# Project 2 - Adversarial Search – Artificial Intelligence

Teacher: Stephan Schiffel

March 9, 2021

Use the Piazza page, if you have any questions or problems with the assignment. Start early, so you still have time to ask in case of problems!

This assignment is supposed to be done in a group of 2-3 students; up to 4 is allowed. Note that everyone in the group needs to understand the whole solution even though you may distribute the implementation work.

You will need an installation of Python and the following libraries:

- `https://pykalman.github.io/`, a python library implementing a Kalman Filter,
- `https://traffic-viz.github.io/`, a library for handling and visualizing air traffic data,
- `https://pypi.org/project/geopy/`, for computing distances between geo-locations.

## Time Estimate

20 hours per student in addition to the time spend in the labs, assuming you work in groups of 2-3 students, did the previous programming assignment and and attended the lectures on Bayesian Networks and Temporal Probability Models or worked through the respective chapters in the book.

## Problem Description

## Kalman Filter math

A Kalman filter assumes that the true state at time $k$ is evolved from the state at time $k-1$ according to the following equation:

$$\boldsymbol{x_k} = \boldsymbol{F_k}\boldsymbol{x_{k-1}} + \boldsymbol{B_k}\boldsymbol{u_k} + \boldsymbol{w_k} \tag{1}$$

where

- $\boldsymbol{F_k}$ is the state transition model,
- $\boldsymbol{B_k}$ is the control-input model, which translates control input to changes in the state,
- $\boldsymbol{u_k}$ is the control-input,
- $\boldsymbol{w_k}$ is the process noise (uncertainty in the transition model), which is assumed to be drawn from a zero-mean normal distribution with covariance $\boldsymbol{Q_k}$

At time $k$ we make an observation $\boldsymbol{z_k}$ of the true state $\boldsymbol{x_k}$ according to the sensor model:

$$\boldsymbol{z_k} = \boldsymbol{H_k x_k} + \boldsymbol{v_k} \tag{2}$$

where

- $\boldsymbol{H_k}$ is the observation model, which maps the true state to the observation
- $\boldsymbol{v_k}$ is the observation noise, which is assumed to be drawn from a zero-mean normal distribution with covariance $\boldsymbol{R_k}$

For this project, you can assume a stationary process. Thus, transition model, process noise, observation model and observation noise are not time dependent and the $k$ subscript can be dropped from $\boldsymbol{F_k}$, $\boldsymbol{Q_k}$, $\boldsymbol{H_k}$, $\boldsymbol{R_k}$. Also, we have no knowledge of the control input, and therefore assume that $\boldsymbol{u_k} = \boldsymbol{0}$ and $\boldsymbol{B_k} = \boldsymbol{0}$.

This leaves us with the following model:

$$\boldsymbol{x_k} = \boldsymbol{F x_{k-1}} + \boldsymbol{w} \tag{3}$$

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}) \tag{4}$$

$$\boldsymbol{z_k} = \boldsymbol{H x_k} + \boldsymbol{v} \tag{5}$$

$$\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}) \tag{6}$$

To specify this model, you need to decide on

- the components of the state vector $\boldsymbol{x_k}$,
- the transition matrix $\boldsymbol{F}$, that is used to predict the successor state,
- the covariance matrix $\boldsymbol{Q}$ for the process noise,
- the components of the observation vector $\boldsymbol{z_k}$,
- the observation matrix $\boldsymbol{H}$, that is used to predict the observation from the state,
- the covariance matrix $\boldsymbol{R}$ for the observation noise.

There are numerous resources online that explain the structure and meaning of these components, for example, I found `https://www.kalmanfilter.net/` and `https://en.wikipedia.org/wiki/Kalman_filter` helpful.

## Tasks

1. Develop model for a tracking an airplane from radar measurements using a Kalman filter. That is, specify the state space, transition model and sensor model by specifying $\boldsymbol{x_k}$, $\boldsymbol{F}$, $\boldsymbol{Q}$, $\boldsymbol{z_k}$, $\boldsymbol{H}$ and $\boldsymbol{R}$.

   For now, assume that the airplane has a constant velocity and we are not interested in the height of the airplane, but only the position above ground. Thus, the state of the airplane should be modeled as position and velocity in 2d space.

   The radar measurement tells us the position of the airplane, but not the velocity. Assume that you get a radar measurement every $\delta t = 10s$ and that the measurement has a uniform standard deviation of $\sigma_o = 50m$ independently for $x$ and $y$. (Be prepared to change these parameters later.)

For the transition model, assume that we are tracking mainly passenger aircraft and they will normally not exceed an acceleration of $0.3g \approx 3m/s^2$. Thus we can think of the velocity deviating from on state to the next due to the acceleration of the airplane during maneuvers or due environmental influences (wind). Since we have no knowledge of the acceleration, we can think of acceleration in both x and y direction as independent mean centered Gaussian random variables with a standard deviation of $\sigma_p = 1.5m/s^2$ (thus assuming the acceleration is less than $0.3g$ in most cases). This leads to the following co-variances:

- the variances of the positions (both x and y) are $\frac{1}{4}\delta t^4 * \sigma_p^2$,
- the variances of the velocities (both x and y) $\delta t^2 * \sigma_p^2$,
- the co-variances of a matching pair of position and velocity (x with velocity x, or y with velocity y) are $\frac{1}{2}\delta t^3 * \sigma_p^2$,
- all other co-variances are 0.

2. Use the traffic library and get some of the sample flights. Understand the data-structure involved and plot the flights.

3. Use the provided code to turn the the data from a flight into simulated radar measurements.

4. Implement your kalman filter model using the pykalman library and use it to compute filtered position estimates for the flight from the simulated radar measurements. Use the first measurement as initial position.

5. Plot the filtered position estimates alongside the original track to make sure your filter is working. For this, you will need to convert the Cartesian coordinates into latitude/longitude, as is used in the original data.

6. Measure the error of the filtered positions, by computing both the mean and maximal distance of the filtered states to the positions in the original data at the same time point. The distance module in GeoPy might be helpful for computing the distances: `https://geopy.readthedocs.io/en/latest/#module-geopy.distance`.

7. Now that you have the filter setup, you should test it on a number of different flights and compare the errors you get for these flights. Which kinds of flights are worse? Which parts of the flights are harder to track? Why?

8. Conduct some experiments to investigate how the process noise ($sigma_p$) and observation noise ($sigma_o$) influence the error of the Kalman filter, that is, vary these two noise parameters and see whether the filtered positions get worse or better. Note, that there may be different results for the mean error and the maximal error. Which sets of parameters seem to work best on a wide range of different flights?

9. Use smoothing to compute estimates for all states based on all the available data. How much better are the smoothed tracks, compared to the filtered ones? Are there specific areas in the flights where smoothed tracks are better?

10. **Bonus** Change your model to include altitude and vertical velocity, that is, track the position and velocity of the airplane in 3d space.

11. Write a short report containing:
    - a description of your model
    - a description of the experiments you ran, the results you got and an interpretation of these results

## Handing in

You must hand in this assignment through Canvas. Hand in a PDF report and a ZIP archive with your code.