Parallel Visual Computing 2012/13
Assignment 3

November 9, 2012

For all tasks that you address, you have to provide a short documentation as well as present it to the class in our exercise meeting.

1 Mandatory Tasks

As discussed in the lecture, there are several ways to speed up the Difference-Of-Gaussians detector that is used in the SIFT pipeline. In the first part of the assignment you will apply those improvements to the naive and slow code provided to you. You can find the Makefiles and Projects under `vision/scale_space` in the zip file provided to you. In the second part you will apply additional parallelizations and optimizations to the code.

For more information on the background we refer to the following resources:

- "SIFT"-paper by David Lowe (IJCV): `http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf`
- Andrea Vedaldi's SIFT page: `http://www.vlfeat.org/api/sift.html`
- Helpful Gaussian Identities: `http://www.cs.nyu.edu/~roweis/notes/gaussid.pdf`

1.1 Incremental Blur (5 Points)

Right now the code always uses the original image and blurs it to the desired scale. This results in a very huge blur kernel for the later scales. Therefore, your task is to replace this approach by an incremental blurring, where you take your last blurred image and apply a much smaller gaussian to achieve the desired scale. Recall that instead of sequentially filtering with σ_1 and σ_2 and can achieve the same results by blurring with $\sqrt{\sigma_1^2 + \sigma_2^2}$. Report speed up.

1.2 Pyramid (5 Points)

Have a look at Figure 1 in the "SIFT"-paper by David Lowe which is referenced above. Another speed-up is described that down-samples the image after each octave. Within one octave you have applied enough blur to the images that you can safely discard every other pixel. Please note that you also have to half the sigma in order to take into account the downsampling. Implement the pyramid and report the speed-up.

1.3 Parallelization of Filtering (5 Points)

Parallelize the filter operations by using OpenMP as discussed in the lecture.

Measure you speed up factor w.r.t. the provided code for a varying number of threads. Plot mean and standard deviation over 5 runs.

1.4 Additional Parallelization (5 Points)

Profile your code (you can do this single threaded) and identify the slow parts. Propose and measure parallelization and optimization of this part.

2 Optional Tasks

2.1 Benchmark the OpenCV implementation (5 points)

OpenCV also provides an implementation of SIFT. Benchmark the detector against your implementation. Also check if their code is parallelized. OpenCV can be compiled with OpenMP and TBB. Make your best effort to make the settings comparable (scale range and steps).

2.2 Explore Intel Threading Building Blocks (TBB) (5 points)

TBB is another option to provide parallelism in your code. Parallelize the filter operation using TBB and present a brief comparison to openmp in terms of learning curve, simplicity, efficency.

2.3 Multi-Core Profiling (5 points)

Pick a multi-core profiling tool and show how to profile parallel code with it and briefly document your experiences with it. There is a wide range of tools available that support multi threaded profiling (visualstudio, xcode, tau, ...).

2.4 Spatial interpolation (5 points)

The current implementation suffers from poor spatial localization of the interest points at higher scales. As also described in the original paper, the exact location of the points should be obtained with sub-pixel accuracy by interpolation. Implement and parallelize/optimize this scheme.