}$$

Notice $\frac{\partial h(V_k^i)}{\partial V_k^i}$, the second term on the right hand side of Equation (11), can be immediately evaluated using Equation (9) using standard vector calculus results. In short: $\frac{\partial h(V_k^i)}{\partial V_k^i} = H_k^i$. Finally, $\frac{\partial V_k^i}{\partial V_k^i}$ is trivial but long-winded to evaluate and so, for brevity, has been omitted here. Therefore:

$$J_k^i = \frac{\partial h(V_k^i)}{\partial V_k^i} \cdot H_k^i$$

### III. Modelling, Data, and Scenarios

#### A. Implementation

The parent process of the HBM is represented by a set of particles. A particle is made up of bias hypotheses (one hypothesis per bias) and an associated weight. Particle weights are continuously updated following evaluation of a likelihood function (see Equation (1)) – an output of the offspring process. Particle states are updated based on calculation of effective sample size against a preset resampling threshold $N_{thr}$. The multinomial resampling strategy has been selected here. Table I describes the two different starting configurations of the particle filter. In a real-world scenario it is feasible that hypothesis initialisation can closely match the favourable case. This can be accomplished either through precursory GPS readings, inertial measurement unit (IMU) reports, or dead reckoning, amongst other techniques.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>SETUP 1</th>
<th>SETUP 2</th>
<th>Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>±5 m</td>
<td>±200 m</td>
<td>−500 m</td>
</tr>
<tr>
<td>$y, z$</td>
<td>±5 m</td>
<td>±200 m</td>
<td>0 m</td>
</tr>
<tr>
<td>$\tau$</td>
<td>0 s to 1 s</td>
<td>0 to 2 s</td>
<td>0 s</td>
</tr>
<tr>
<td>$\alpha, \beta, \gamma$</td>
<td>0° to 5°</td>
<td>0° to 10°</td>
<td>2°</td>
</tr>
</tbody>
</table>

TABLE I: Particle initialisation for more and less favourable starting configurations: Setups 1 and 2, respectively. Columns 2 and 3 show the range within which initial hypotheses are randomly generated. In the spatial bias case, it is the range either side of the truth (column 4).
\[ f(x, y) = 0.05x^3 + y^3 - x^2y \]

\[ \sigma_\theta^2 = 0.01 \text{ rad} \]

\[ \sigma_\psi^2 = 0.01 \text{ rad} \]

B. Model Definitions

In order to make some inference about a dynamic system, both a process model and a measurement model are required:

\[ \tilde{x}_k = f(\tilde{x}_{k-1}, \tilde{v}_k) \]
\[ \tilde{z}_k = h(\tilde{x}_k, \tilde{w}_k) \]

where \( \tilde{x}_k \) is the 'hidden' state (i.e. the true state - a vector of six dimensions describing object position and velocity in 3D Cartesian space) at time instance \( k \), while \( \tilde{z}_k \) is the observation associated with it (a three dimensional vector for each sensor consisting of observations of object azimuth, elevation, and range). \( \tilde{v}_k \) and \( \tilde{w}_k \) are process and measurement noise, while \( f(\cdot) \) and \( h(\cdot) \) are the process and measurement functions, respectively. \( \tilde{v}_k \) is zero-mean Gaussian process noise with covariance given by:

\[ Q = \text{blkdiag} \left\{ \Sigma_0, \Sigma_0, \Sigma_0 \right\} \]

where [17]:

\[ \Sigma_0 = \begin{bmatrix} 1/\delta t^4 & 1/\delta t^2 \\ 1/\delta t^2 & \delta t/2 \end{bmatrix} \sigma_v^2 \]

\[ \sigma_v^2 \] is known as the process noise intensity level ([18], pg. 273) and its value determines how closely the object follows constant velocity (CV) motion. The choice of value for \( \sigma_v^2 \) is directed by knowing the order of magnitude of the possible change in velocity from one timestep to the next, \( k \to k + 1 \).

This is given approximately by: \( \Delta v \approx \sqrt{Q_{zz}} = \sqrt{\delta t \times \sigma_v^2} \) ([18], pg. 270). The greater the value of \( \sigma_v^2 \), the further the object deviates from CV motion. The measurement noise, \( \tilde{w}_k \), is drawn from a zero-mean Gaussian distribution, \( \mathcal{N}(\tilde{w}_k|0, \Sigma_k) \). Per sensor, \( R_k \) takes the following form:

\[ R_k = \begin{bmatrix} \sigma_v^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_\psi^2 \end{bmatrix} \]

C. Scenarios

This section reviews the artificial scenarios used to test the bias estimation algorithm. Uncertainty quantification of the algorithm is performed for the tracking of a random object trajectory (Section III-C.1) as well as for a bow trajectory (Section III-C.2). However, for all trajectories, sensor biases are identical and remain fixed. For the numerical detail see the final column of Table I in Section III-A. In the interest of reducing computation time, only two sensors are considered here: an uncalibrated sensor and a reference sensor. Note that this algorithm can be applied to the registration of any number of sensors to a chosen reference point. However, bear in mind the limitation in the use of the particle filter which scales up poorly in terms of computation time (see the conclusion for a suggested solution to this constraint). Sensor sampling interval is \( \delta t = 0.1 \text{ s} \) and the simulation is run for a total of \( t = 40 \text{ s} \). Clutter and false alarms are not considered in these analyses, and probability of detection, \( P_d \), is assigned the value 1. The impact of \( P_d \) on the accuracy of the algorithm is investigated in future work. The full set of simulation parameters in the case of the random trajectory scenario is provided in Table II.

(a) Contour plot of bivariate function \( f(x, y) \) for \( a = 0.05 \) and \( b = c = 1 \); \( a \) is chosen to emphasise the central bow-like line, which satisfies \( f(x, y) = 0 \).

(b) Example bow trajectory extracted from \( f(x, y) = 0 \) and enveloped with an exponential function.

Fig. 2: The bivariate function and a bow-like trajectory.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_R )</td>
<td>Range noise SD</td>
<td>10 m</td>
</tr>
<tr>
<td>( \sigma_\theta )</td>
<td>Azimuth noise SD</td>
<td>0.01 rad</td>
</tr>
<tr>
<td>( \sigma_\psi )</td>
<td>Elevation noise SD</td>
<td>0.01 rad</td>
</tr>
<tr>
<td>( \sigma_v^2 )</td>
<td>Process noise intensity</td>
<td>100 m^2 s^{-3}</td>
</tr>
<tr>
<td>( \delta t )</td>
<td>Sampling interval</td>
<td>0.1 s</td>
</tr>
<tr>
<td>( t )</td>
<td>Total simulation time</td>
<td>40 s</td>
</tr>
<tr>
<td>( n_s )</td>
<td>Number of sensors</td>
<td>2</td>
</tr>
<tr>
<td>( P_d )</td>
<td>Probability of detection</td>
<td>1</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>False alarm rate</td>
<td>0.83</td>
</tr>
<tr>
<td>( X )</td>
<td>Number of particles</td>
<td>100</td>
</tr>
<tr>
<td>( N_{thr} )</td>
<td>Resampling threshold</td>
<td>( \frac{4}{3} N )</td>
</tr>
</tbody>
</table>

TABLE II: All simulation parameters. The abbreviation SD stands for standard deviation.

1) Random Trajectories: Data is generated for the tracking of a single object with nearly constant velocity (NCV) motion. Initial states for the object following a random trajectory are drawn from a uniform distribution. Starting location is drawn from within a cubic region of dimension \( 2 \text{ km} \times 2 \text{ km} \times 2 \text{ km} \), while starting velocities are drawn within \( \pm 100 \text{ ms}^{-1} \). The variable \( q \) determines how closely the object follows CV motion (Section III-B): the greater the value, the further the object is from constant velocity motion.

2) Bow Trajectory: The bow trajectory generated for this paper is based on a representative scenario. In the scenario, aircraft, with speed around 222 ms^{-1}, approach a (stationary) point at an altitude of 60 m. They follow a bow-like path close over the coordinates and retreat at an altitude of 9 km. The trajectory generated to closely...
match this case is based on the following bivariate function: \( f(x, y) = ax^4 + by^2 - cx^2y \). This function produces a bow-like shape when \( f(x, y) = 0 \) (see Figure 2a for contour plot of \( f(x, y) \)). The bow-like contour line must be extracted from the function and the data points ordered so that it is temporally coherent. Thereafter, data points are generated such that the speed of the object is as desired (i.e., varying smoothly between 200 ms\(^{-1}\) and 300 ms\(^{-1}\)) for the given simulation time-step. At this point, the trajectory is still only traced in two dimensions. However, adding a third dimension is trivial once one has decided the required three dimensional shape of the trajectory. In this case, an exponential curve was preferred. To achieve this, \( z \) coordinates were produced by setting: \( z = e^{d \times y} \). The value of \( d \) can be varied to alter the rate of increase of the exponential function; here \( d = -0.0006 \) was found to produce the most suitable trajectory (see Figure 2b).

**IV. RESULTS**

The results in this section have been obtained by averaging over a series of MC trials. In the random trajectory case 1000 MC trials were executed for each PF setup, and in the bow trajectory case 50 MC trials were executed (due to time constraints). Box plots and error ellipses are provided comparing bias estimation in setup 1 with setup 2. The difference in these setups is in their hypotheses initialisation – see Table I in Section III-A. This is reflected in the graphs that follow: setup 1 leads to results that are at least 10 times more accurate than setup 2. This result is expected since the performance of the PF is sensitive to initial conditions. Particle updates begin at \( k = 2 \). Likelihood is evaluated, particle weights updated, and effective sample size calculated to assess whether resampling is necessary. When particle weights are updated with the likelihood, the particle with the highest weight is selected and its state (the set of bias hypotheses) extracted and stored. This builds a large array containing only the apparent best hypotheses per timestep. This process runs for the full simulation time. At the end of the simulation, the array containing the best hypotheses is averaged. For example, the final temporal bias estimate is the mean of the most likely temporal bias hypotheses. Since parameters being estimated are constant values, hypotheses appear to oscillate continuously around the truth as opposed to being guided by a motion model. Estimation error is calculated as the difference between the final estimate and the true value. The box plots (see Figures (3) and (4)) show this error; they are labelled with either setup 1 or 2 to indicate how the PF is initialised to generate the results. The notation \( \epsilon_i \) along the \( x \)-axes of the plots simply denote ‘error on bias parameter \( i \)’. Note that, spatial, temporal, and angular biases have units m, s, and \( \circ \), respectively. Finally,
error ellipses are provided: these error ellipses define the region that contain a specified percentage of the data that can be drawn from an underlying Gaussian distribution. For this paper, 95% error ellipses have been generated. Only pitch-yaw is shown as a representative case. For the purpose only of obtaining ballpark figures (and keeping in mind that simulation setups vary from those presented in this paper) a brief comparison to those papers mentioned in Section I-B where numerical values have been quoted is offered: Sigalov et al [7] come within 0.013° of true angular biases in their best case scenario and the work in this paper shows that – in the favourable case – angular estimates come within 0.6°. In the paper by Ge et al [8] mean spatial bias falls within 2 m and in this paper – again, in the favourable case – spatial bias falls within 0.5 m. In the same paper by Ge et al their mean error on angular bias is 0.02°. The temporal bias estimation results can be compared with the work by Bu et al [15] who obtain root mean squared error (RMSE) values for temporal bias estimates that fall within 0.1 s of the truth: that is, with a minimal error of 0.074 s and maximal error of 0.097 s. The less favourable setup in this paper has $\epsilon_T = 0.8525 s$.

V. CONCLUSIONS & FUTURE WORK

This paper performs uncertainty quantification on a joint sensor registration and object tracking algorithm. The algorithm uses a HBM which allows for simultaneous registration and an adaptive architecture. The results demonstrate the importance of PF initialisation in both the slower, simple trajectory case, as well as for a marginally subsonic object describing a more complex trajectory. There are multiple avenues for future work: integrating the bias estimation and object state estimation updates such that they are carried out by a single filter rather than in parallel with two different types of filter; the PF EKF setup can be replaced with the recently proposed adaptive kernel Kalman filter [19] (which also addresses the issue of scalability); and improvements to the simulation for algorithm testing – such as variation of probability of detection and inclusion of false alarms.

REFERENCES