Streaming Multiframe Deconvolution of Atmospherically Distorted Images on GPUs

idies Institute for Data Intensive Engineering and Sciences Johns Hopkins University

Matthias A Lee & Tamás Budavári
The Johns Hopkins University

NVIDIA CUDA CENTER OF EXCELLENCE

NVIDIA

Motivation

Atmospheric distortion is an issue in various fields relying on long-distance imaging, especially in astronomy. Earth-based telescopes face the challenge of peering though the ever-changing and blur-inducing atmosphere. This can be especially detrimental to observations of small and faint objects. The point spread function (PSF) defining this blur is unknown, hard to predict and varies wildly making recovery more difficult.

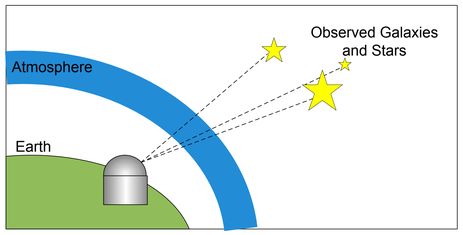

Figure 1: Earth-based telescopes have the disadvantage of peering through earth's ever-changing and blur inducing atmosphere. This is especially detrimental for faint and distant observations.

Modern telescopes produce high volumes of extremely large images, upwards of 100PB in 10 years, yielding an even more compute intensive and time consuming reconstruction. The current "State-of-the-Art" images are commonly Co-Add images made by overlaying multiple faint observations to create one higher quality image. (See Figure 2) These Co-Add images are very noisy and missing detail. More advanced reconstruction is computationaly complex and therefore slow and laborious. We need GPU-accelerated, statistically sound and extensible tools to keep up with, explore and deblur these images in real time.

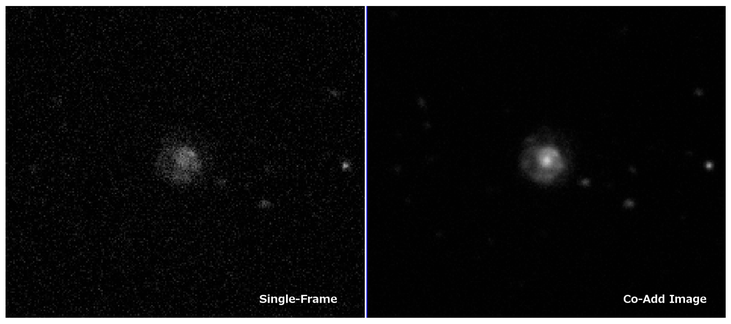

Figure 2: [Image: CFHT] *Left: faint and blurry Single Frame Observation, very low Signal-to-Noise; Right: Blurry 18*-frame Co-Add Image, higher Signal-to-Noise ratio, hard to identify objects

Problem

The general problem can be defined as shown in equation 1, where y_t and f_t are the observed image and the Point Spread Function defining the blur at time *t*. *x* is the underlying "true" image which we assume to be constant across all observed frames. The observed image *y* consists of the "true" image, *x,* which has been convolved with an unknown blur *f*. Both *x* and f_t are unknown but can be constrained to being at least non-negative. To recover the "true" image we need to estimate the PSF to be able to deconvolve it from the observed image.

In order to get a good estimate of the PSF, we extract information from each of a series of single frames and iteratively apply it to our "true" image estimate.

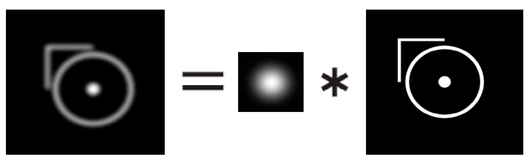

Figure 3: The blurry observed image on the *left* is a composition of the blur function or PSF(*center*) and the underlying "true" image on the *right*.

Methods & Implementation

Our tool currently features two separate iterative methods, a Gaussian-based Multiframe Blind Deconvolution (MFBD) as described by [1] and [2] and a Poisson-based Richardson-Lucy(RL) deconvolution[3].

$$y_t = f_t * x$$

Equation 1: *Where y is the observed image, f is the PSF and x is the true image*

Both approaches use an update formula, which is iteratively applied to every observed image y_t in an attempt to estimate it's PSF. This is done by holding the last iteration's estimate of x constant and applying the update formula repeatedly while solving for f_t. Once we have estimated the PSF we take a swing in reverse, this time holding the PSF constant and updating the x image. Currently we have both the Gaussian-based[2] and Richardson-Lucy-based[3] update formulas implemented, Equation 2 and Equation 3 respectively. We repeat this process for every image in the series of images, therefore incrementally improving our estimate of what the underlying "true" x image is.

$$x_{t+1} = x_t \odot F^T \frac{y}{Fx}$$

Equation 3: Richardson-Lucy based update formula as described by Fish et al

$$x_{t+1} = x_t \odot \frac{F^T y}{F^T F x}$$

Equation 2: Gaussian based update formula as described by Harmeling et al

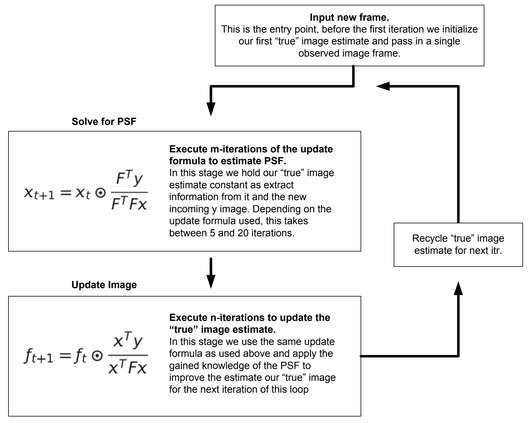

Figure 4: Multiframe Blind Deconvolution Iteration steps

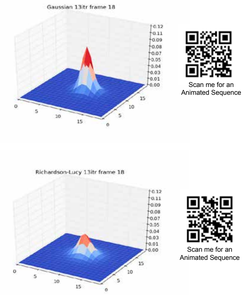

Figure 5: *top*: Progression of Gaussian PSF estimation *bottom:* Progression of Richardson-Lucy PSF estimation.

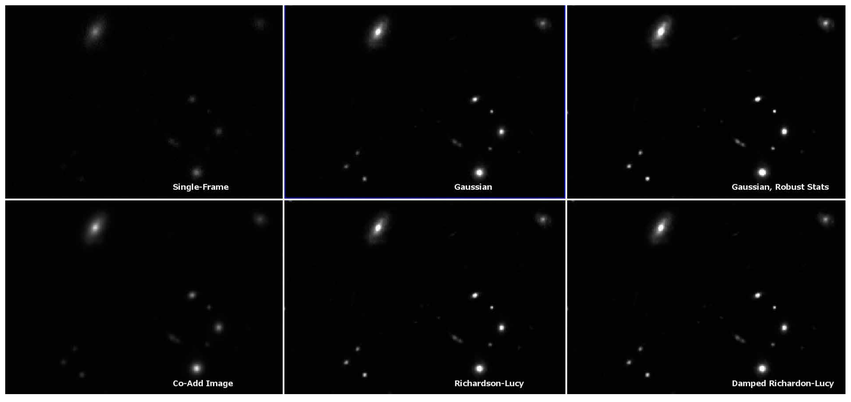

Figure 6: *Top: Left:* Single Frame Observation *Center:* Gaussian Approach *Right:* Gaussian with Robust Statistics Weighting; *Bottom: Left:* Co-Add Image *Center:* Richardson-Lucy Approach *Right:* Damped Richardson-Lucy (T=2σ, N=10) [Image: CFHT]

Besides the standard Gaussian and Richardson-Lucy methods, we also implemented improved methods and features such as wavelet filtering, robust statistics weighting and the "damped" Richardson-Lucy method[4], which weights the likelihood function to prevent over-fitting as it approaches convergence. With the combination of the above mentioned methods as well as GPU-acceleration we can rapidly test and evaluate a wide variety of different parameters and images.

This tool has been implemented in Python, relying heavily on pyCUDA for GPU acceleration. The grand vision for pyMFDB is to be a framework to test, compare and ultimately process distorted images using different approaches to Multiframe Blind Deconvolutions. Therefore we have emphasized modularity and easy CPU-GPU execution path switching.

Results

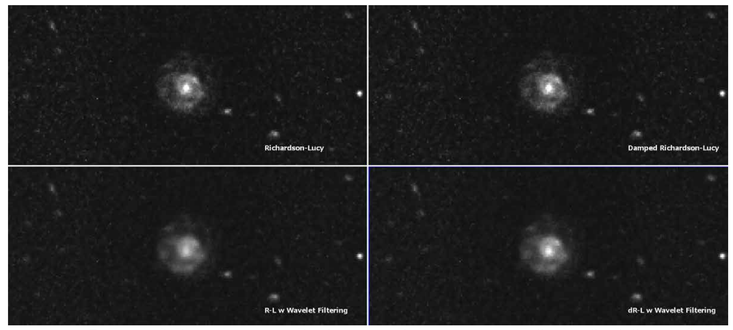

Figure 7: *Top: Left:* Richardson-Lucy; *Right:* damped Richardson-Lucy (T=2σ, N=10); *Bottom: Left*: Richardson-Lucy with Wavelet filtering; *Right*: damped Richardson-Lucy (T=2σ, N=10) with Wavelet filtering; [Image: CFHT]

At the moment the majority of the GPU-acceleration has been done by implementation of naïve element-wise kernels. This provides the majority of the speedup available from the GPU, but also keeps the path of execution simple and uncluttered. Once the code has reached a more mature state, we will consider optimizing beyond this.

Our GPU accelerated tool allows us to easily experiment with different algorithms and parameters. CPU-based processing of a set of 20 images can take upwards of an 60 minutes depending on the settings and algorithms used. The same task can now be accomplished in under 3 minutes. That is an over **20x speed up**. Since parameter adjustments are vital to suit diverse sets of images, it is essential to be able to experiment with image results in near real time.

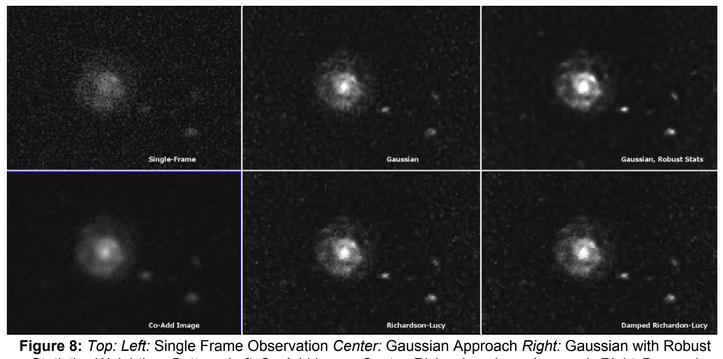

Figure 8: *Top: Left:* Single Frame Observation *Center:* Gaussian Approach *Right:* Gaussian with Robust Statistics Weighting; *Bottom: Left:* Co-Add Image *Center:* Richardson-Lucy Approach *Right:* Damped Richardson-Lucy (T=2σ, N=10) [Image: CFHT]

In the future we aim to add support for super resolution, which will likely yield a better reconstruction. While implementing wavelet filtering option, we found no viable python-based GPU-accelerated wavelet library. Currently, even when running in GPU mode, we compute the wavelet transform on the CPU which causes a major slowdown. To remedy this issue we are working on a python and pyCUDA wrapper for gpudwt. Our tool is also Open Source under GPLv3 and available at:
http://github.com/madmaze/pymfbd

References

[1] Stefan Harmeling, Michael Hirsch, Suvrit Sra, and Bernhard Scholkopf. Online blind image deconvolution for astronomy. 2009.

[2] Stefan Harmeling, Suvrit Sra, Michael Hirsch, and B Scholkopf. Multiframe blind deconvolution, super-resolution, and saturation correction via incremental em. In Image Processing (ICIP), 2010 17th IEEE International Conference on, pages 3313-3316. IEEE, 2010.

[3] Fish, D. A., Brinicombe, A. M., Pike, E. R., & Walker, J. G. (1995). Blind deconvolution by means of the Richardson–Lucy algorithm. JOSA A, 12(1), 58-65.

[4] White, R. L. (1994, June). Image restoration using the damped Richardson-Lucy method. In 1994 Symposium on Astronomical Telescopes & Instrumentation for the 21st Century (pp. 1342-1348). International Society for Optics and Photonics.

Contact us:

Matthias A Lee
MatthiasLee@jhu.edu
Department of Computer Science
The Johns Hopkins University
3400 N Charles Street
Baltimore, MD 21238

Tamás Budavári
Budavari@jhu.edu
Department of Astronomy and Physics
The Johns Hopkins University
3400 N Charles Street
Baltimore, MD 21238