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 through the ever-changing and blur-inducing atmosphere. This can be especially detrimental for faint and distant observations. The point spread function (PSF) defining this blur is unknown, hard to predict and varies wildly making recovery more difficult.

Modern telescopes produce high volumes of extremely large images, upwards of 1000GB in 10 years, yielding an even more compute-intensive time-consuming reconstruction. The current “State-of-the-Art” images are commonly Co-Add images made by overlaying multiple faint objects, which are very noisy and missing detail. More advanced reconstruction is computationally 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.

Problem

The general problem can be defined as shown in Eq. (1), where \( v_i \) and \( f \) 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. In case there are moving objects, these can be masked. The observed image \( v_i \) consists of the “true” image, \( x \), which has been convolved with an unknown blur \( f \) and some noise \( e_i \). Initially both \( f \) and \( e_i \) are unknown but can be constrained to being at least non-negative, which allows us to use methods for Non-Negative Matrix Factorization.

To recover the “true” image we need to deconvolve the PSF and the observed image. The better our estimate for the PSF is, the more precise our model of the underlying “true” image will be. In order to get a good estimate of the PSF, we extract information from each of a series of single frames and iteratively update it to our model.

Methods & Implementation

Our approach mainly focuses on the Gaussian-based Multiframe Blind Deconvolution (MBD) as described by Stefan Harmeling et al. [1]. We have also experimented with the Poisson-likelihood based Richardson-Lucy deconvolution as described by Rick White [3].

\[
y_t = f_t * x + e_t
\]

Equation 1: Where \( y_t \) is the observed image, \( f_t \) is the PSF and \( x \) is the true image

\[
f_t = \arg \min_{f \geq 0} ||y_t - F x_t||^2
\]

Equation 2: Function to be minimized for every observed image

\[
x_{t+1} = x_t + C \left[ \frac{F^T W y_t}{F^T W F x_t} \right]
\]

Equation 3: Gaussian-based update formula as described by Harmeling et al. [1]

Equation 4: Enhanced update formula, adding Clipping and Weighting, see [4]

To estimate our PSF we need find the \( f \) which minimizes the residual between the observed image and our model image, see Eq. (2). With this PSF we update the model image via our multiplicative update formula, see Eq. (3). This process is repeated for every observed image. The model is thereby iteratively updated with new information from every image.

The simple update formula as described by Stefan Harmeling et al. in [1] and [2] work well on clean images void of defects and background subtraction issues. In order to combat issues with real data, we added two main enhancements to the update formula, see Eq. (4). We introduce a clipping parameter as well as a weighting matrix. The clipping, \( C \), limits the effect a single update may have on our model and thereby prevents bad images or failed PSF estimates from breaking our algorithm. The weighting matrix, \( W \), lets us introduce both masks as well as robust statistics weighting, which prevents bright sources from dominating the residual and therefore converging before the rest of the image has converged.

![Figure 1](image1.png)
![Figure 2](image2.png)
![Figure 3](image3.png)
![Figure 4](image4.png)

Results & Future Work

Our tool is built with python and relies on pyCUDA for GPU-acceleration. We achieved an over 40x speedup over the previous CPU version, currently it takes approximately 5 minutes to process 70 images. Depending on parameter adjustments and input images, the run time will vary as the number of iterations will vary to reach convergence.

Currently we are working on super resolution which promises higher shape and location precision of sources. Another upcoming feature is multi-parameter background estimation. This adds only a few more parameters to solve for, but will allow us to automatically detect and remove gradient backgrounds. Beyond these algorithmic improvements, there are also some regions of the code that could be vastly sped up with more GPU optimizations.

References


Contact us:

Matthias A Lee
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218

Tamás Budavári
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218