A General Toolkit for "GPUtilisation" in SME Applications

David Williams Valeriu Codreanu Jos Roerdink

Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen

Introduction

We work with European SMEs who would like to utilise the power of modern GPUs in their software. They have use cases in several application domains:

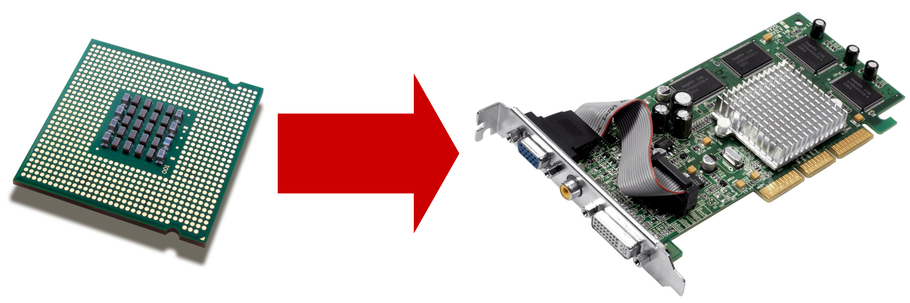
- Computer aided diagnosis (MedicSight, UK)
- Image-based search (Rotasoft, TR)
- Image forensics (Imagemetry, CZ)
- Medical data processing (B3C, IT)

However, programming GPUs is complex. Most SMEs do not have access to the specialist skills which they need to take advantage of these devices.

Translating code from CPUs to GPUs

We aim to provide an automatic tool which can convert C/C++ codebases into OpenCL or CUDA. This is being built on the existing Rose[1] and Mint[2] platforms.

- The source code is annotated with compiler directives to give clues about which parts of the algorithm are suitable for execution on the GPU.
- We test the system with the real-world code provided by the SME partners.
- To implement this system we are undertaking research in areas such as *memory management, loop pattern classification and dependency analysis.*

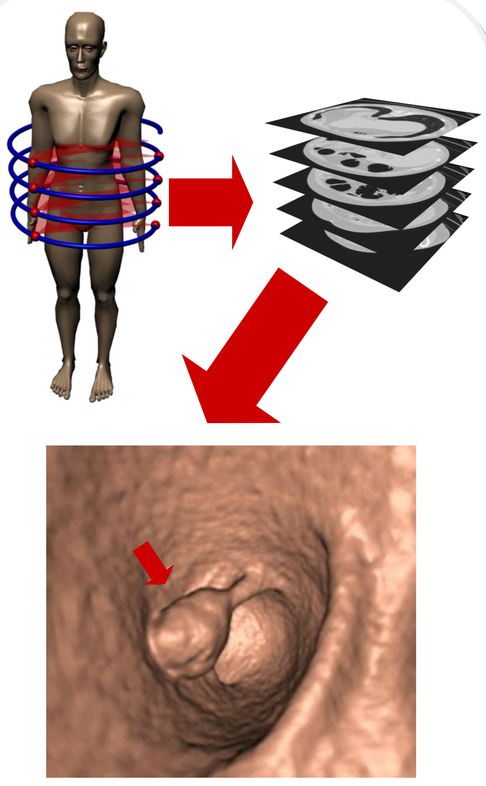


Use Case: Medical Imaging

MedicSight uses curvature analysis to identify potential colon polyps in CT data. Such analysis is computationally expensive but falls within the range of applications which GPUs can accelerate. Our GPU implementation:

- Reduces the execution time from 5 minutes to 1.2 seconds.
- Consumes significantly less memory than the original versions.

There are further opportunities to improve the accuracy of the result now that more computational power is available.

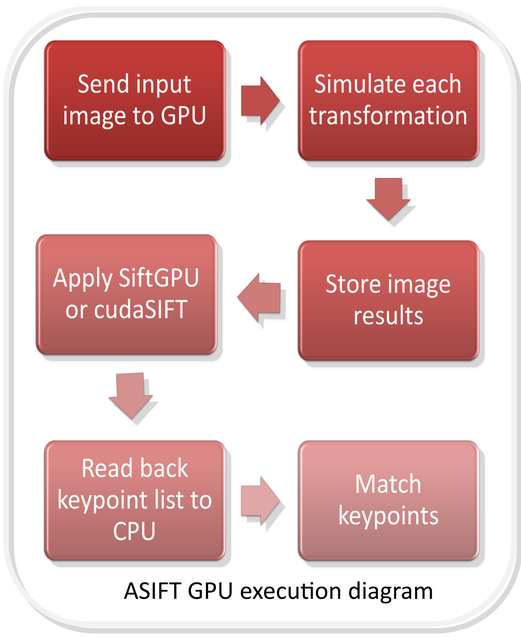


Data obtained from a CT scan of the patient (top) can be used to identify polyps (bottom). Such polyps typically have specific curvature characteristics.

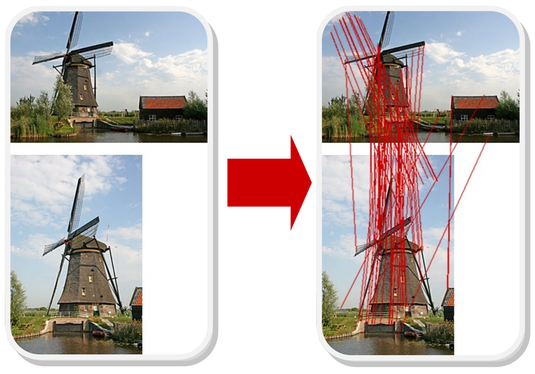
Use Case: Computer Vision

Rotasoft uses the powerful ASIFT algorithm to detect and match keypoints of images present in AR (augmented reality) books. ASIFT is very computationally expensive, but is a good fit for the GPU. Our GPU implementation:

- Is based on two distinct state-of-the-art SIFT GPU implementations.
- Achieves up to 120x speed-up compared to the CPU implementation.
- Is scalable and has potential for real-time detection using a multi-GPU approach.
- Results' accuracy is almost identical.



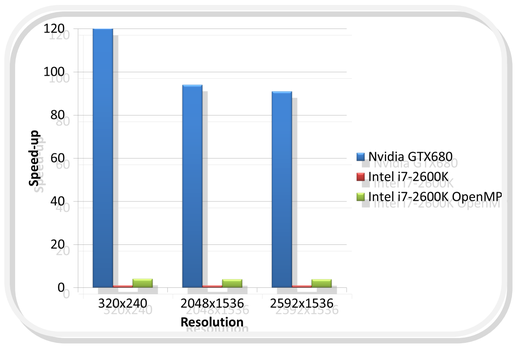
ASIFT GPU execution diagram



ASIFT keypoint matching

Results

Manual parallelization of source code has already demonstrated a huge potential for performance improvement across all use cases. Initial work on automatic translation has produced functional versions which reproduce some of that improvement. Future work will improve the translation in areas where it is currently sub-optimal.



References

[1] www.rosecompiler.org

[2] D.Unat, X.Cai, and S. Baden, *"Mint: Realizing CUDA performance in 3D Stencil Methods with Annotated C"*, ICS'11

Research funded by the EU, FP7-SME-2011-1 - 286545 - GPSME

See: *http://www.gp-sme.eu/*