UNIVERSITY OF BATH
Department of Computer Science

Fast Dense Pixel Correspondence using DAISY descriptors and OpenCL

Author: Ioannis Panousis
ip223@bath.ac.uk
Supervisor: Matthew Brown
m.brown@bath.ac.uk

Abstract

Correspondence is one of the fundamental blocks of computer vision applications and is concerned with finding matches of the same real objects across different images.

Its dense application aims to find matches for every pixel in two or more images to aid high level applications such as; **structure from motion, object categorisation and 3D reconstruction.**

However until now the dense computation of features for every pixel is proven too expensive and higher level applications use sparse correspondence, i.e. only a few pixels, often called keypoints, but at the high cost of;

robustness and accuracy

This project makes a coupling of recent appearances in vision and GPU programming in order to make this vital operation reach real-time speeds. The first is a new, faster and reliable descriptor algorithm called DAISY [1] and the second is the recent standard by the Khronos Group called OpenCL [2], for generic processor and platform programming.

Why use DAISY?

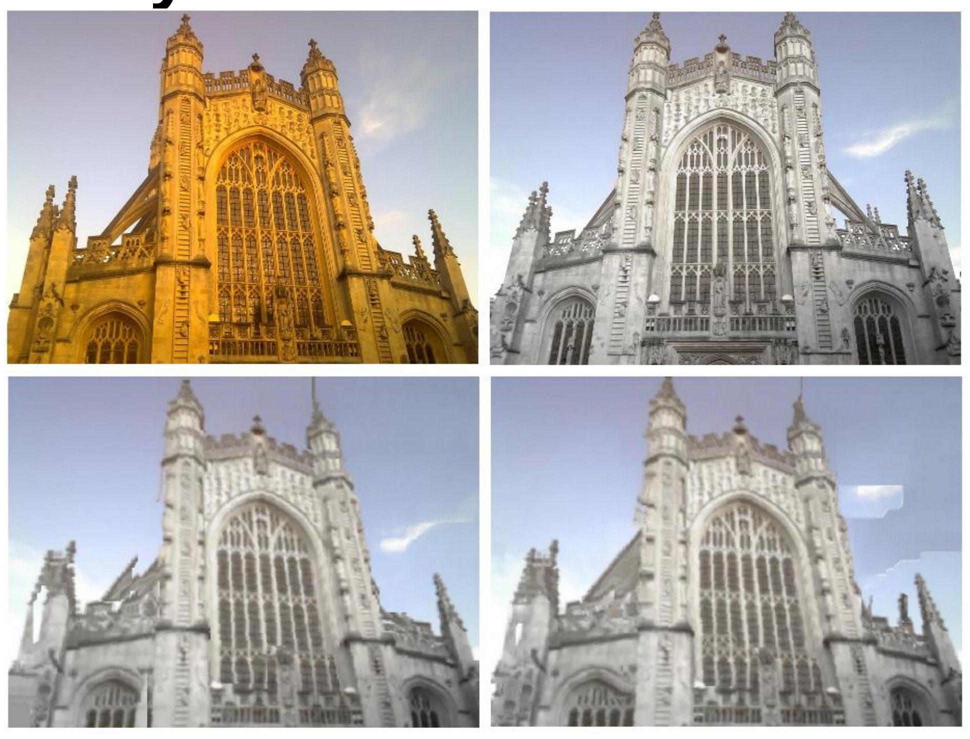

Figure A) Two pixel warping outputs, using Scale Invariant Feature Transform (SIFT) features to produce the bottom left perspective and DAISY for the bottom right. The discovered match disparities between the top left and top right are used to construct a new image that has the perspective of the former image but the image data of the latter. The results of SIFT and DAISY look very similar, with DAISY even catching the second pylon on the far right.

Firstly, its constituent operations naturally map onto a parallel architecture and due to its efficient computation design it has already been proven **faster** than other descriptors [1].

It is just as crucial to ensure DAISY **maintains the high quality** standard of performance set by the SIFT descriptor and it has also been proven to surpass it [1].

DAISY's quality is demonstrated in figure A on the bottom right image. Incidentally, the DAISY inputs used in that case were products of the developed OpenCL program.

Simply the access to the hundreds-fold increased processing power of a GPU over a multi-core CPU does not immediately lead to performance gains.
The DAISY algorithm has to be tailored top to bottom to the fit the GPU model. See the key factors that affect each stage.

Speed Factors

Costs
Global Memory Latency (100c)
Local Memory Latency (10c)
Local Memory Bank Conflicts
Local Memory Space
Register Space
Boosts
1. Global Read Coalescence
2. Global Write Coalescence
3. Conflict-free Local Access
Tuning
Workgroup size
Work steps per workgroup
Strategies
Compute/Transfer Overlap
CPU/GPU work allocation
Kernel Code Generation*

* = not employed

Stage 1: Gradients
Dimensionality of data increases by a factor of 8 as multiple-orientation gradients are computed.

Serialised memory accesses is a major thread here, prevented using coalescence.

Boosts employed: all.

$$\frac{dI^+}{do}$$

Stage 2: Convolutions
While the separable and consecutive properties of Gaussian convolutions do improve efficiency, careful memory and compute patterns on this large data volume are absolutely critical for speed.

Tuning of work load per thread was done.

Boosts: all.

$$((\mathrm{I}_o * G_1) * G_2) * G_3$$

Stage 3: Transpose A
The full data volume needs to be first transposed, in preparation for the final DAISY descriptors.

The GPU is not intended for such large data shifts but with appropriate design of memory access the full bandwidth can be used with great impact.

Implementation is made open to speedups of newer GPUs with larger memory segment sizes.

Boosts: all.

Stage 4: Transpose B
This most complex operation has two primary concerns;
1) a factor of 8 data dimensionality increase
2) the 8-way scattering of the data locally in memory.

The first is not favoured by the GPU model and the second is not favoured by the backup plan; coalescence.

The complexity of this operation is handled in part by;

- Generating lists, on the CPU, of coalescable data and the impact of memory on GPU is mitigated.

Boosts: 1,2.

1864 MPixel/s **1429 MPixel/s** **212 MPixel/s**

Results

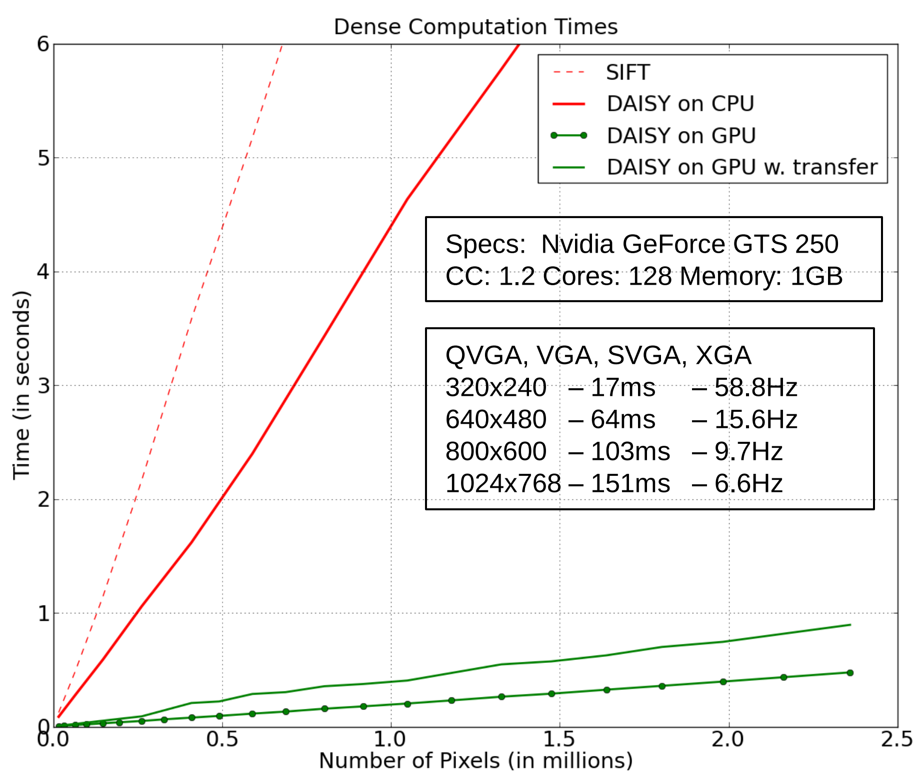

Figure B) Benchmarking of GPU DAISY against its CPU equivalent on 1 core and the dense SIFT fast implementation of the VLFeat library [3]. The GPU operation is measured with and without the data transfer back to RAM.

Implementation in C/C++ and OpenCL proves up to 22.7 times faster than the CPU C++ version, averaging 20.8 in the figure B ranges. Transferring the results to RAM is more costly, but, with strategies of pinned memory and overlap of computation and transfer, the data transfer impact is mitigated.

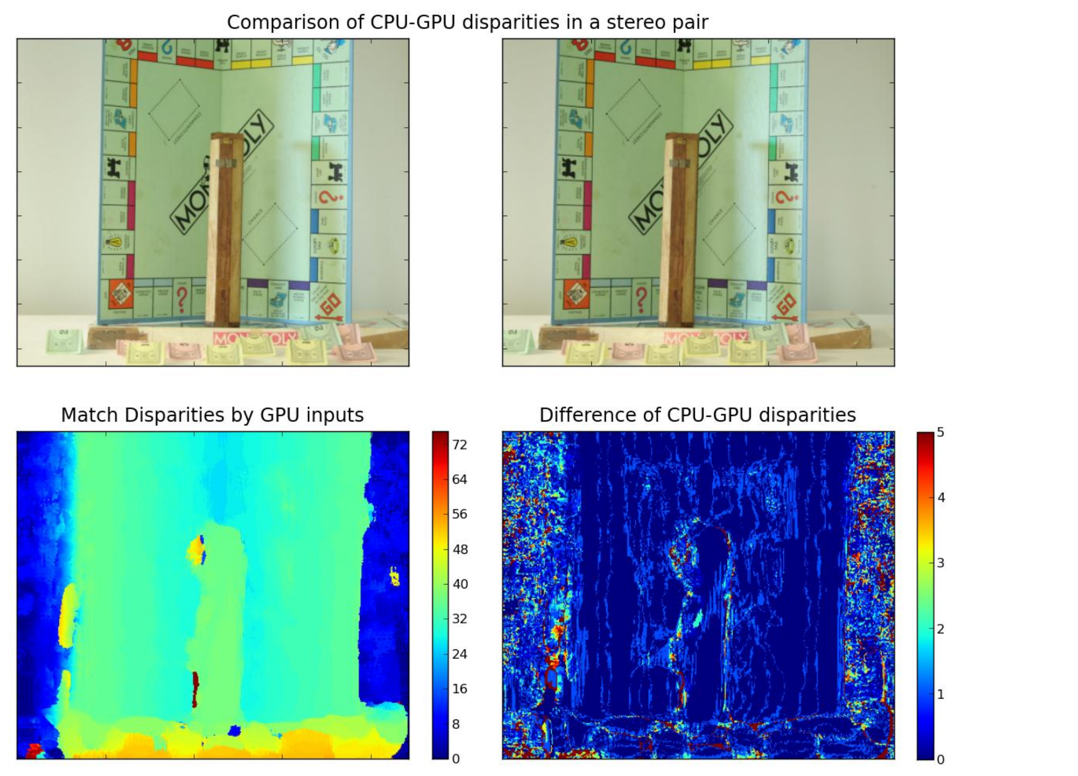
Comparison of CPU-GPU disparities in a stereo pair
Match Disparities by GPU inputs
Difference of CPU-GPU disparities
72
64
56
48
40
32
24
16
8
0
5
4
3
2
1
0

Figure C) Validation of GPU implementation by disparity measurement between the top left and top right images. The Euclidean disparities resulting from GPU inputs are on bottom left. Their difference from CPU inputs on the bottom right. A naïve nearest neighbour search was used.

Amongst the methods used to validate the GPU implementation output, one is shown above where discovered disparities between standard images [4] are compared with the CPU version of DAISY. The percentile of the differences in the bottom right that lies below 5 pixels is 96.9%, a very good result. Similar tests with ground truth data were done, figure A output is also another functionality demo.

Conclusions

- Huge speed-up over existing dense feature algorithms
- Enabling of real-time higher level processing that is robust and accurate
- Cross-device and cross-platform
- Any higher vision application is interested in processing on GPU

Further potential

- Gaussian pyramids, greatly reduce data volume (approx. by 32) to the huge benefit of performance speed
- Kernel code generation, more compiler arguments can reduce register usage for greater speeds
- New GPUs will perform even better, CPU versions are largely limited in the respect of hardware evolution

References

[1] Tola, E., 2010. DAISY: An efficient dense descriptor applied to wide baseline stereo. IEEE Transactions on PAMI May, 2010.
[2] "OpenCL - The open standard for parallel programming of heterogeneous systems" [online]. https://www.khronos.org/opencl/
[3] "VLFeat - Home" [online]. https://www.vlfeat.org
[4] "vision.middlebury.edu" [online]. vision.middlebury.edu/stereo