GPU Application to Real-Time Astronomical Image Processing Pipelines

Applying massively parallel graphics processing to rapid image processing for computationally intensive algorithms

Variable Object Study

(Temporal Astronomy)

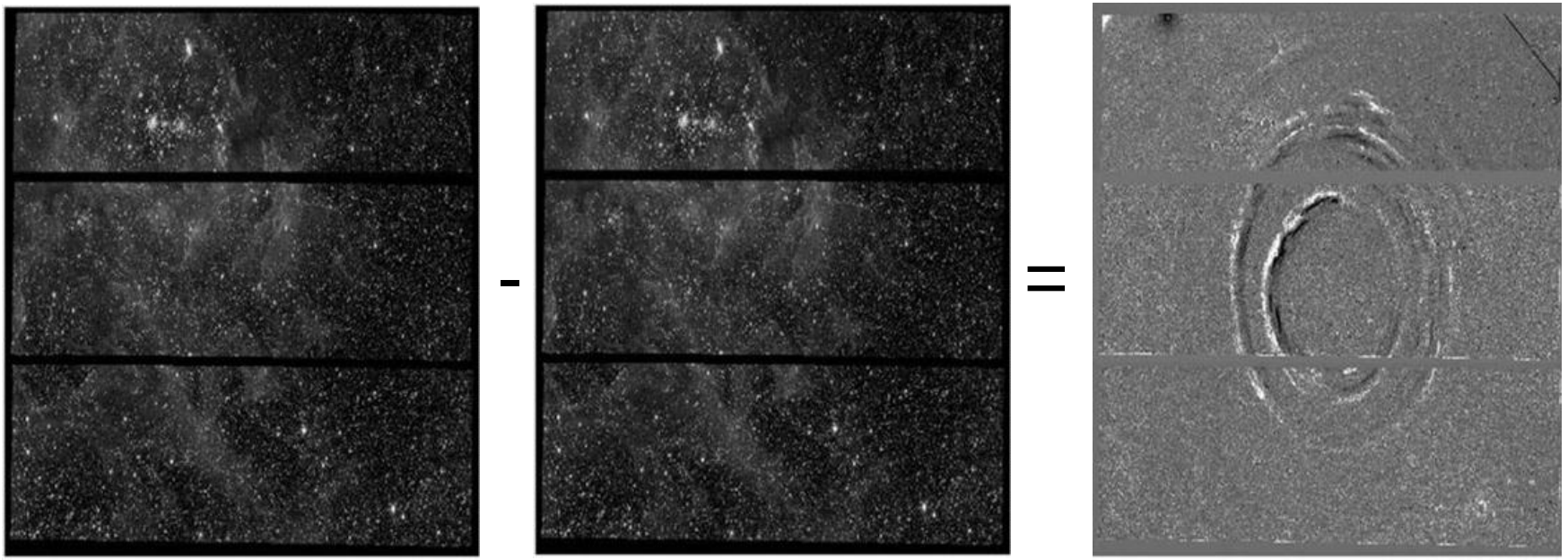
- Changes in magnitude or position on human timescales
  - seconds to centuries
  - principal tool is differential photometry
- Dynamic celestial phenomena
  - Variable stars, eclipsing binaries
  - Supernovae
  - Active Galactic Nuclei (AGN)
  - Moving solar system bodies (comets, asteroids, moons, minor planets)
  - Exo-planet transits
  - Proper motion
  - Outburst events (GRB, star forming regions, …)

Image Subtraction

Discovery and analysis that goes beyond blinking and aperture photometry

- Pixel-by-pixel subtraction
- Differential photometry on an entire image
- Finds localized changes in magnitude
- Identifies and characterizes moving and fixed position variables targets

Subtraction on right shows progress of a light echo from an earlier nova in the star field on the left



Miller, Pennypacker, & White, 2008, PASP Source images: Astronomical Research Institute

Convolved Image Subtraction

Find a transfer function that redistributes the flux point spread function (PSF) to match, with no change in net flux for a target object. Kernel *K* convolved with image *R* matches image *I* when no variable objects are present:

$$R \otimes K = I$$

The difference image D then characterizes all variable objects:

$$\left(R \otimes K\right) - I = D$$

Challenge: produce the best possible kernel.

Optimal Image Subtraction (OIS)

- Assume a basis function
  - Alard (ApJ 1998, A&A 1999,2000) uses a Gaussian basis in the original implementations
  - Miller (PASP 2008) introduced optional Dirac delta function basis (better asymmetric fit)
- Kernel formed by a superposition of scaled basis functions

$$K = \sum_i \left[a_n \left(x, y \right) \right]_i B_i$$

- Solve the kernel by a least-squares fit until PSF in $R \otimes K \approx I$ across a large number of sampled non-variable image locations

The Rising Data Torrent

170 GB

Image source: SDSS and the Astrophysical Research Consortium

Image data per night of operation

PS1 1-1.5 TB
PS4 4-6 TB

Image source: Institute for Astronomy, University of Hawai`i, Photograph by Rob Ratkowski for the PS1SC

30 TB

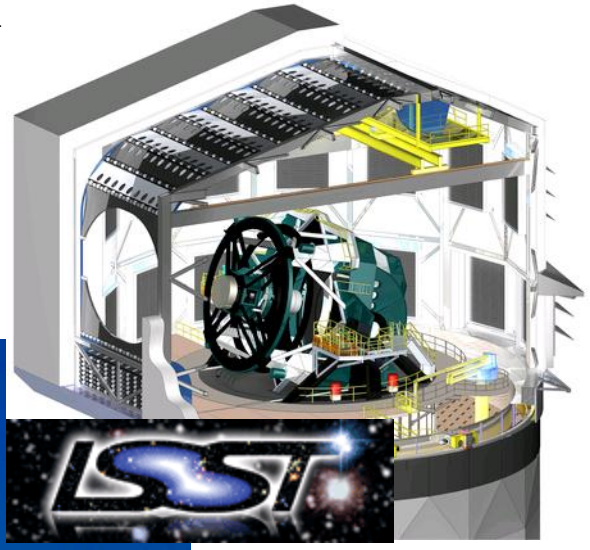


Image source: LSST Corporation, Image credit: J. Andrew, NOAO/LSST

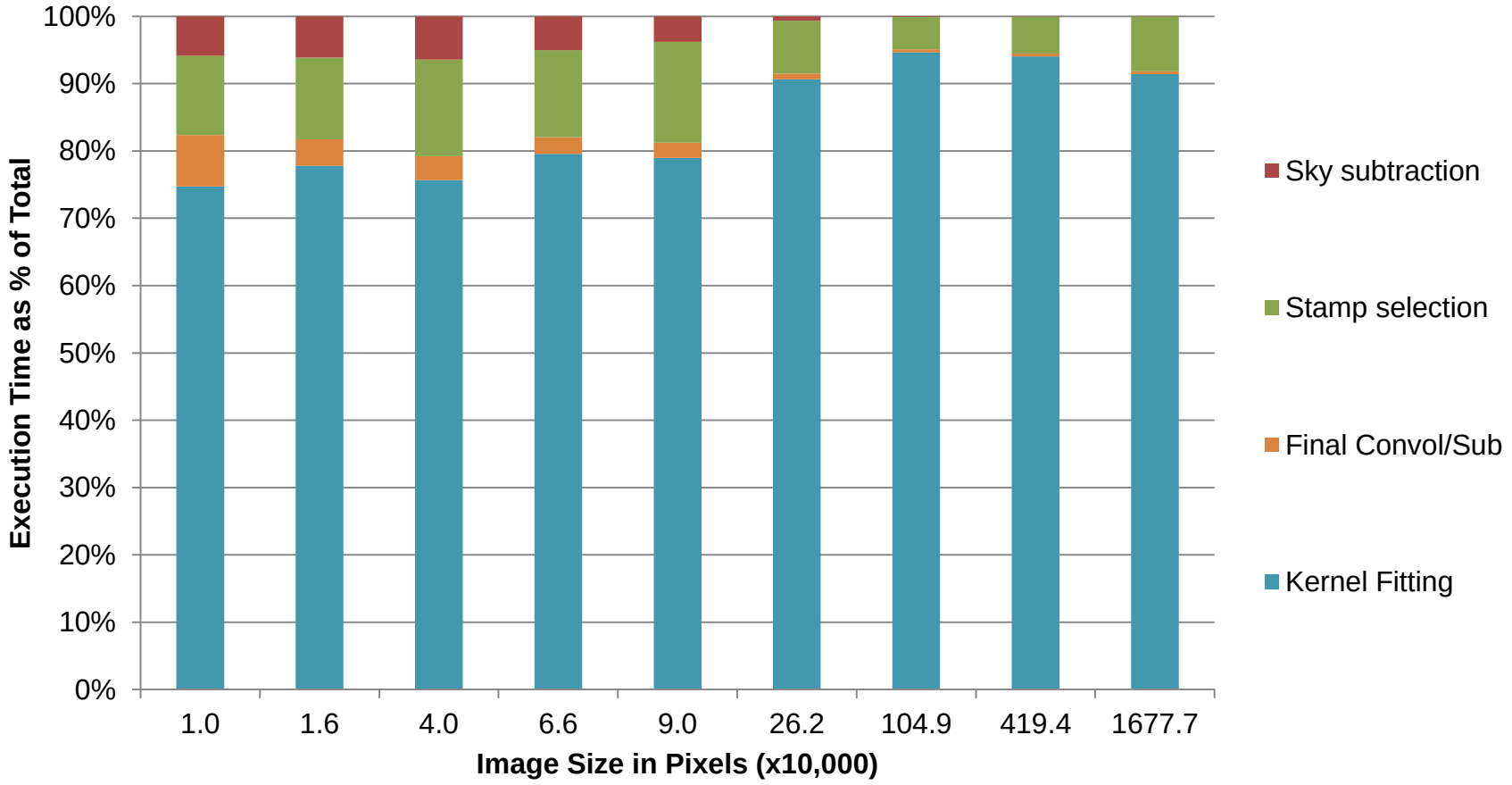
In Pan-STARRS and LSST nearly all of the images will be differenced at least once.

OIS – computation requirements

- As exemplified by Pan-STARRS, 1.4 Gpx mosaic camera
  - Using Miller-Buie OIS IDL code
    - Miller's IDL OIS optimized by Buie (and his Harvey Mudd team)
  - 2nd order space-varying kernel fit for a single tile 4800x4800 pixels
    - ~3 TFLOP/subtraction for a tile image pair
    - ~100 sec on a modern high-performance PC under IDL
      - » Operating at ~30 GFLOPS
  - 60 tiles in a Pan-STARRS image [PS1 takes ~1 image/2 min]
    - ~180 TFLOP/subtraction for a complete image pair
    - ~6000 sec (1.67 hrs) on the same high performance PC as above

Computation Time by Operation

(Miller-Buie IDL OIS, Constant Gaussian Kernel)



Convolution Load

Convolution is the major processing load in OIS. Hundreds to hundreds of thousands of small convolutions for determining the kernel based on small sample regions (*P* in number). The kernel is developed as a superposition of basis functions determined by a least-squares fit:

$$F = \sum_{k=1}^{P} \sum_{i=1}^{\eta_k} \sum_{j=1}^{\eta_k} \left(\sum_{n=1}^{N} \left(a_n(x,y) \left(R_k \otimes K_n \right)_{i,j} \right) - \left(I_k \right)_{i,j} \right)^2$$

Where a perfect kernel is found when $\nabla F = 0$

Plus the time for the convolution in the final full difference image *D*:

$$\left(R \otimes K \right) - I = D$$

Amdahl's Law Scaled By Problem Size

Amdahl's speedup function is most meaningful for a fixed problem size. If we apply it to execution time:

$$t_{\|} = t_0 / S(P,N) = t_0 \left[(1-P) + P/N \right]$$

Where *t* is the time of execution. Time is the real challenge we must address, not S.

But the problem size is growing.

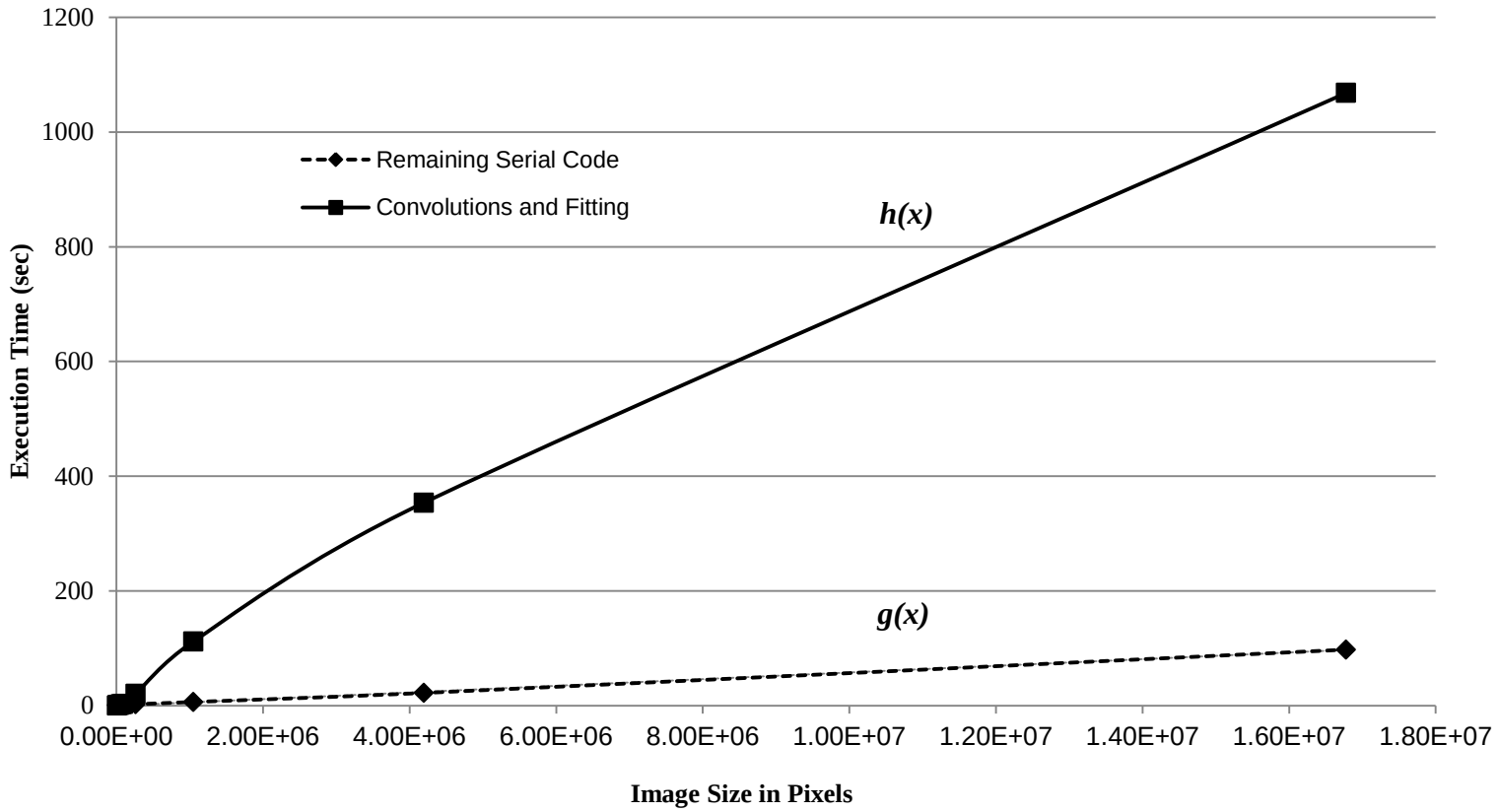
$$t_x = t_{01} f(x,P,N) = t_{01} \left[g(x)(1-P) + h(x)P/N \right]$$

Where *f(x,P,N)* is the effective problem increase as a function of image size in pixels *x*; g(x) and h(x) are the serial and parallel affected execution scaling functions respectively which increase proportional to x.

- When possible, reformulating the code so that g(x) is constant is the ideal case

Computation Time by Class

(Constant Gaussian Kernel)

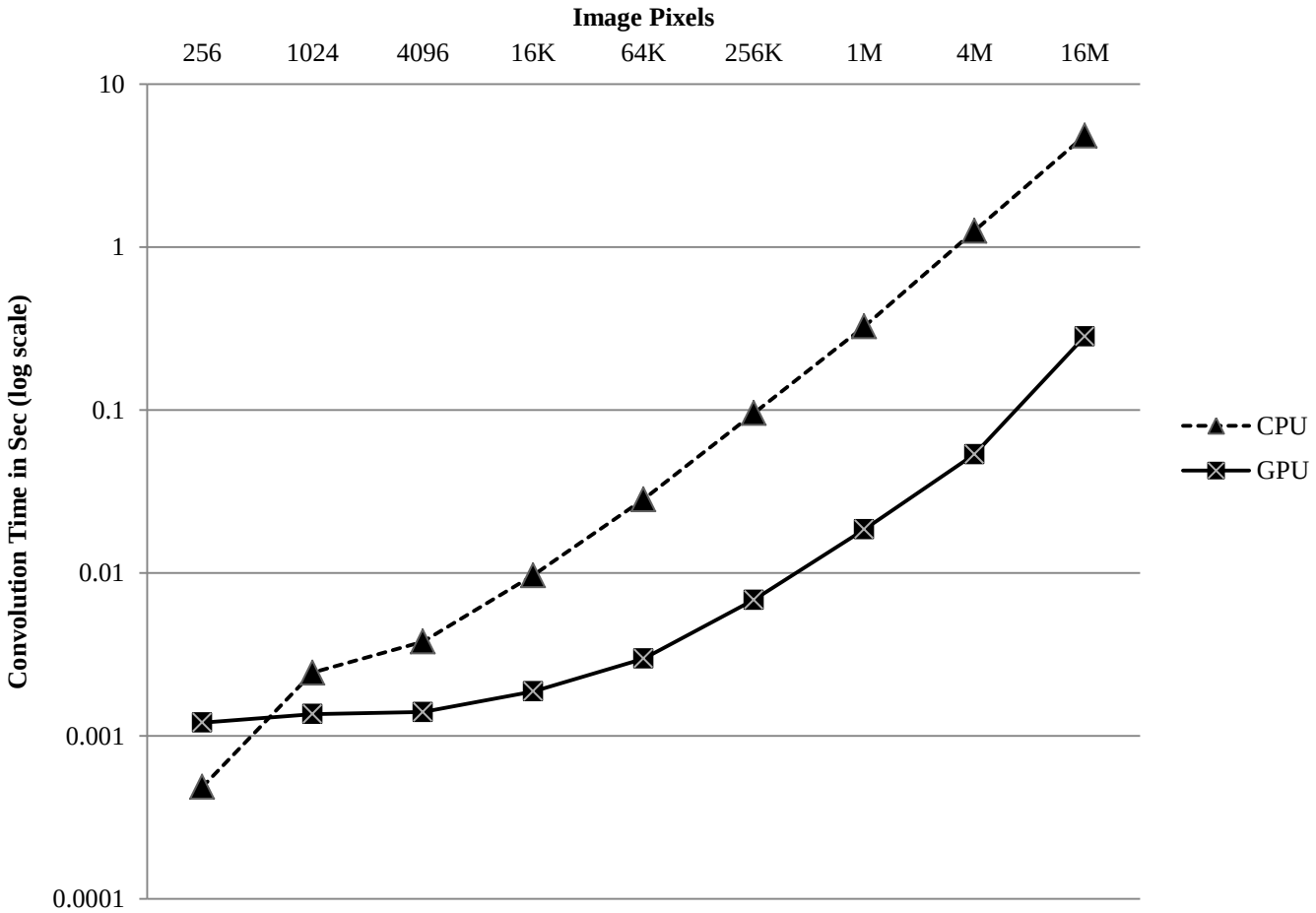


GPULib for IDL

- Commercial Product
  - High-level-language (HLL) implementation GPU API
  - Free for academic use
  - HLL frontend for CUDA, requires CUDA and NVIDIA hardware
- IDL code must be modified for GPULib calls
  - GPU variables (explicitly or implicitly declared) are required for GPU operations
  - GPULib library is a small subset of IDL, complex IDL cannot be replaced one-for-one
  - Limited dimensionality in many functions, call parameters different in some cases
  - Code must be applicable to a SIMD paradigm (same serial to GPU parallel challenge as everywhere else)
  - Operator overloading for IDL v8.0 (a nice feature)
- Every function call is a new CUDA kernel
  - Memory transfer and kernel initiation overhead apply
  - Problem must be stated in a way that minimizes function calls or overhead swamps out gains
- Implicit (hidden) GPU memory allocations must be managed in some cases
  - GPU memory resources can be consumed rapidly by some calling methods
  - A CPU variable can be morphed into a GPU variable if needed by a GPULib function (allocating GPU global memory)
  - Implicit variables may need to be freed manually to recover memory, especially when working with large arrays

GPULib vs CPU Convolution Time

(Constant Gaussian Kernel)



Conclusions

- OIS is well suited for improvement by a GPU solution
- Growth in time of execution is dominated by code that can be parallelized effectively
- GPULib can achieve a 10x-20x gain for OIS on larger images
  - But probably not much more without heroics
  - Further gains are best approached in a native GPU development environment
- In general, characterizing Amdahl's Law in terms of time that scales by problem size for the problem at hand can provide insight into practical optimizations, and limits of optimization with respect to problem size increases

Thank you,

Questions?