Stone Soup: An Open-Source Framework for Tracking and State Estimation

Paul Thomas*
Jordi Barr
Steve Hiscocks
Defence Science and Technology Laboratory Porton Down
Salisbury, United Kingdom
* pathomas@dstl.gov.uk

Charlie England
Defence Science and Technology Laboratory Portsdown West
Fareham, United Kingdom

Simon Maskell
University of Liverpool
Liverpool, United Kingdom

Bhashyam Balaji
Defence Research and Development Canada
Ottawa, Canada

Jason Williams
Defence Science and Technology Group
Brisbane, Australia

***Abstract*—**The ability to detect and unambiguously follow all moving entities in a state space is important in many domains both in defense (e.g., air surveillance, maritime situational awareness, and ground moving target indication) and the civil sphere (e.g., astronomy, biology, epidemiology, and dispersion modeling). However, tracking and state estimation researchers and practitioners have difficulties recreating state-of-the-art algorithms to benchmark their own work. Furthermore, system developers need to assess which algorithms meet operational requirements objectively and exhaustively rather than driven by intuition or individual preference. We have, therefore, set up a collaborative initiative to create an open-source framework for production, demonstration, and evaluation of tracking and state estimation algorithms. Stone Soup is designed to be a (MIT–licensed) software framework for researchers and practitioners to test, verify, and benchmark a variety of multisensor and multiobject state estimation algorithms. The initiative is supported by four defense laboratories (Defence Research and Development Canada, Defence Science and Technology Laboratory, Defence Science and Technology Group, and Naval Research Laboratory), who are contributing to the development effort for the framework through the Technical Cooperation Program. The tracking and state estimation community will derive significant benefits from this work, including access to repositories of verified and validated tracking and state estimation algorithms, a framework for the evaluation of multiple algorithms, standardization of interfaces, and access to challenging data sets.[1]

"The fundamental requirement of Stone Soup is to enable the comparison of different algorithmic approaches against the same data or simulated scenario. Thus, algorithms must be interchangeable at the state and measurement model level."

INTRODUCTION

Imagine the scene: you are in a conference listening to a paper on a new approach to tracking or state estimation. The work looks impressive, and the author has shown good results on standard data sets. Using the conference Wi-Fi, you download the algorithm and reproduce the tests on your laptop while listening to the paper. Before the end of the session, you have applied the new algorithm to your data with your own metrics and compared it to your own work!

Does this sound far-fetched? As any researcher or practitioner knows, it is currently difficult and time consuming to recreate state-of-the-art algorithms to benchmark one's own work. Comparison of new algorithms with existing solutions involves recoding the alternative algorithms from the academic literature. Industrial users of algorithms also find it difficult to assess which algorithms meet requirements objectively and exhaustively rather than driven by intuition or personal favorites.

However, we see a solution to these issues in the open-source paradigm [1], [2], [3] that encourages coding in the open and exposure of an algorithmic source code. To this end, a consortium of government laboratories has started a project to develop an open-source tracking and state estimation framework suitable for populations with open-source (MIT-licensed[2]) com-

[1] Content includes material subject to © Crown copyright (2018), Dstl. This material is licensed under the terms of the Open Government Licence except where otherwise stated. To view this license, visit http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3 or write to the Information Policy Team, The National Archives, Kew, London TW9 4DU, or email: psi@nationalarchives.gsi.gov.uk.

[2] https://opensource.org/licenses/MIT.

ponents. The project described in this article aims to provide a flexible and unified software platform for researchers and engineers to test, verify, and benchmark a variety of existing multisensor and multiobject estimation algorithms. It will also allow rapid prototyping of new algorithms in three language sets (Python, MATLAB, and C/C++), by providing a set of libraries that implement the necessary functions for tracking and state estimation. Note that the project is not intending to mandate a uniform "straightjacket"; instead, we would like to develop a modular repository of algorithms along with some convenient and popular frameworks for using those components.

STONE SOUP

The project, titled "Stone Soup," has six component types [4]: framework, data, algorithms, metrics, simulators, and sensor models (see Figure 1). The framework is the core of the project. This is the software infrastructure necessary to construct appropriate combinations of components from a repository. The framework is designed to mirror the mathematics in the components and be extensible to incorporate new algorithms developed by the research community, as well as data sets from different fields of interest, thus providing the ability to evaluate new algorithms on different problems more efficiently.

Evaluation takes place via the (interchangeable) metric module. Algorithms can also be evaluated against simulations that are generated by the interchangeable simulator module. The framework will also include an interface to the detector, which is the component that transforms raw data into discretized detections. These are likely to exist as external components, for example, open-source computer vision libraries for target detection from imagery, with a Stone Soup–compatible interface.

Components are logically connected by the data flow diagram in Figure 2. This illustrates how algo-

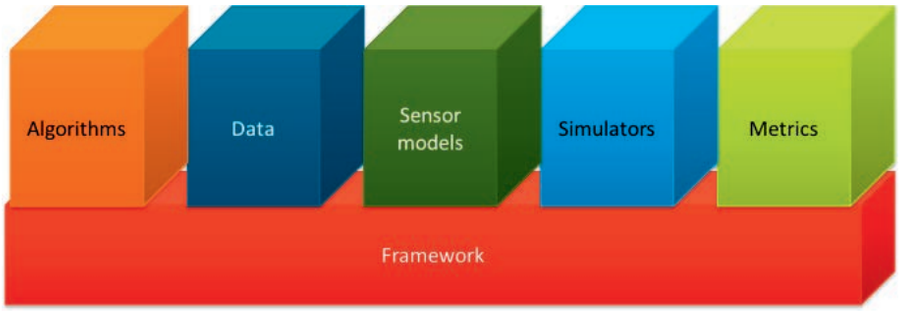

Figure 1
The components of Stone Soup.

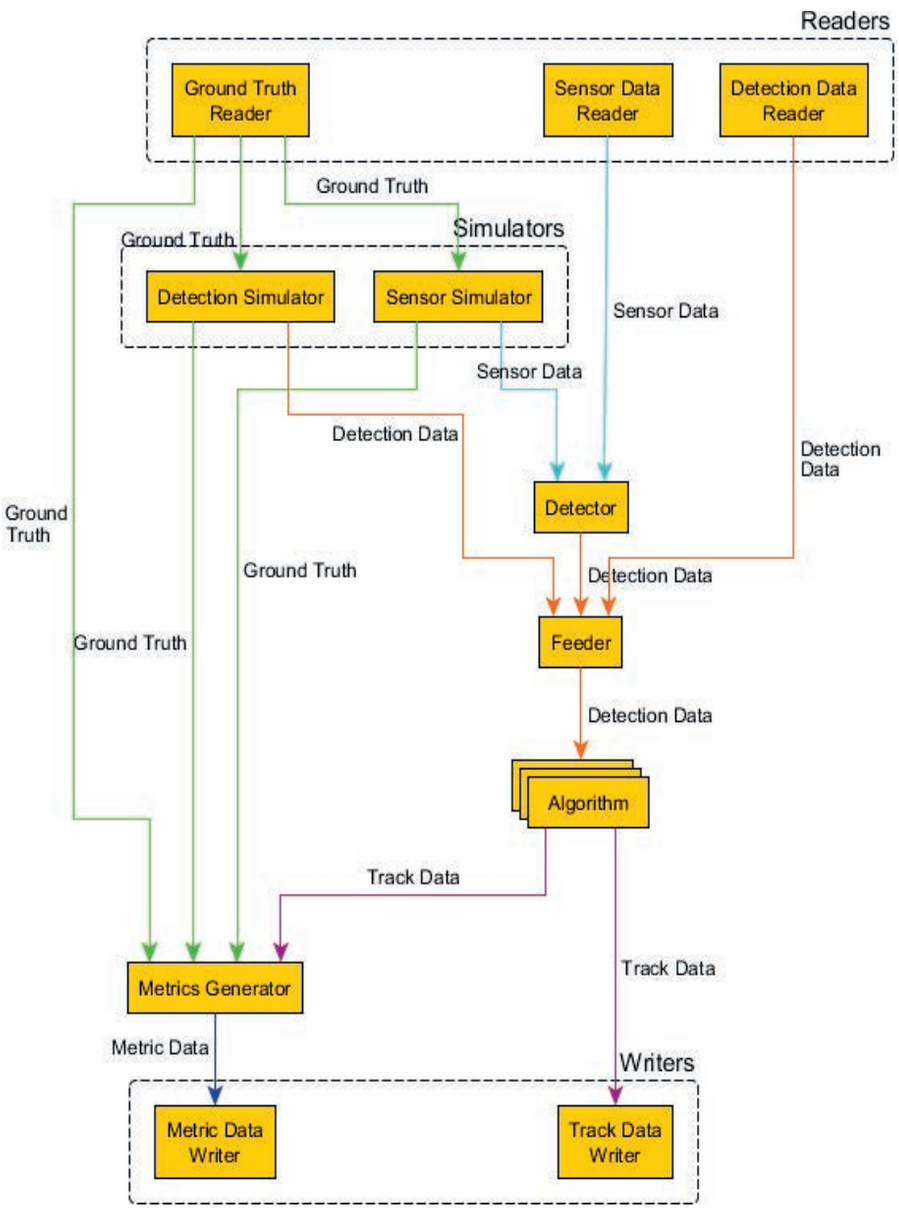

Figure 2
One possible example of a logical data flow that can be configured in Stone Soup.

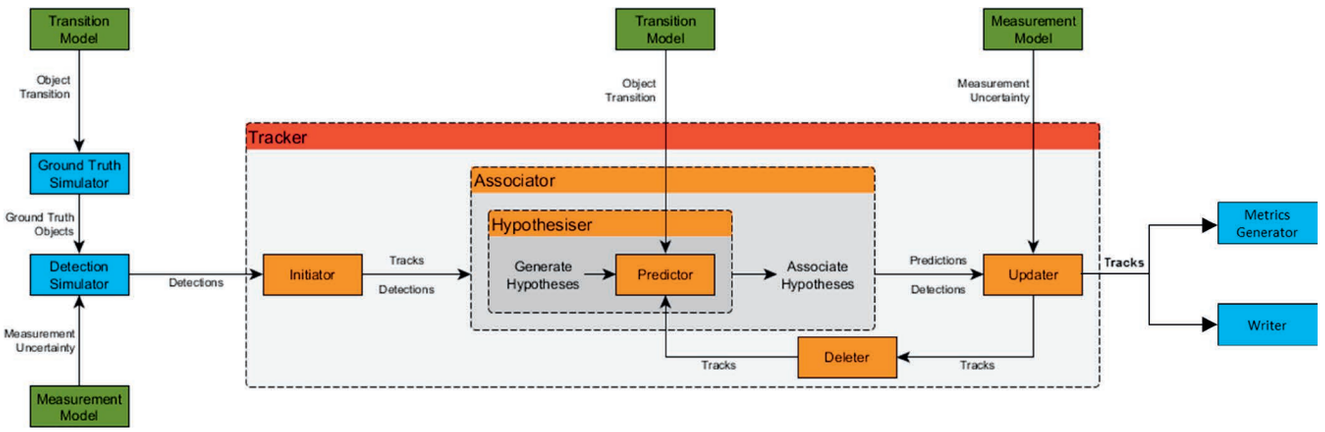

Figure 3
Simple tracker implementation.

rithms are fed with detection data and output track data. However, algorithms are not treated as monolithic code blocks in Stone Soup. Subcomponents of the algorithm can be interchanged for other functionally equivalent approaches. A simple tracker is shown in Figure 3. This makes a very powerful engine that can be used to understand exactly what part of a proposed algorithm is driving its performance. This is achieved by constructing a standard interface between subcomponents, using inheritance where necessary, such that the calling routine is agnostic to the approach within the code block. The framework is coded in Python.[3] Python's class-based, object-oriented, duck-typed[4] paradigm elegantly enables this approach. The interface that will allow MATLAB and C/C++ algorithms to be called from the Python framework is currently being developed and will be published in advance of the beta release.

The Stone Soup development team is working toward a beta release[5] of the software in April 2019 on a public repository of GitHub. A recent code sprint at the University of Liverpool is shown in Figure 4. The beta release will contain a set of basic Kalman filter– and particle filter–based algorithms, but future aspirations include things such as scalable multitarget track before detect, parameter estimation, support for parallel particle filtering, and support for real-time interfacing. We hope that these and many other research topics will be advanced by the open community with this framework. Indeed, we expect that contributions to future International Society for Information Fusion (ISIF) fusion conferences and the *Journal of Advances in Information Fusion* would, by default, produce algorithms in a Stone Soup–compliant format for exposure and use standard Stone Soup data sets and scenarios for comparisons. Although this represents the desired steady state, for accelerated development, the team hopes to integrate selected components drawn from the Tracker Component Library [5], published on GitHub. Similarly, Stone Soup would be released with an initial collection of data from the contributing government laboratories, with

[3] See www.python.org.
[4] https://en.wikipedia.org/wiki/Duck_typing.
[5] See https://en.wikipedia.org/wiki/Software_release_life_cycle.

Figure 4
A recent Stone Soup code sprint at the University of Liverpool.

the desired aim that all parties who collect relevant data would contribute in the future.

ACCESSIBILITY

The key to wide adoption of this framework is to make it as accessible as possible. Most of the code developed up to the beta release of the Stone Soup project will be released under the MIT license, which is one of the most permissive forms of an open-source license. After beta release, others will be able to contribute additional components under any "permissive license"[6] (with the MIT license preferred), and existing components may be modified under their original license.

DEVELOPMENT PRIORITIES

Open-source software, unlike the traditional software development model, tends to develop amorphously according to the

[6] https://en.wikipedia.org/wiki/Permissive_software_licence.

will and enthusiasm of members of the user/developer community. However, to rapidly build the initial framework in the consortium phase, it is useful to have a set of development priorities. These priorities relate to compatibility with specified classes of algorithm and types of data and models that might be added in the open phase, rather than requiring specific functions to be added immediately.

DATA

Priority for the initial phases of Stone Soup will be to enable the use of the following data types. This involves the creation of an appropriate data model in the framework for handling the data and making available representative data sets in the repository. The initial data types will be:

- Airborne radar detection data, which can be two dimensional or three dimensional, from a rotating radar or from a planar array;
- Coincident automatic identification system data;
- Astrodynamic data (electro-optical [EO]): satellites against a star background; and
- Detections of ground targets in airborne EO and infrared data.

The intent is to build wider compatibility with further data types and provide more example data sets as the project progresses.

ALGORITHMS

Priority for the initial phases of Stone Soup will be to ensure compatibility with the following generic algorithm types:

- Filtering algorithms: discrete-time state and measurement models
  - Standard Kalman filter for the linear state and measurement model
  - Extended Kalman filter
  - Derivative-free Kalman filters
- Particle filter class of algorithms, e.g.,
  - Sequential Monte Carlo
  - Variations in sampling strategy (sampling importance resampling, sequential importance sampling, and auxiliary particle filter)
  - Rao–Blackwellized particle filter
- Multiple model filtering algorithms for Kalman filter class of algorithms

Sample outputs from two different algorithms are shown in Figure 5.

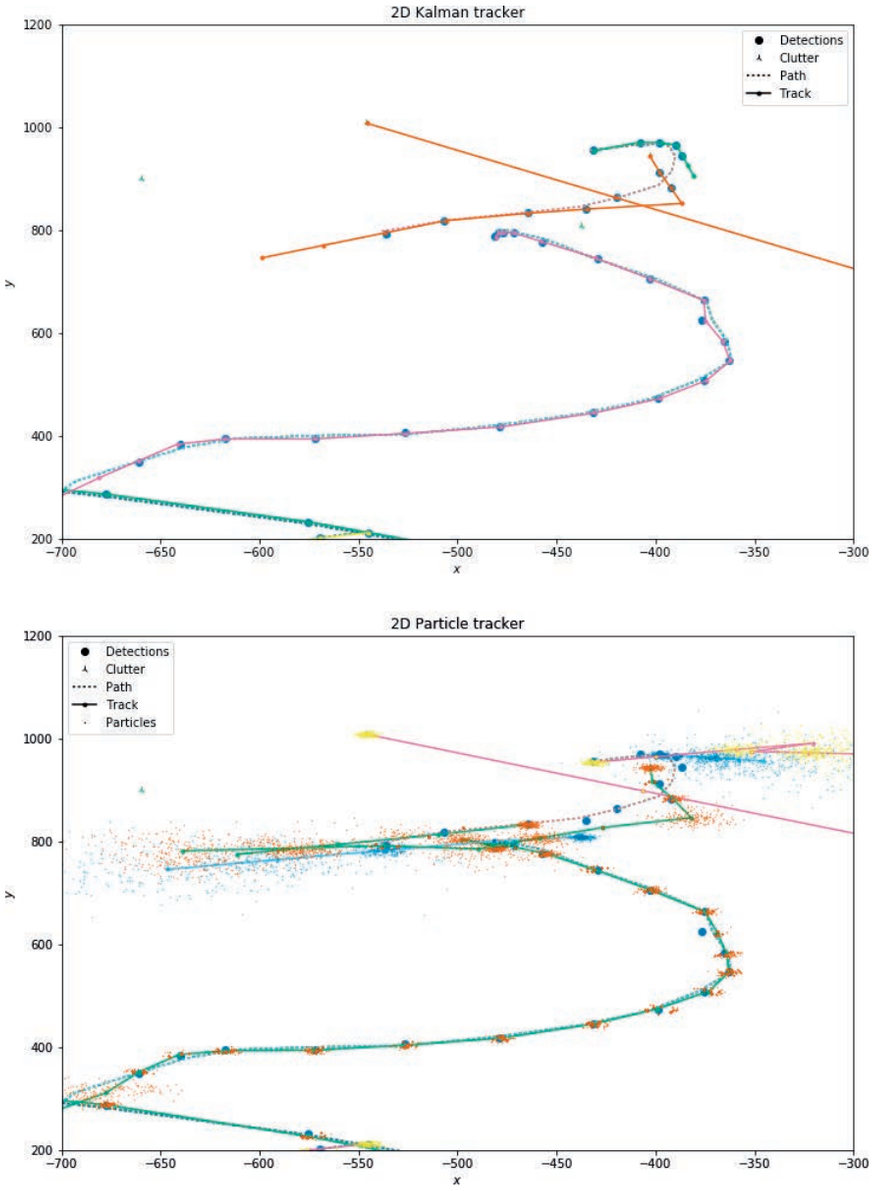

Figure 5

Output from the alpha version of Stone Soup, showing swap of Kalman filter (KF; above) with particle filter (PF; below) component against the same data with all other components the same. Note the differences in behavior during the target's execution of the tight turn at the top right and also the slow turn at the center right. Note: These plots were produced with simplistic implementations of both filters to test algorithm exchange and the interfaces only. No general conclusions of KF or PF performance should be taken from these plots.

THE STONE SOUP STORY

Stone soup is an old European folklore story, with many versions told in several countries. Here is the authors' version, capturing the essence of all the versions but not corresponding exactly with any particular one.

A traveler arrives in a village seeking food and shelter. Upon arrival, he discovers all the inns and shops are closed, and there is no food at the market. The villagers tell him there is no food in this village, and he should move on. Undeterred, the traveler produces a large pot from his backpack and proceeds to fill it with water from the stream. He then drops several large stones in it and places the pot over a fire in the market square. The villagers become curious and gather round, asking what he is doing. The traveler announces that stone soup is being made and that it is very nutritious, although it needs some time to develop its full potential.

By this time, most of the village has gathered round, thinking our traveler to be mad. He is playing the crowd, "Nothing better than stone soup," he says, "except perhaps stone soup with herbs." One of the villagers does not mind parting with a few stalks of thyme from the garden. "Could do with a little bit of body," says the traveler, and another villager finds some old potatoes from the larder. "Still needs a little bit of garnish to improve the flavor," he says, and an old cabbage and some salt beef is found by the villagers. Eventually, more and more villagers make contributions and a delicious soup is produced. Finally, the traveler removes stones from the pot, and the nourishing soup is shared around the village.

Moral: By working together, with everyone contributing what they can, a greater good is achieved.

Of course, in the case of the original story, the soup is consumed. However, in the case of a software project, the output from the collaboration will remain to benefit everyone into the future.

INTERFACE STRUCTURE

The fundamental requirement of Stone Soup is to enable the comparison of different algorithmic approaches against the same data or simulated scenario. Thus, algorithms must be interchangeable at the state and measurement model level.

Therefore, the key to the success of the framework is the creation of an underlying interface structure that enables algorithm interchangeability across all supported algorithm classes and can cope with all supported data types. This is more complex than it seems because, for example, the algorithms represent uncertainty in different ways, e.g., algorithms based around the Kalman filter model a target state distribution as a mixture of one or more Gaussian distributions, whereas particle filters represent the distribution by a set of weighted samples.

Furthermore, the framework must support application of a measurement model to data from a variety of sensors. In effect, this enforces "data interchangeability." Examples of sensor models will include range and bearing, bearings only, range only, categorical (identity) data, or a combination of them, among many others.

"The Stone Soup development team is working toward a beta release of the software in April 2019 on a public repository of GitHub."

METRICS

Assessment of candidate tracking and state estimation algorithms requires a means of objective evaluation by using a set of metrics corresponding to characteristics of interest to operational systems [6]. In certain circumstances, knowledge of the actual number of entities and their states may be available, such as in simulations. In other circumstances, e.g., when real data has been collected, full knowledge of the number of entities and their states may not be available.

Priority for the initial phases of Stone Soup will be to enable assessments in the presence of entity "truth" information. In this case, an absolute evaluation of each alternative algorithm can be conducted by comparing the tracking and state estimation output against the truth data. Typically, a data association approach is first used to associate track data with entity truth data [6]. Subsequently, performance metrics can be applied to quantify tracking characteristics. As with other aspects of Stone Soup, the intention is to develop further performance metrics as the project progresses.

JOINING IN

The Stone Soup project will be the core of a new Working Group of ISIF,[7] namely, the Open Source Tracking and Estimation Working Group (OSTEWG). This was approved at the ISIF Board of Directors meeting that coincided with FUSION 2018. OSTEWG will enable a wider outreach with the firm aspiration of building Stone Soup into the heart of the fusion community, centered around meetings at each annual FUSION conference. OSTEWG's charter is currently in development, and the first meeting will be aligned with FUSION 2019 in Ottawa, Canada. This will be timely, as it will occur shortly after Stone Soup's beta release.

The intention is for OSTEWG to set challenges supported by data within the Stone Soup framework each year at the fusion conferences. This is a popular model (exemplified by

[7] http://isif.org/working-groups/isif-working-groups.

computer vision competitions at CVPR,[8] data science competitions hosted through Kaggle[9]) that encourages participation across the community. The intention is that data would be publicly accessible (although each data set will likely have caveats), and challenge entrants would provide their solution in a Stone Soup–compatible format for evaluation at the following FUSION conference. Innovative approaches could also become papers at a special session each year.

We predict that the tracking and state estimation community will derive significant benefits from this work, including access to a repository of tracking and state estimation algorithms, framework for evaluation of multiple algorithms, and access to challenging data sets. Membership of the Working Group of ISIF is growing, as this initiative becomes known, and much interest was generated at the recent FUSION 2018 conference. Membership is still open, and any interested party should contact the corresponding author.

ACKNOWLEDGMENTS

Software engineering support and consultancy is gratefully acknowledged from Richard Green, Sam Thomas, Alex Mulliner, and Matt Cook, Defence Science and Technology Laboratory, Porton Down, Salisbury, United Kingdom; David Crouse, Naval Research Laboratory, USA; Lyudmil Vladimirov, Yifan Zhou, James Wright, Joshua Coates, Paul Horridge, Robert Moore, and Angel Garcia-Fernandez, University of Liverpool, United Kingdom; Mike McDonald, Emilie Altman, Peter Carniglia, Tanya Gatsak, and Sang Bin Lee, Defence Research and Development Canada; Kruger White, Defence Science and Technology Group, Australia; Thia Kirubarajan, McMasters University, Canada; Felix Govaers, Fraunhofer FKIE, Germany; and Miodrag Bolic, University of Ottawa, Canada.

[8] http://cvpr2018.thecvf.com/.
[9] https://www.kaggle.com/.

REFERENCES

1. Raymond, E. S. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, CA: O'Reilly Media, 1999.
2. Government Digital Service. Coding in the open. Available: https://gds.blog.gov.uk/2012/10/12/coding-in-the-open/, last access Nov. 2018.
3. Civic Commons Wiki. Open source development guidelines. Available: http://wiki.civiccommons.org/Open_Source_Development_Guidelines/, last access Nov. 2018.
4. Thomas, P. A., Barr, J., Balaji, B., and White, K. An open source framework for tracking and state estimation ("Stone Soup"). In *Proceedings of the SPIE Signal Processing, Sensor/Information Fusion, and Target Recognition XXVI*, Anaheim, CA, Apr. 2017.
5. Crouse, D. F. The tracker component library: free routines for rapid prototyping. *IEEE Aerospace and Electronic Systems Magazine*, Vol. 32, 5 (2017), 18–27.
6. Blackman, S., and Popoli, R. *Design and Analysis of Modern Tracking Systems*. Boston, MA: Artech House, 1999.