Human-Oriented Robotics
Prof. Dr. Kai Arras, Social Robotics Lab
Lab instructors: Timm Linder, Luigi Palmieri, Billy Okal

Winter term 2014/2015
University of Freiburg
Department of Computer Science

Submission: Send your solution via email to `palmieri@informatik.uni-freiburg.de` until February 3, 2015 with subject "[`exercises`] `Sheet 10`". All files (Matlab scripts, exported figures, hand-written notes in pdf/jpg format) should be compressed into a single zip file named `lastname_sheet10.zip`.

Exercise 10: Tracking and Data Association

You will need to download the Matlab frame `PeopleTrackingFrame.m` and `matchnnsf.m` and the log files `dataobs3.mat` and `dataobs4.mat` from the course website. The functions `predictstate.m` and `updatestate.m` from the last exercise are also provided.

Exercise 10.1: Getting Started, Plot Observation Sequences

In this exercise we extend the Kalman filter of the last exercise by the capability to handle multiple tracks using the nearest neighbor data association filter (NNSF). The result will be a full-blown people tracking system.

a) Start by loading the provided Matlab frame files and observation log files. Familiarize yourself with the variables that are read in, particularly the structure `obsdata`. We again assume the target has been observed at a constant frequency f and let $\Delta t = 1/f$.

b) Plot all observation sequences into a figure (**Hint:** use the line style pattern `'.'` with a large enough `'MarkerSize'` property value). Use the respective `zvalid`-fields to only plot the valid observations.

Exercise 10.2: Data Association: Nearest Neighbor Standard Filter (NNSF)

We now implement the NNSF for multiple targets. The algorithm differs only very slightly from the single-target variant and is described below.

a) **Data structures**. We suggest to use arrays of struct `tracks` that holds the track-specific information (e.g. state predictions and posteriors) for each track, and `obs` with the observation-specific information (e.g. $\mathbf{z}$-vectors) for each observation. These structs may have status-fields with values `'matched'` and `'non-matched'`. The `tracks` structure may also have a field `'obsid'` to store the associated observation per track.

b) **State prediction**. Set up the main loop over the observation sequence, and as a first step, predict the states of all tracks using `predictstate.m`. Make sure to store all the information $\mathbf{x}(k+1|k)$, $\mathbf{P}(k+1|k)$, $\hat{\mathbf{z}}(k+1)$, $\mathbf{H}$ and mark the tracks as `'non-matched'`.

c) **Observation**. Get the observations for the current time step. Note that not all tracks are observed in each step. Find the valid observations, copy their values into the `obs` struct and give them the status `'non-matched'`.

d) **Data association**. Implement the nearest neighbor standard filter (NNSF) using the interface `[tracks,obs] = matchnnsf(tracks,obs,ALPHA)` (α is defined in the frame) to associate observation with tracks. In each step, the multi-target NNSF carries out the following operations:

- Compute the $n_T(k+1) \times n_Z(k+1)$ Mahalanobis distance matrix D that stores the squared Mahalanobis distances between all actual and predicted observations. $n_T(k+1)$ and $n_Z(k+1)$ are the number of tracks and observations, respectively, in the current time step $k+1$.
- Iteratively find the minimum distance in D, test if it is smaller than the gating threshold from the cumulative χ^2 distribution (using `chi2invtable(ALPHA,2)`), and if yes, mark the pairing as matched and remove the corresponding row and column from D. Terminate if the minimum distance does not pass the validation test anymore. **Hint:** Instead of deleting the rows and columns, mark them with `Inf`.

e) **Update**. For all matched tracks, update their states using the function `updatestate.m`. For the non-matched tracks, copy the predictions into the posteriors.

f) **Initialize tracks**. We follow a greedy track initialization strategy and create new tracks every time an observation was not associated. For all non-matched observations, add a track to the `tracks` array of struct and initialize its state with the x and y values of `z` and 0 velocities. Use the provided `P0` for the initial state covariance. Like in the previous exercise, we are maintaining the history of all relevant variables per track. Here is the place to initialize them as empty matrices.

g) **Track histories**. Update the track histories with state $\mathbf{x}(k+1|k+1)$ and state covariance $\mathbf{P}(k+1|k+1)$.

Exercise 10.3: Plot Tracks

a) Plot the estimation histories of the track either into a new figure or on top of the observation sequences. Make sure you can distinguish the tracks from each other and from the observations, e.g. by using different colors and/or plot symbols.

b) Load the log file `dataobs4.mat` and re-run the tracker. Describe what happens. Why does it happens?