**Stone Soup Run Manager**

**User Guide**

|  |  |  |
| --- | --- | --- |
| Revision History | |  |
| V1.0 | 3 June 2019 | Initial Version |
|  |  |  |
|  |  |  |

1. **Run Manager Overview**

The purpose of the Run Manager in Stone Soup is to enable users to run multiple experiments, collect the results, compute metrics over these results, and compare the metrics of these experiments head-to-head. These experiments could be run across entirely different trackers, they could be run across multiple variants of the same tracker (with variations of different parameters or swapping out different sub-components), or they could be Monte Carlo runs on the same tracker. These experiments can be run using simulated data, real-world sensor data treated as detections, or real-world data treated as groundtruth with simulated sensor detections of the groundtruth. The Run Manager also provides multiple options for saving simulated data and experiment configurations to that they can be shared with colleagues or saved for future reference.

1. **Basic Run Manager Experiment**

This code can be found in the Jupyter Notebook Run\_Manager\_Demo\_01\_Basic.ipynb.

In order to create a Run Manager experiment, first create a Stone Soup tracker in the normal way.



 Figure 1. Creating a Stone Soup experiment.

Next, specify the metrics that you would like to calculate on the results of the experiment. Several of the metrics will depend on a Metrics Associator, so you will need to include one of these.

 Figure 2. Defining a Metrics package.

Encode the tracker and the metrics in the Run Manager experiment configuration format. Here, we demonstrate how to encode multiple Trackers for the same experiment using the same tracker; typically you would build a second tracker to include in the experiment.

 Figure 3. Assembling a Run Manager experiment.

Finally, create a Run Manager object, load the experiment configuration into it, run the experiment, and save the results.

 Figure 4. Creating a Run Manager object and running the experiment.

The Run Manager will export the results of the experiment to a file called YYYY-MM-DD\_hh-mm\_experiment\_results.txt in the run\_manager\_results folder. The experiment results file will include information defining the Tracker configuration, the metrics specified, and the results of the metrics calculations.

 Figure 5. Run Manager experiment output results.

1. **Parallel Experiments – Varying Parameters and Components**

The code in this section can be found in the Jupyter Notebook Run\_Manager\_Demo\_02\_Vary\_Parameters.ipynb.

Sometimes it may be desirable to run a series of experiments on the same Tracker except with one or more parameters varied, or using different sub-components, in order to observe the effect these parameters/components have on the performance of the Tracker. This section will demonstrate how to automatically create multiple experiments with these variations.

Building off of the code in Section 2, we will use multiple values/components for several of the parameters and sub-components of the Tracker. We will operate on detection\_sim.detection\_probability, detection\_sim.clutter\_rate, and tracker.deleter. The relevant changes to the code are shown below.

 Figure 6. Creating multiple experiments by varying parameters.

In this example, detection\_sim.detection\_probability now has three possible values (0.8, 0.9, and 0.95); these are encoded in an Iterator to indicate that we are varying the parameter. detection\_sim.clutter\_rate is defined to have one value (3), but two alternate values are defined in components (4 and 5). Note that when only one value is defined in either of these locations, an Iterator is not necessary, but when multiple values are defined, an Iterator must be used. tracker.deleter is defined as None in the Tracker definition, but we define two different deleters and then specify their use in components. Note that if a parameter or sub-component is defined as None, then valid options *must* be defined in components.

Given this combination of parameters, the Run Manager will create 18 different experiments and record their results accordingly.

1. **Sharing Components Between Experiments**

The code in this section can be found in the Jupyter Notebook Run\_Manager\_Demo\_03\_Shared\_Components.ipynb.

Sometimes it may be desirable to run two different Trackers using a common sub-component. One approach is to simply define the sub-component and define both Trackers using that sub-component. Another two approaches are shown below, where the sub-component is defined in components and then both Trackers are directed to use this component.

 Figure 7. Sharing a component between different Trackers.

The first definition of components would be the typical way to use a common sub-component in the common case where the sub-component is a memory-less object. However, some components contain memory that we wish to carry between different Trackers, in which case the second definition of components would be used; this approach will be discussed later as a means to share simulated data across different experiments.

It should be noted that when using either of these approaches to share a common component, all sub-components and parameters of the shared component will also be shared. In this example, when utilizing a common instance of universal\_deleter, it would not be possible to have different values of covar\_trace\_thresh for the different Trackers.

1. **Monte Carlo Simulations**

The code in this section can be found in the Jupyter Notebook Run\_Manager\_Demo\_04\_Monte\_Carlo.ipynb.

Many users will want to use Monte Carlo simulations to run multiple experiments (using different data) against the same tracker in order to evaluate its performance in diverse situations. In order to avoid combinatorial explosion by combining Monte Carlo experiments with the Tracker variants discussed previously, the Stone Soup Run Manager only allows Monte Carlo simulations for a single variant of a single Tracker (attempts to do otherwise will be flagged by a pre-experiment check). However, should the user wish to run Monte Carlo experiments using the exact same data against different Trackers, there is an option to save the data and run it against a different Tracker.

In the Stone Soup Run Manager, a Monte Carlo experiment *must* use simulated data; multiple experiment runs performed by importing saved data is considered a regular experiment. This is discussed in Section 6.

 Figure 8. Defining a Monte Carlo experiment.

The condition monte\_carlo specifies how many runs of the experiment should be performed. The optional condition output\_data indicates that the data generated should be saved to output files; this is useful if the user wishes to run a Monte Carlo experiment against a different Tracker using the exact same data. Data will be saved in files entitled YYYY-MM-DD\_hh-mm\_ExperimentXXX\_data.txt in the run\_manager\_results folder, where XXX indicates the experiment number.

1. **Data Options**

Within Stone Soup, there are two levels of sensor data; groundtruth data indicates the true positions of ships, aircraft satellites, etc. and sensor data that represent a sensor’s detection of this groundtruth data. Sensor data has associated uncertainty and clutter measurement, and detections can be missed.

Users of Stone Soup can run experiments where real-world sensor data is ingested from external sources and treated as sensor detections. This approach is appropriate when the real-world data comes from sensors that have uncertainty and clutter as part of the dataset. Stone Soup users are free to utilize existing data readers or adapt custom readers tailored to their input dataset. Since groundtruth is rarely available for these types of real-world sensors, the presence of groundtruth is abstracted away. Metrics that rely on groundtruth cannot be calculated in this situation. This approach is demonstrated in the Jupyter Notebook Run\_Manager\_Demo\_05\_Import\_Data.ipynb, with the relevant lines of code shown in Figure 9.

 Figure 9. Importing Detection data from a file.

It is also possible to ingest goundtruth data from an external file and then apply detection models to this data to simulated sensor detections. This capability does not currently exist, but a StoneSoup user could easily create it by modifying a detection reader (as used in Figure 9) so that ingested data is represented as GroundTruthState objects rather than Detection objects.

Both the groundtruth and sensor detections can be simulated; this is the approach taken by most of the examples in this document.

As noted previously, a user may only run a Monte Carlo simulation on a single variant of a single tracker. However, there are times when a user may wish to use the exact same data for Monte Carlo runs against multiple trackers. In this case, a user can specify that the data from a Monte Carlo experiment should be exported to a file (Figure 8). Subsequent trackers can be created by defining the Detections input to the Tracker as Readers pointing towards the output data files. Code demonstrating this approach can be found in the Jupyter Notebook Run\_Manager\_Demo\_06\_Monte\_Carlo\_Repeat.ipynb.

 Figure 10. Experiment using data from a Monte Carlo experiment on another Tracker.

A slightly more complicated situation is where a user wishes to run a single simulated dataset (not a Monte Carlo experiment) against multiple Trackers. One approach is to export the simulated data from the first Tracker and run an experiment using this saved data for the second Tracker (Figure 10). However, a more sophisticated approach that accomplishes this in a single experiment is to define a data simulator as a component and then reference this component for both of the Trackers (Figure 11, code can be found in the Jupyter Notebook Run\_Manager\_Demo\_07\_Same\_Data\_Multiple\_Trackers.ipynb). The Run Manager code interprets this by saving the data that is generated the first time the Detector is called, and then using this data for subsequent calls to the Detector. The saved data is output in the format YYYY-MM-DD\_hh-mm\_shared\_data.txt in the run\_manager\_results folder.

 Figure 11. Using the same simulated dataset for two Trackers.

1. **Running Saved Experiments**

The code in this section can be found in the Jupyter Notebook Run\_Manager\_Demo\_08\_Run\_Experiment\_From\_Saved\_Config.ipynb.

We expect that many Stone Soup users will be part of a larger collaborative community. Therefore, users may wish to save experiment configurations and share them with colleagues. The Stone Soup Run Manager allows users to easily export experiment configurations and later import them back into the Run Manager.

 Figure 12. Running an experiment from a saved configuration.

1. **Displaying and Analysing Experiment Results**

This code can be found in the Jupyter Notebook Run\_Manager\_Demo\_01\_Basic.ipynb.

Once users have used the Run Manager to run a batch of experiments, we expect that they will want to view, analyse, and process the results. The following code demonstrates a few approaches.

First, we import the experiment results from the output text file (Figure 13).

 Figure 13. Importing experiment results from output text file.

Then we can graph the tracks, groundtruth, and detections to see how well they align with each other (Figure 14, Figure 15).

 Figure 14. Graphing tracks, detections, and groundtruth.

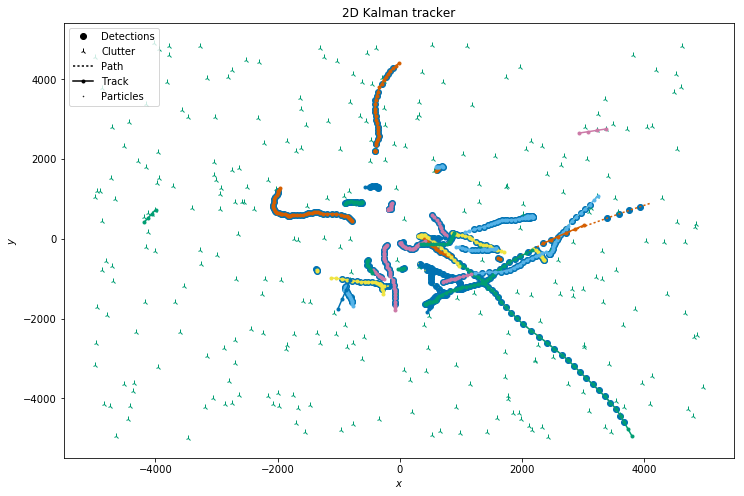


Figure 15. Graph of tracks, detections, and groundtruth.

Most of the calculated metrics are a single value for the entire experiment (Figure 16, Figure 17).

 Figure 16. Printing single-value metrics.

 Figure 17. Single-value metrics from experiment.

The OSPA metric is a list of OSPA distances at each time step in the experiment (Figure 18, Figure 19).

**Figure 18. Displaying OSPA distance metric.**

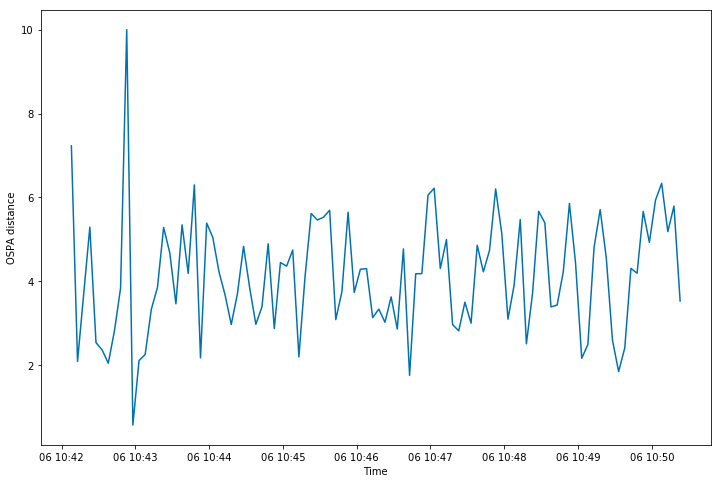


Figure . OSPA distance metric from experiment.