

Target Tracking: exercise 3

Gustaf Hendeby
`gustaf.hendeby@liu.se`

Version: 2025-11-12

The purpose of this exercise is to implement a basic *multi hypothesis tracker* (MHT). The intention is to gain fundamental understanding of what it takes to implement an MHT, not to write a MHT ready for use in production. The developed tracker will be compared to the *single hypothesis trackers* (SHT) studied in the previous exercise. To make it easy to compare the results, the same synthetic and mysterious data is used to evaluate the algorithm.

Note, the requested functionality is available, *e.g.*, in MATLAB's Sensor Fusion and Tracking toolbox, but it is not allowed to use these.

1 Infrastructure

Use the simulated trajectories created in Exercise 2, see Figure 1, when developing your algorithm.

2 Multiple Hypothesis Multi-Target Tracker

In this part of the exercise, you will extend and modify the *global nearest neighbor* (GNN) SHT you created in Exercise 2, into a basic hypothesis oriented MHT. Developing a full fledged MHT with all bells and whistles needed to deal with real problems is outside of the scope of this exercise. (If you need an off the shelf MHT, the one provided by the Sensor Fusion and Tracking MATLAB Toolbox is probably a good candidate.) Instead, the focus is on understanding the fundamental principles of the tracker; *i.e.*, you are not expected to deal with clustering and other tricks needed to make your code very efficient.

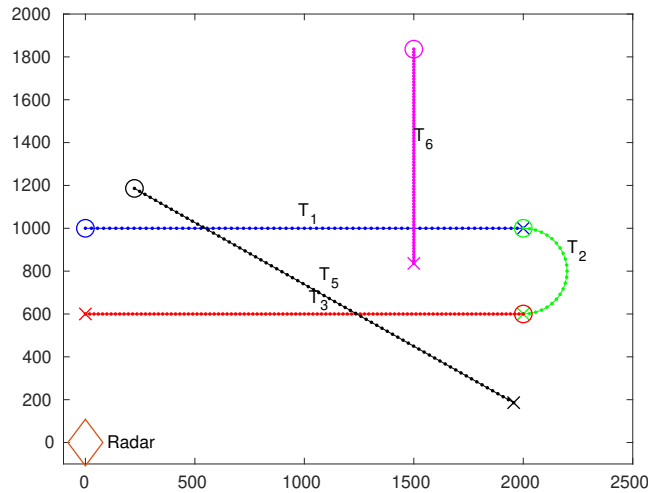


Figure 1: Illustration of the specified trajectories \mathcal{T}_1 – \mathcal{T}_6 .

2.1 Tracker Structure

The suggested structure of your MHT is very similar to that of the SHTs developed in the last exercise, with obvious extensions to handle keeping multiple hypotheses going at the same time.

The code revolves around manipulating different hypotheses, hence combine all information needed to describe a hypothesis in one structure. Furthermore, the MHT implicitly deals with the track management so there is no need for a separate container of confirmed and tentative tracks. That is, let the hypothesis comprise a list of tracks (it might be a good idea to include the full trajectory in the tracks) and the probability associated with the hypothesis. A more complete implementation of a HO-MHT would also have to store the (recent) history of the hypothesis to allow for, *e.g.*, N -scan pruning and more fancy presentation of the results, but for sake of simplicity we leave this out.

As a new scan arrives with new observations, do the following for each current hypotheses:

1. For all tracks in the hypothesis: Predict the tracks to the time of the scan.
2. For all tracks (confirmed and tentative) in the hypothesis:
 - (a) Gate with the measurements in the scan, to obtain the validation matrix.
 - (b) Create the association matrix.
 - (c) Use Murty's algorithm to generate the N_h best new hypothesis.
 - (d) For each association hypothesis:
 - i. Update the tracks; both track logic and the filters using the obtained associations. (Do not forget to update the tracks that end up without measurements.)
 - ii. Remove targets that are considered dead after updating the track logic.

- (e) Remove the measurements from the scan that has been gated with any confirmed target, even if the measurement was in the end not used in the association hypothesis or the track it was associated with died.
3. For all remaining measurements (not gated with any track):
 - (a) Randomly pick one of the remaining measurements, and create a new tentative track, initialize both filter and track logic. Gate with the remaining measurements, and remove any measurements gated with the new track.
 - (b) Repeat (a) until no more measurements are left.
4. Compute the probability of all newly created hypotheses.

Performing the above procedure for all current hypotheses will produce up to N_h^2 new hypotheses, N_h hypotheses per original hypotheses.

In the next step, reduce the number of hypotheses. There are several way to do this. In this exercise, simply keep the N_h most probable hypotheses will be enough. Feel free to experiment with more advanced pruning or merging methods to see if that yields a better end result.

Task: Implement a basic hypothesis oriented MHT tracker, as outlined above. An implementation of Murty's algorithm has been provided for your convenience.

As in the previous exercise. Introduce new difficulties in steps to be able to more easily debug your code. Start with noise-free measurements, no missed detections, and no clutter. Then add the difficulties one at the time once you are content your code works as expected, this way it is easier to predict the expected behavior and find where the algorithm breaks down.

2.2 Evaluation

Task: Apply the MHT to the simulated measurements from trajectories \mathcal{T}_1 , \mathcal{T}_3 , \mathcal{T}_5 , and \mathcal{T}_6 . Evaluate the tracking performance, by considering:

- Evaluate the performance both in position plots and in time. Use different colors (or symbols) for different track labels. Compare with the true known trajectory.
- Elaborate on tracking accuracy in terms of track loss etc, if applicable.

Make sure to indicate tracks and parts of tracks that are tentative.

One way to obtain an output from the MHT is to consider the most likely hypothesis in each time instance for comparison. Feel free to experiment with other options.

Compare the new results to those you derived in Exercise 2.

2.3 Mysterious Data

In the last part of the multi target tracking exercise, you should estimate the trajectories of the targets given by the measurements from a range-bearing radar in the file `ex2data.mat` available as `Y`. The data is formatted the same

way as the output of the simulator described above. The data is realistic, but is constrained to 2D, containing clutter and lost detections. The sample time is approximately $T = 0.26$ s (between measurement scans), and the radar sensor was placed in the origin.

Task: Evaluate and compare the MHT to the GNN and JPDA on the mysterious dataset.