Assignment 4: Kanade-Lucas-Tomasi (KLT) Tracker. Vision and Image Processing

Kim Steenstrup Pedersen, François Lauze, and Jan Kremer

January 7, 2014

This is the fourth mandatory assignment on the course Vision and Image Processing. The goal is for you to get familiar with programming for computer vision with a focus on a classical 2 dimensional tracking algorithm.

You have to pass this and the following mandatory assignments in order to pass this course. There are in total 4 mandatory pass/fail assignments. This is a **group assignment**, i.e., we expect that you will form small groups of 2 to 4 students that will work on this assignment.

The deadline for this assignment is Monday 24/01 2014. You must submit your solution electronically via the Absalon home page. Go to the assignments list and choose this assignment and upload your solution prior to the deadline. Remember to include your name and KU user name (login for KUnet) in the solution. If you do not pass the assignment, having made a SERIOUS attempt, you will get a second chance of submitting a new solution.

A solution consist of:

- Your solution source code (Matlab / Python scripts / C / C++ code) with comments about the major steps involved in each Question (see below).
- Your code should be structured such that there is one main file that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions / classes.
- Your code should also include a README text file describing how to compile (if relevant) and run your program, as well as a list of all relevant libraries needed for compiling or using your code. If we cannot make your code run we will consider your submission incomplete and you may be asked to resubmit.
- The code, auxiliary files and README file should be put into a compressed archive file in either the ZIP or tar format (RAR is not allowed - we simply cannot read your archive).
- A PDF file with notes detailing your answers to the questions, which may include graphs and tables if needed (**Max 5 pages** text including figures and tables). Do NOT include your source code in this PDF file.

A note on relevant software

We recommend that you select either Matlab / Python / C / C++ as the programming language you use for your solutions for these assignments. We also recommend that you select the language you are most confident in - the focus should be on learning the methods and not learning a new programming language.

If you wish to use Matlab, the University of Copenhagen has a license agreement with MathWorks that allow students to download and install a copy of Matlab on personal computers. On the KUnet web site you find a menu item called Software Library (Softwarebiblioteket):
`https://intranet.ku.dk/Sider/Software-bibliotek.aspx`.
Under this menu item you can find a link to The Mathworks - Matlab & Simulink + toolboxes. Click this link and follow the instructions for how to install on your own computer.

If you use Python or C / C++ we recommend that you use the OpenCV library `http://opencv.org` and / or the VLFeat library `http://www.vlfeat.org/`. Both libraries provide an extensive collection of implementations of central computer vision algorithms. OpenCV provides an interface for Python. Follow the installation instructions on these websites to install on your own computer. OpenCV is also available via MacPorts on MacOSX.

If you wish to program your solutions in C++ we recommend the use of the CImg Library, which can be obtained at `http://cimg.sourceforge.net`

Kanade-Lucas-Tomasi (KLT) Tracker

The goal of this assignment is to implement and experiment with the classical Kanade-Lucas-Tomasi (KLT) feature tracking algorithm as described in [2]. A copy of the original paper is available for download in Absalon. Read the paper carefully and then make an implementation of the outlined algorithm. We suggest that you use the parameter choices outlined in the paper. Please indicate in the report what choices you make for the different parameters including the choice of weight window function $w(\mathbf{x})$.

A possible extension is to include the features introduced by Shi and Tomasi [1] (OpenCV has an implementation, see `GoodFeaturesToTrackDetector`).

1 Testing

Run your implementation on a couple of video sequences of your choice. You can for instance use the sequence you find at `http://www.cs.toronto.edu/~fleet/research/DataSoftware/dudek.tar.gz` or use OpenCV video capture capability to capture your own video sequence. Do the parameter choices from [2] generalize to your video sequences?

Try to tune the parameters such that your implementation provides good results on your video sequences.

Illustrate your results for both experiments with a couple of frames with the matched features drawn on top of the video image.

2 Extension (Optional)

Try to come up with an approach to tracking an object by combining all features on the object of interest. That is, how would you form an object region (i.e. a bounding box or circle) from the feature points on the object?

Explain your approach and illustrate your implementation on one or more video sequences by showing selected frames.

References

[1] J. Shi and C. Tomasi. Good features to track. In *Proceedings of CVPR'94*, pages 593–600, 1994.

[2] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.