Multi-Core Architectures and Programming

Speeding up autopano-sift with CUDA

Wolfgang Schnurrer
Christopher Dreher

uni@wschnurrer.de christopher.dreher@web.de

Hardware/Software Co-Design
University of Erlangen-Nuremberg

25. September 2009
Agenda

1 Introduction
2 Implementation
3 Results
4 Conclusion
Basics

autopano-sift automatically creates control points for groups of overlapping photographs.
Basics

autopano-sift automatically creates control points for groups of overlapping photographs.

generatekeys
uses the sift algorithm to find important keypoints on every image
  ► load image to a ImageMap structure (CUDA)
  ► SIFT-LoweDetector (partially CUDA)
    * upscale image with b-linear interpolation (CUDA)
    * gaussian convolution (CUDA)
    * ... further steps in chapter SIFT algorithm on page 5
Basics continuation

autopano

generates panorama project file (pto) from keypoint data of all images.

▶ keypoint matching
  * keypoints are merged into a large kd-tree (128 dimensions)
  * for every point a „nearest neighbour“ is searched (BBF) and the matches are grouped in partitions\(^1\)
  * partitions are checked for geometric consistency using an algorithm called RANSAC\(^2\)

▶ successful matches are grouped into partitions and combined in control points

---
\(^1\) this searching step is still the most time consumptive and forbids real-time application use of the SIFT algorithm
\(^2\) RANdom SAmple Consensus [http://www.npac.syr.edu/projects/](http://www.npac.syr.edu/projects/)
SIFT algorithm I

The Scale Invariant Feature Transform algorithm, proposed by D. Lowe in 1999 identifies visually distinct features (keypoints) in an image and creates Feature Vectors. These features are invariant to image scale and rotation.

Abbildung: The keypoint is defined by its location x, y, scale and orientation.
SIFT algorithm II - The 5 steps

1 Convert into intensity, up-sample and prefilter (CUDA)

Abbildung: step 1

2 Build Gaussian image pyramids and calculate DoG

Abbildung: Difference of Gaussian Pyramids (one octave), nach [3]
3 Keypoint detection

Abbildung: find local minima and maxima, aus [3]
4 Compute feature orientations
5 Compute feature descriptors (vectors)

Abbildung: keypoint descriptor, aus [3]
Short Examples

Abbildung: unfiltered keypoints

Abbildung: matched keypoints without filtering
Short Examples cont.

Abbildung: panorama with 4x 640x480 images
Code Design

source of our project was autopano-sift-c which is a part of the hugin project

+ no .net/mono environment needed
- unlovely code due to the porting from the object oriented code

high-level transparency

reuse the source project and apply the same functions and signatures to achieve an easy way to switch between cpu and gpu computation.

```c
#ifdef CPU
#ifdef GPU
#ifdef DEBUG
```

```c
```
libgeneratekeys_cuda.so is our shared library which links to the existing autopano-sift-c

```c
#ifdef GPU
    CUDA.vars.globalpointer=CUDA.init_struct(pic->width,pic->height);
    DisplayImage_ConvertToImageMapDevice_cuda(pic);
#endif    //GPU

//der alte aufruf, der die imagemap wieder runtergeladen hat
//ImageMap* picMap1 = DisplayImage_ConvertToImageMap_cuda(pic);
#ifdef DEBUG
    ImageMap* picMap1 = CUDA.Download_D2H_ImageMap();
#endif    //DEBUG

#ifdef CPU
    CPUTIME_START
    ImageMap* picMap = DisplayImage_ConvertToImageMap(pic);
    CPUTIME_STOP
#endif    //CPU
```
Speedup

Timings (in seconds): Computation of a 2816x2112 image

(1) ImageMap
   CPU 0.100000
   GPU 0.000801 (factor 125)

(2) scaleDouble
   CPU 0.320000
   GPU 0.213126 (factor 1.5)

(3) prefiltering with Gauss
   CPU 6.510000
   GPU (0.203379+0.202111)=0.40549 (factor 16)
Plot

Abbildung: GPU vs. CPU Timings

University of Erlangen-Nuremberg
Wolfgang Schnurrer   Christopher Dreher

design
Abbildung: Livedemo: How to make a nice stitched panorama image
Impressions

* image processing is a big field for parallel computation
* missing memory management leads to hours of debugging
* porting a complete project to cuda and evaluate the results takes a lot of more time than the week we had
References

M. Brown and D.G. Lowe.
Automatic panoramic image stitching using invariant features.

S. Heymann, K. Muller, A. Smolic, B. Frohlich, and T. Wiegand.
SIFT implementation and optimization for general-purpose GPU.

D.G. Lowe.
Distinctive image features from scale-invariant keypoints.

GPU-based video feature tracking and matching.

A. Vedaldi and B. Fulkerson.
VLFeat: An open and portable library of computer vision algorithms.

C. Wu.
SiftGPU.
Web: http://cs.unc.edu/ccwu/siftgpu.
Questions