Target Tracking: exercise 2

Gustaf Hendeby
gustaf.hendeby@liu.se

Rickard Karlsson
rickard.g.karlsson@liu.se

Version: 2021-08-03

The purpose of this exercise is to introduce *multi-target tracking* (MTT), and to get acquainted with common algorithms for *single hypothesis trackers* (SHT). To do this, first, the simple simulation environment will be extended to generate measurements from a multiple target scenario. After that, a framework for multiple target tracking will be developed. In which in this exercise it will make up the basis for a *global nearest neighbor* (GNN) tracker and then a *joint probabilistic detection association* (JPDA) tracker. In the next exercise, the GNN solution will be extended to a basic *multi-hypothesis tracker* (MHT). The filters and track logic developed in the previous exercise will be used in the process. Finally the developed GNN and JPDA algorithms should be evaluated using a provided realistic dataset.

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

Task: Extend the software developed in Exercise 1 to be able to simulate measurements from several targets at a measurement scan.

In MATLAB, we suggest the same function as before:

```
function Y = generatedata(X, sensor, clutter)
```

where `X` is a cell array of state trajectories (one time per column), `sensor` is a struct comprising `h` the measurement function, `R` the measurement covariance (assume white Gaussian noise), `PD` the probability of detection; and `clutter` a struct comprising `volume` that span the tracking volume, and `beta` clutter rate (assume the number of clutter to be Poisson distributed, and the clutter uniformly distributed in the volume). The output is a cell array for each time instance, with zero or more measurements.

Task: Extend the set of trajectories, to the following set of trajectories:

- $\mathcal{T}_1$ Target travels 2000 m due east, at 100 km/h, starting in $(0, 1000)$ m.
- $\mathcal{T}_2$ Target traveling due east makes a 180° clock-wise turn following a circular path with radius 200 m, at 50 km/h, ending up heading west, starting in $(2000, 1000)$ m.
- $\mathcal{T}_3$ Target travels 2000 m due west, at 70 km/h, starting in $(2000, 600)$ m.

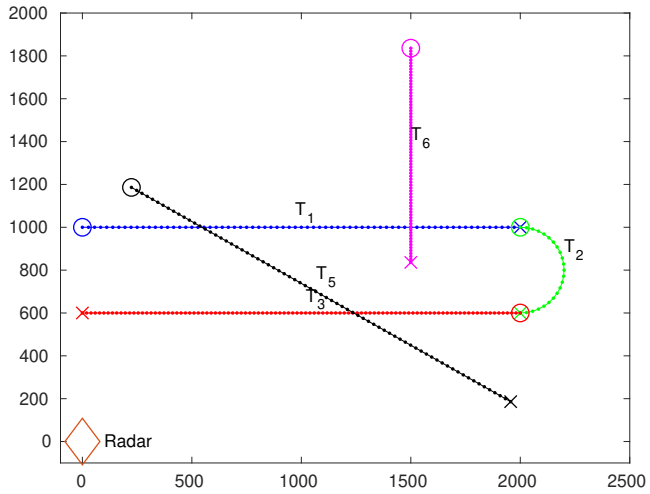

Figure 1: Illustration of the specified trajectories $\mathcal{T}_1$–$\mathcal{T}_6$.

$\mathcal{T}_4$ Connect the three segments ($\mathcal{T}_1$–$\mathcal{T}_2$–$\mathcal{T}_3$) above to get a trajectory of a maneuvering target.

$\mathcal{T}_5$ Target travels 2000 m with 30° clock-wise angle to the x-axis, at 100 km/h, starting in $(200, 1200)$ m.

$\mathcal{T}_6$ Target travels 1000 m due south, 50 km/h, starting in $(1500, 1800)$ m.

See Figure 1 for an illustration.

These trajectories will be used in the rest of this exercise. Unless otherwise stated, assume a range-bearing radar ($\sigma_r = 10\,\text{m}$ and $\sigma_\phi = 0.001$) with probability of detection $P_\text{D} = 0.9$ to generate the measurements. Furthermore, assume Poisson clutter with intensity $\beta_\text{FA} V = 2$, uniformly distributed in the tracking volume. The sample time can be set to $T = 1$.

For this exercise, simulate the trajectories $\mathcal{T}_1$, $\mathcal{T}_3$, $\mathcal{T}_5$, and $\mathcal{T}_6$ at the same time. That is, your data should have four simultaneous targets present at most of the time.

2 Single Hypothesis Multi-Target Tracker

In this part of the exercise you will design two single hypothesis multi-target trackers, for different types of targets, and add functionality to handle both missing data and clutter. It will rely on code developed in the previous exercise: simulation, track logic, etc. The focus is the principles, hence getting the most efficient implementation possible is not very important and things such as clustering and other tricks will be disregarded.

2.1 Tracker Structure

The same basic structure can be utilized when designing most single hypothesis trackers. It is strongly recommended to set up the following template code, and

then use it for both the GNN and JPDA part of this exercise. (The structure will also help you in coming exercises, hence it will pay of to put some effort into getting this right.)

First of all, make sure to combine all relevant information about a track (*i.e.*, tracking filter, track logic, and possibly track label) into one class or structure. These will be passed around different functions, and keeping it all in one object simplifies the bookkeeping considerably. Throughout the algorithm, keep tentative and confirmed tracks separated in two different containers. In many cases they are treated separately.

As a new scan arrives with new observations, do the following:

1. For all tracks: Predict the tracks to the time of the scan.
2. For all confirmed tracks:
   (a) Gate with the measurements in the scan, to obtain the validation matrix.
   (b) Create the association matrix.
   (c) Associate measurement to the tracks.
   (d) 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.)
   (e) Remove targets that are considered dead after updating the track logic.
   (f) 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 tentative tracks:
   (a) Repeat the steps for the confirmed tracks, this time using the set of tentative tracks.
   (b) Remove tracks that have been confirmed from the set of tentative tracks and add them to the set of confirmed tracks.
4. For all remaining measurements (not gated with any confirmed or tentative 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.

2.2 Global Nearest Neighbor (GNN) Tracker

Task: Implement a GNN tracker. Use the track logic from exercise 1 (select one method). An implementation of the auction algorithm has been supplied for your convenience, to allow for the focus to be on the tracking algorithm itself.

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, this way it is easier to predict the expected behavior and find where the algorithm breaks down.

Task: Apply the GNN 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.

2.3 JPDA

Task: Implement a JPDA tracker. Use the track logic from exercise 1 (select one method). An implementation of a function to compute the probabilities for associating a measurement to a specific target has been provided for your convenience, to allow for the focus to be on the tracking algorithm itself.

Task: Evaluate the JPDA tracker the same way as you just evaluated the GNN using simulated data.

2.4 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 $T = 0.26\,\mathrm{s}$ (between measurement scans), and the radar sensor was placed in the origin.

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