Kalman Filtering for Pose Tracking

COSC450 Assignment 1

Marks: **20%** Due: **Wednesday 1 May, 11:59pm**

1 Overview

The naïve AR system provided (in ~`steven/Public/ARCode/`) tries to detect the checkerboard in every frame. The last successful detection is used to provide the pose estimate for the camera. As discussed in lectures, Kalman filters can be used to provide more consistent tracking through short periods of occlusion and to reduce the effects of noise in the individual frames' measurements. For this assignment you will:

- *Implement* a basic Kalman filter for pose tracking,
- *Conduct experiments* to determine good parameters for the Kalman fitler
- *Implement* one or more variants of the basic Kalman filter,
- *Conduct experiments* to compare different tracking approaches,
- *Document your findings* in a technical report.

2 Implementing a Kalman Filter

The sample code has two different tracking methods (subclasses of the `Tracker` class). The first, `LastObservedTracker`, just remembers the last successfully determined pose from checkerboard detection, and uses that as the current estimate. The second, `KalmanTracker`, is the same in the sample code, but you should update it to implement the Kalman filter. You will need to implement the two versions of the `update` method.

The first version of the `update` method takes no parameters and is used in the case where there has not been a pose estimate made due to failed checkerboard detection. In this case the state and its covariance can be updated as

$$\tilde{\mathbf{s}}_t = \mathrm{A}\tilde{\mathbf{s}}_{t-1}$$
$$\mathrm{P}_t = \mathrm{A}\mathrm{P}_{t-1}\mathrm{A}^\mathsf{T} + \mathrm{Q}_t$$

The second version of the `update` method takes a measured pose as a parameter, and requires the full Kalman update to integrate this measurement into the state estimate as

$$\begin{aligned} \tilde{\mathbf{s}}_t^- &= \mathrm{A}\tilde{\mathbf{s}}_{t-1} & \mathrm{P}_t^- &= \mathrm{A}\mathrm{P}_{t-1}\mathrm{A}^\mathsf{T} + \mathrm{Q}_t \\ \tilde{\mathbf{m}}_t &= \mathrm{B}\tilde{\mathbf{s}}_t^- = \mathrm{B}\mathrm{A}\tilde{\mathbf{s}}_{t-1} & & \\ \tilde{\mathbf{s}}_t &= \tilde{\mathbf{s}}_t^- + \mathrm{K}_t(\mathbf{m}_t - \tilde{\mathbf{m}}_t) & \mathrm{P}_t &= \mathrm{P}_t^- - \mathrm{K}_t\mathrm{B}\mathrm{P}_t^-, \end{aligned}$$

where the Kalman gain is given by

$$\mathrm{K}_t = \mathrm{P}_t^-\mathrm{B}^\mathsf{T}\left(\mathrm{B}\mathrm{P}_t^-\mathrm{B}^\mathsf{T} + \mathrm{R}_t\right)^{-1}.$$

A matrix package, `Eigen`, is included which may be useful. Documentation for that package is available at `http://eigen.tuxfamily.org/index.php`.

You may need to add some member variables to store the various state estimates and covariances, and to add code to the constructor, destructor, and/or intialisation methods. You may also wish to add command-line parameters, which should be documented via the `printHelpMessage` function in `ARSkeleton.cpp`, as well as in your report.

2.1 Experiments 1: Determining KF Parameters

The performance of the Kalman filter depends on the choice of the matrices Q and R which specify the covariance in our state update and measurement functions respectively. You should design and conduct experiments to determine good values of Q and R and to compare your `KalmanTracker` to the `LastObservedTracker`. Questions you should ask when designing these experiments include:

- How can you tell if you have 'good' Tracker?
- What range of values for Q and R should you test?
- What experiment(s) might you conduct to evaluate different trackers/values?
- What sequence(s) should you test the trackers on?
- What results would indicate a good/better or bad/worse tracker/value?
- Is it possible to determine any answers without experiments?

You should clearly describe the design of your experiments in your report, as well as the results and conclusions you draw from those results.

3 Variations on the Kalman Filter

Next, you should implement one or more variants of the basic Kalman filter, and compare them to the `LastObservedTracker` and `KalmanTracker`. Possible variants include:

- Different motion models (e.g. constant velocity vs. constant acceleration)
- Different rotation representations (e.g. axis-angle vectors, quaternions, Euler angles, etc.)
- Using an Extended Kalman Filter (EKF) with a non-linear state-update and/or measurement function*
- Using partial measurements from corner detection during occlusion*
- Using a particle filter rather than a Kalman filter*

Those marked * are more complicated, so any one of those would be sufficient to earn full marks. If you choose the simpler options, then you should do several variations. If you have other ideas you'd like to try out then talk to me about how hard they're likely to be before proceeding too far.

To implement these you will likely have to add more command-line parameters and/or further subclasses of `Tracker`. Be sure to document this clearly in your report and via the `printHelpMessage` function.

3.1 Experiments 2: Comparing Tracking Methods

Your second round of experiments should compare the basic Kalman filter to your variant(s). The same sorts of considerations apply as in the first round of experiments. It is OK to re-use experiments from the first round. In fact this is good – suppose you ran an experiment to compare the `LastObservedTracker` and the `KalmanTracker`. Running the exact same test on a `VariantKalmanTracker` would let you then compare the variant tracker to both of the others. If you gave careful thought to how you can tell if a tracker is good or not in the first round, then the same approach should apply in the second round.

4 Deliverables and Marking Scheme

You should submit (via email):

- The source code for your tracker.
- A CMakeLists.txt file that allows me to build your solution on the lab machines.
- A report in PDF format.

Please do not submit executables, build directories, files that some operating system has randomly decided to create, etc. If you want to package your submission in an archive format (such as `submission.zip`), then be aware that the University spam filters block .zip extensions (among others). They are, however, easily fooled so renaming files `submission.piz` will get around it. Submissions sent from University accounts should not have this problem.

If you wish to include videos or other large media files, then please make them available via a file-sharing site (Dropbox, Google Drive, One Drive, etc.) or on YouTube or similar rather than emailing them.

4.1 Suggested Report Structure

A suggested structure for your report is given below. You may wish to vary this depending on exactly what you choose to do for the assignment, but the points given cover the assignment requirements.

1. Abstract/Introduction – give an overview of your work and what you chose to do for the assignment.

2. Kalman Filter Implementation – discuss how the basic Kalman filter is implemented and give instructions for running your program to track using the basic Kalman filter.

3. Experiments 1 – discuss the design of your experiment(s), give the results you found, and draw conclusions. When is your Kalman filter better/worse than the default tracker? What are the best values for Q and R?

4. Variant Filters – discuss the variant(s) that you chose to implement, how they differ from the base filter, and give instructions for running your program using the variant filter(s).

5. Experiments 2 – discuss the design of your experiment(s), give the results you found, and draw conclusions. It is OK to re-use experimental approaches from the first round of experiments.

6. Discussion and Conclusions – sum up your findings and discuss what you have learned from this assignment.

You should make good use of mathematics, figures, tables, and illustrations in your report to make your ideas and results clear and precise. Your report should be as long as it needs to be, but no longer. I expect that the main sections would be 2-3 pages each, depending on the number and size of illustrations.

4.2 Marking Scheme

20 marks are available for this assignment, distributed as follows:

5 Marks Basic Kalman filter implementation

5 Marks Experiments to determine Kalman filter parameters

5 Marks Choice and implementation of variant filter(s)

5 Marks Experiments to evaluate your variant filter(s)

Marks for the implementation are based on your code and the description in the report. Your code should be correct, clear, concise, and efficient *in that order*. A clear implementation is much preferred over one that is efficient due to a convoluted implementation. Comments, use of appropriate programming abstractions (classes, methods, access modifiers, functions, etc.) and so forth are expected.

Marks for the experiments are based solely on your report. I will be looking for clear descriptions of what experiments you did, why you chose to do those experiments, what the results were, and what conclusions you draw from them.