Christian Fischer Pedersen, cfp@ece.au.dk

1 Exercise: Location based services

1. What is a location based service?
2. Why is geographical position data relevant for location based services?
3. Discuss some of the common location based services you know
   - How do those services obtain position data (e.g. what sensor and network technologies are utilized)?
   - What level of position accuracy is required for those services to work (will that include/exclude some sensor or network technologies)?
   - Does those services collect and use location data for anything other than rendering the particular service (e.g. for adds or other third-party purposes)?
4. Discuss privacy issues in relation to location based services (e.g. how do we design a location based service with privacy in mind).

2 Exercise: RSSI as a distance metric

1. Understand what RSSI and dBm means
2. On your laptop, acquire WiFi RSSI data as a function of Tx–Rx distance, d, in two ways
   - Fixed distance data, $\mathbf{x}_{fix}$:
     600s of data sampled at 1Hz at distance d_{fix}
   - Multiple distances data, $\mathbf{x}_{mult}$:
     120s of data sampled at 1Hz at distances $D = (d_i)_{i=1}^{5}$ where $d_{i+1} - d_i = 0.10$m
3. For the **fixed** distance WiFi RSSI data, $\mathbf{x}_{fix}$
   - Conduct a mean and median filtering of the data with window sizes $ws = \{1, 3, \ldots, 11\}$ to achieve $(\mathbf{x}_{fix,mean,i})_{i \in ws}$ and $(\mathbf{x}_{fix,median,i})_{i \in ws}$ respectively
   - Plot $(\mathbf{x}_{fix,mean,i})_{i \in ws}$ and $(\mathbf{x}_{fix,median,i})_{i \in ws}$ to illustrate the filtering effects appropriately
   - Plot the mean and standard deviation of $\mathbf{x}_{fix,mean}$ and $\mathbf{x}_{fix,median}$ as functions of window length
   - Explain the plots above and the effects of filtering in your own words
4. For the **multiple** distances WiFi RSSI data, $\mathbf{x}_{mult}$
   - Mean and median filter, with window size $ws = 3$, the data to achieve $(\mathbf{x}_{mult,mean,d})_{d \in D}$ and $(\mathbf{x}_{mult,median,d})_{d \in D}$ respectively
   - With $(\mathbf{x}_{mult,mean,d})_{d \in D}$ and $(\mathbf{x}_{mult,median,d})_{d \in D}$ conduct the following experiment:
     Find the minimum Tx–Rx distance, d_{min}, needed to achieve a statistically significant distance separation at the 95% confidence level. Let the test statistic be the mean RSSI, i.e. null hypotheses are equal means. In other words, we want to find out how far we need to move (spatial resolution) the receiving antenna to/from the transmitting antenna in order for the RSSI mean values to have increased/decreased to such an extend that we can statistically conclude the movement.
   - Consider what the spatial resolution, d_{min} would be without filtering

- Discuss whether mean or median filtering is most appropriate for the RSSI spatial resolution case
- In your own words: under which type of noise conditions would you choose a mean filter over a median filter and vice versa?

3 Exercise: Absolute positioning by tri-angulation/lateration

1. Implement the triangulation algorithm "Algorithm 1: Final version of the ToTal algorithm", page 572 in reading:

   - Vincent Pierlot and Marc Van Droogenbroeck. "A New Three Object Triangulation Algorithm for Mobile Robot Positioning". In: IEEE, Transactions on Robotics 30.3 (2014).

   You do not have to consider the special cases described in "Algorithm 2" page 573. Run some numerical experiments and plot the results thereof to illustrate how the algorithm works.

2. Device your own trilateration algorithm (or re-implement an existing one) and run some numerical experiments and plot the results thereof to illustrate how the algorithm works. Consider using RSSI as a distance metric so you can take advantage of your exercise solution from earlier. Note, the RSSI and Tx-Rx distance are not proportional; use a more correct relationship if you can (hint: look at a common radio propagation model).

4 Exercise: Relative positioning by dead reckoning

Note: Matlab specifics are typeset with `typewriter` font.

1. Create a map with paths and dead reckoning points

   - Start point: Dept. of Engineering, Finlandsgade 22 (`wmmarker`)
   - End point: Railway station, Banegårdspladsen 1 (`wmmarker`)
   - Create suitable waypoints from start point to end point
   - Plot the straight line path from start point to end point on a map (`wmline`)
   - Plot the path along the waypoints from start point to end point (`wmline`)
   - Compute the distance in meters along both paths (`distance`). Find the distance difference
   - You plan to walk at noon from Dept. of Eng. to the railway station along the waypoints. Compute the dead reckoning (DR) points (`dreckon`)
   - Mark (`wmmarker`) the DR points and associate a note (`FeatureName`) with the DR timestamps

2. Re-implement the `dreckon` function. Adhere to the following requirements:

   - Speed entries are given in m/s rather than knots
   - Allow for slight symmetric variations, v_l, in the speed per leg, s_l, entries, i.e. $s_l \pm v_l$
   - Consequently, variations, d_i, should be determined for the DR points $(x_i \pm dx_i, y_i \pm dy_i)$
   - The variations in the DRs lead to a range of possible paths; the best case path, coincide with the normal `drecon` path, and the worst case path ends at point which is offset the most from the target end point, i.e. the railway station. Compute the worst case offset in meters.
   - Explain how your modified `drecon` works

3. Combination of absolute and relative positioning

    - Explain how it would make sense in a positioning system to combine your absolute positioning method (ToTal triangulation) with your relative positioning method (modified dreckon)

5 Exercise: Hybrid positioning by the Kalman estimator

1. Discuss how the Kalman filter fuses predictions with measurements.
2. The Kalman filter relies on that the product of two Gaussian functions is another Gaussian function.

    - Convince yourself about the correctness of this fact
    - Derive the expressions for the mean and the standard deviation for the product Gaussian
    - In Matlab, multiply two Gaussians and consider the results: What should the new mean be? Will the standard deviation be wider, narrower, or the same? Consider how this relates to fusing predictions with measurements.

3. In Python (or Matlab), implement a one dimensional Kalman filter. Add Gaussian noise to a constant (you can imagine this is, e.g. a sensor measurement of a one-dimensional location like in the train example in the reading) to simulate a *true* value and the *measurements* thereof. Apply your Kalman filter to *estimate* the constant.

    - How does the estimations converge (in absolute and relative terms) to the true value as a function of the standard deviation of the Gaussian noise?
    - What would happen to the convergence if the noise model was different from Gaussian?

4. Discuss how to implement the Kalman filter so that it merges the information from your relative and absolute positioning algorithms.

    - For those really keen: do the implementation.