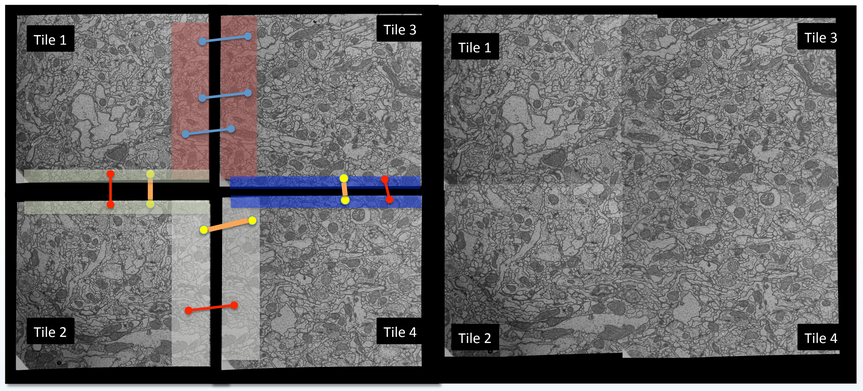
Investigating GPUs as a faster way to Stitch Multi-Terabyte ssTEM Image Data

Roy Rinberg
Janelia Research Campus, Howard Hughes Medical Institute, Ashburn, VA USA

Introduction

To create the panoramic of multi-terabyte image data, overlapping images need to be stitched together; this demand is ubiquitous among all kinds of image data. To study the connectome, large 3D images need to be analyzed, which can then be used to elucidate the anatomical connectivity among neurons. We present a strategy to make the current image stitching process faster through the implementation of Graphics Processing Units (GPUs) to parallelize the process without a loss in quality of stitching.

At Janelia Research Campus, terabytes of Transmission Electron Microscopy (ssTEM) and optical microscopy image data need to be stitched together to create 3D image volumes. The algorithm currently used at Janelia runs on a large number of CPUs in a distributed fashion where each process will find point matches between pairs of images through cross-correlation. Then, a database of point matches are ingested into a database and then used to stitch the images together, into a mosaic.



Our Aim

Our aim is to utilize GPUs for the largely repetitive and simplifiable tasks of finding key features and matching key points between images. In addition, we seek to investigate if efficient memory reuse through intelligent caching is possible, or if this problem does not lend itself to GPU calculations effectively.

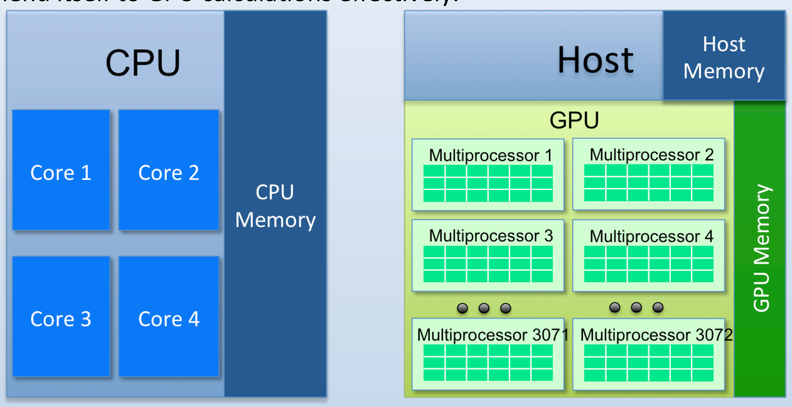
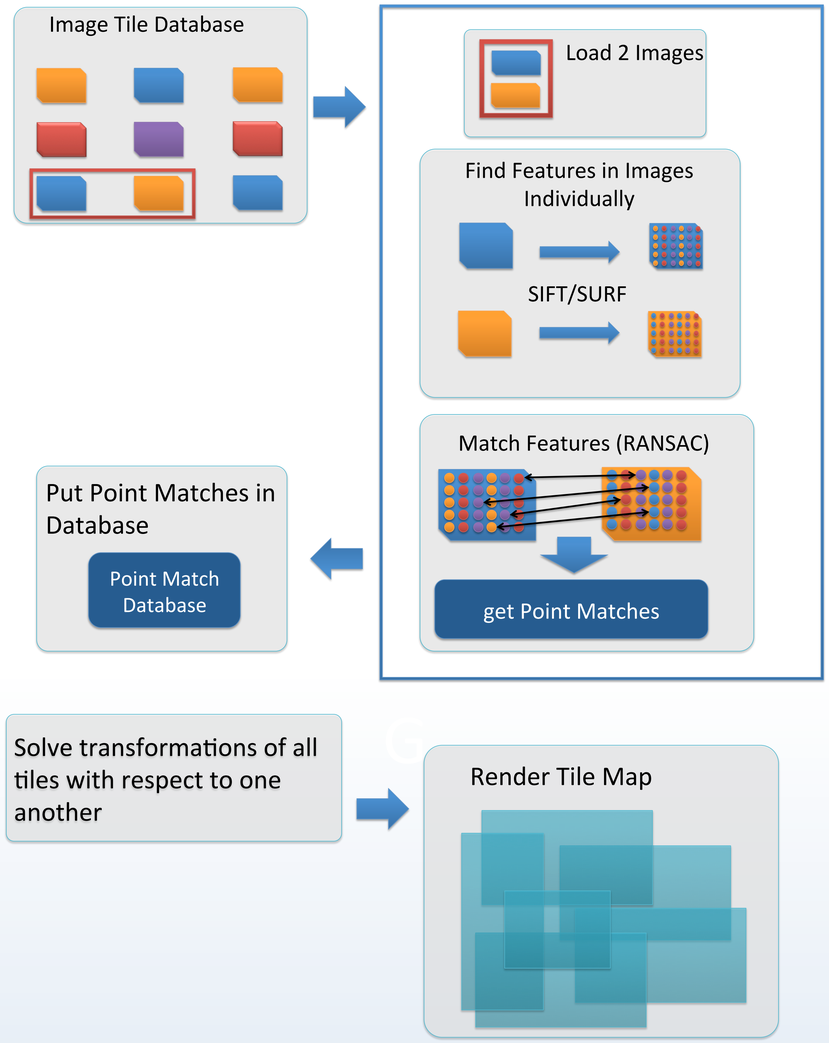


Image Stitching Pipeline



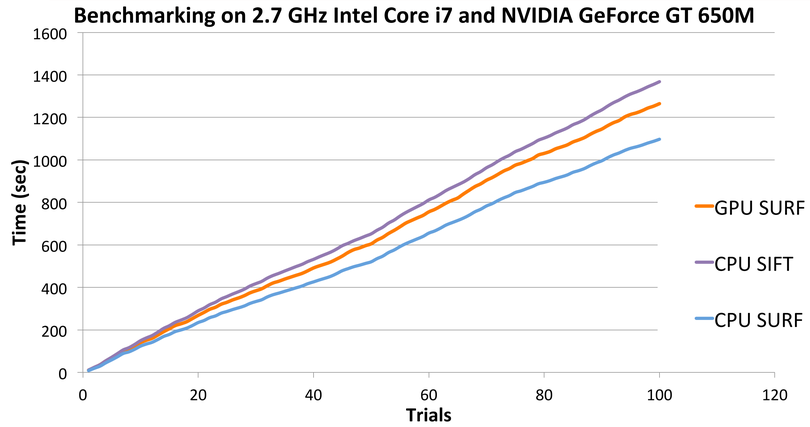
The program created operates on any PC regardless of OS[1]. As well as Janelia's cluster which operates on Scientific Linux v6.x. The program operates on the latest version of OpenCV (3.0.0) and libcurl (7.43.0).

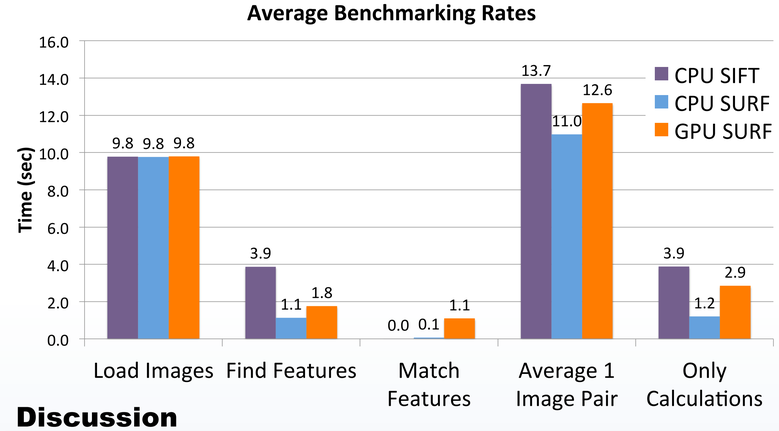
All external information is communicated through HTTP calls. Image files are retrieved through HTTP calls to a rendering service; through this service, the appropriate filters are applied to the image requested. Images take significantly longer to load (up to 9 seconds), but no file system structure is needed to be maintained.

Because there is no dependence on location or file system structure, the program easily scales to any size data set, and can be run on any device.

[1]The GPU code must be run on a CUDA-enabled device

Benchmarking





Discussion

Preliminary results lead us to believe that the serial, CPU algorithm works better. However, this data is rather preliminary and still needs to be compared to the cluster's benchmark tests.

There are more than 8x cores on an NVIDIA Titan X (3072 blocks) than on the Mac OSX's GeForce GT 650M (372 blocks). With up to 32 Titan X nodes at our disposal the preliminary benchmarking offers little information about performance on the cluster.

The real question becomes not, is a GPU faster, but will the advantage the expensive NVIDIA cards give, be worth their cost?

References

Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features." Computer Vision and Image Understanding (CVIU) 110.3 (2008): n. pag. Print.

Fatahalian, K., J. Sugerman, and P. Hanrahan. "Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication. *Graphics Hardware (Stanford) (2004): n. page. Print.*

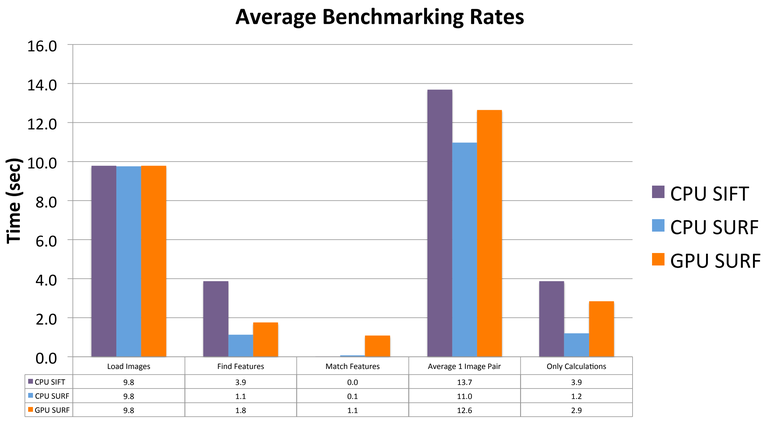
Lowe, David G. "Distinctive Image Features from Scale-Invariant Keypoints." International Journal of Computer Vision (2004): n. pag. Print.

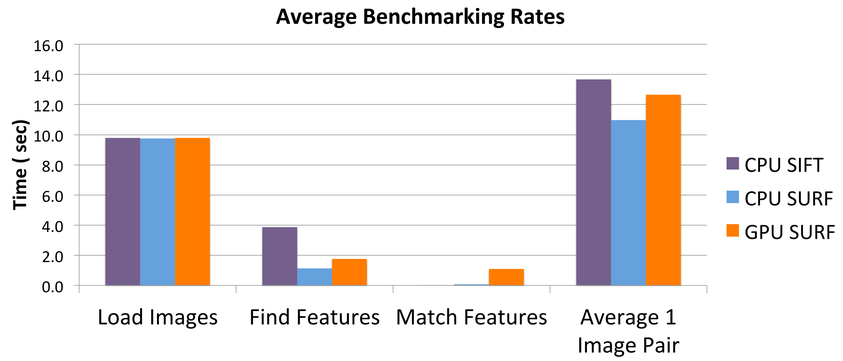
Acknowledgements

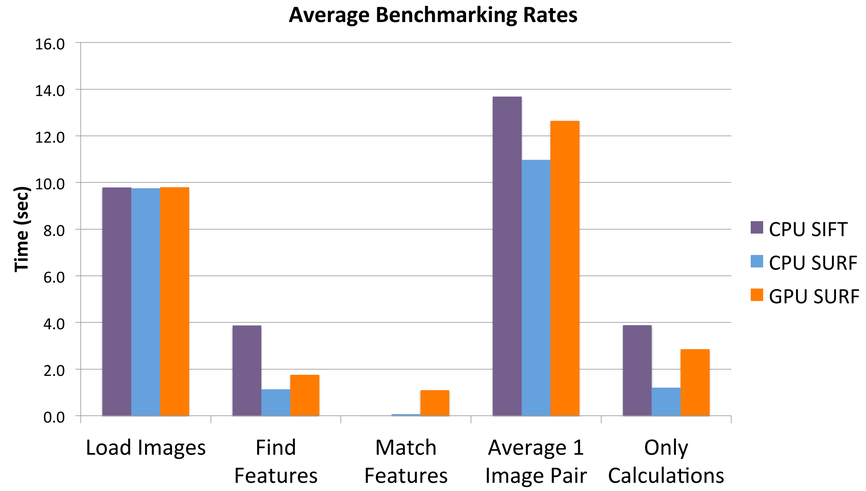
Khaled Khairy, Sean Murphy, Yang Yu, Eric Trautman Ken Carlile, Nathan Soules, Mark Bolstad, Tom Dolafi, Chandan Singh

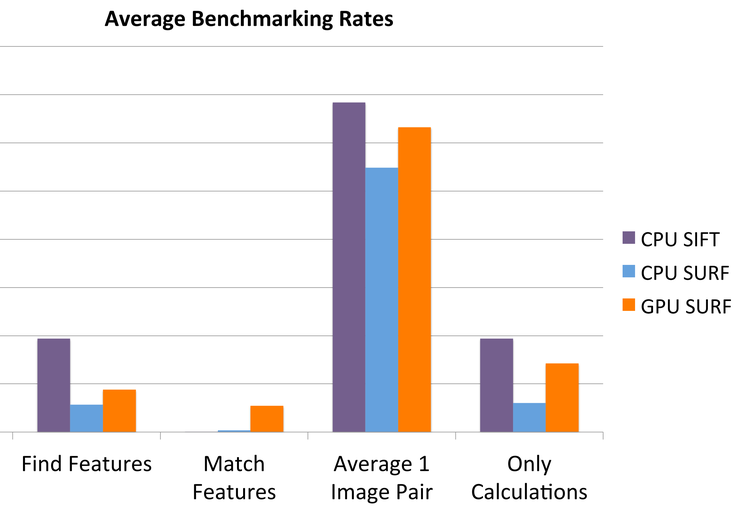
Contact Information

Roy Rinberg – royrinberg@nyu.edu
https://github.com/RoyRin


Average Benchmarking Rates
Time (sec)
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0
CPU SIFT
CPU SURF
GPU SURF
Load Images
Find Features
Match Features
Average 1 Image Pair
Only Calculations
CPU SIFT 9.8 3.9 0.0 13.7 3.9
CPU SURF 9.8 1.1 0.1 11.0 1.2
GPU SURF 9.8 1.8 1.1 12.6 2.9


Average Benchmarking Rates
Time (sec)
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0
CPU SIFT
CPU SURF
GPU SURF
Load Images
Find Features
Match Features
Average 1 Image Pair


Average Benchmarking Rates
Time (sec)
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0
CPU SIFT
CPU SURF
GPU SURF
Load Images
Find Features
Match Features
Average 1 Image Pair
Only Calculations


Average Benchmarking Rates
CPU SIFT
CPU SURF
GPU SURF
Find Features
Match Features
Average 1 Image Pair
Only Calculations