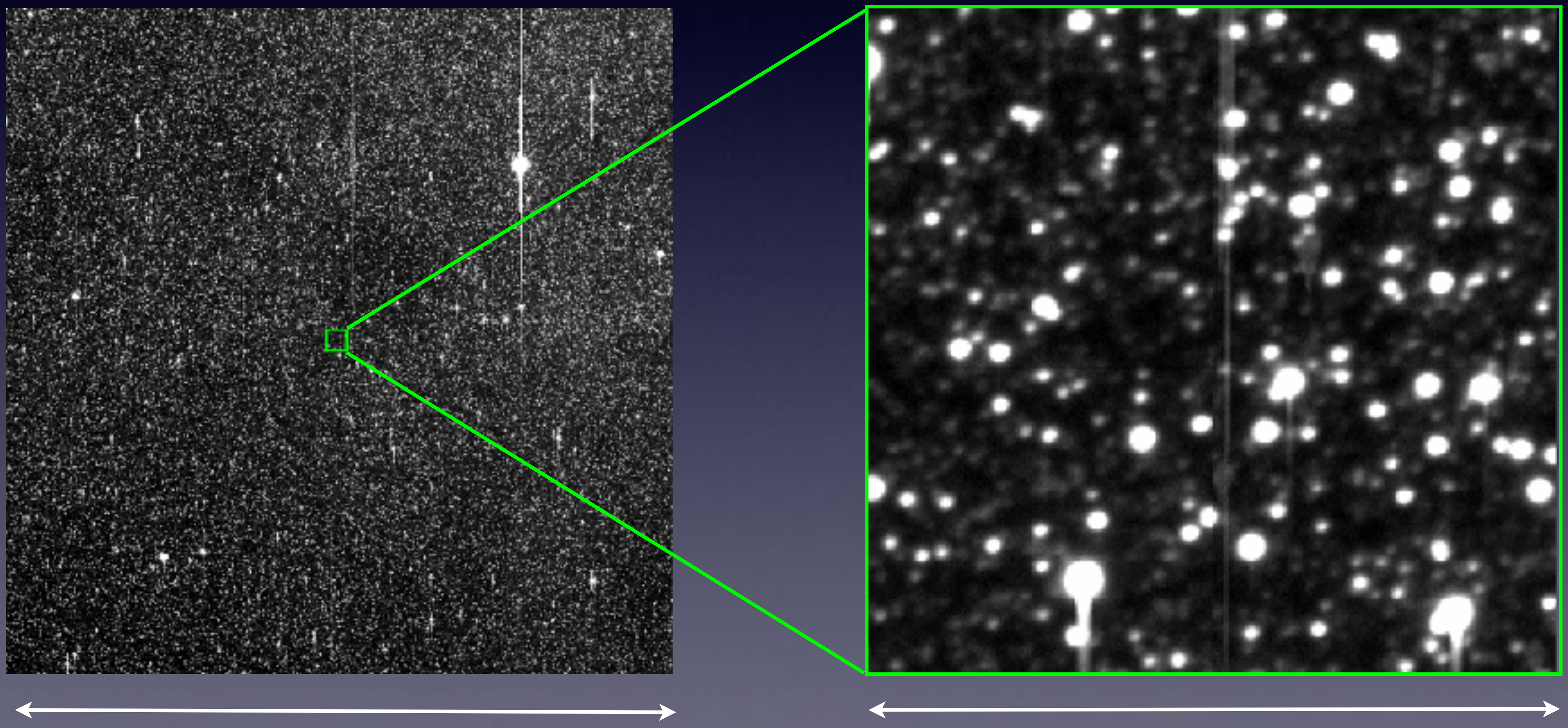
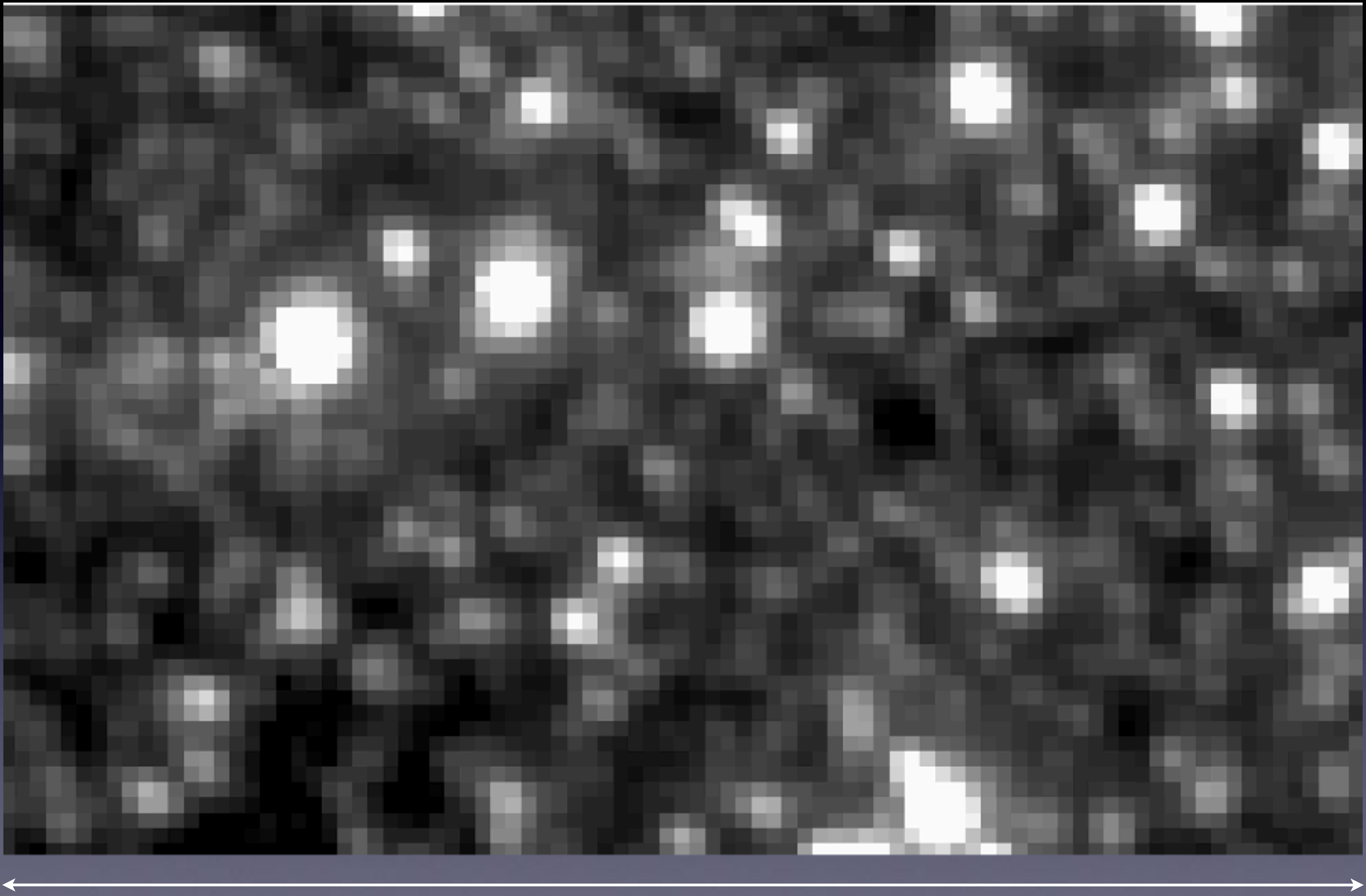
Difference-imaging photometry with pyDIA

Michael Albrow
University of Canterbury

Crowding




35 arcsec

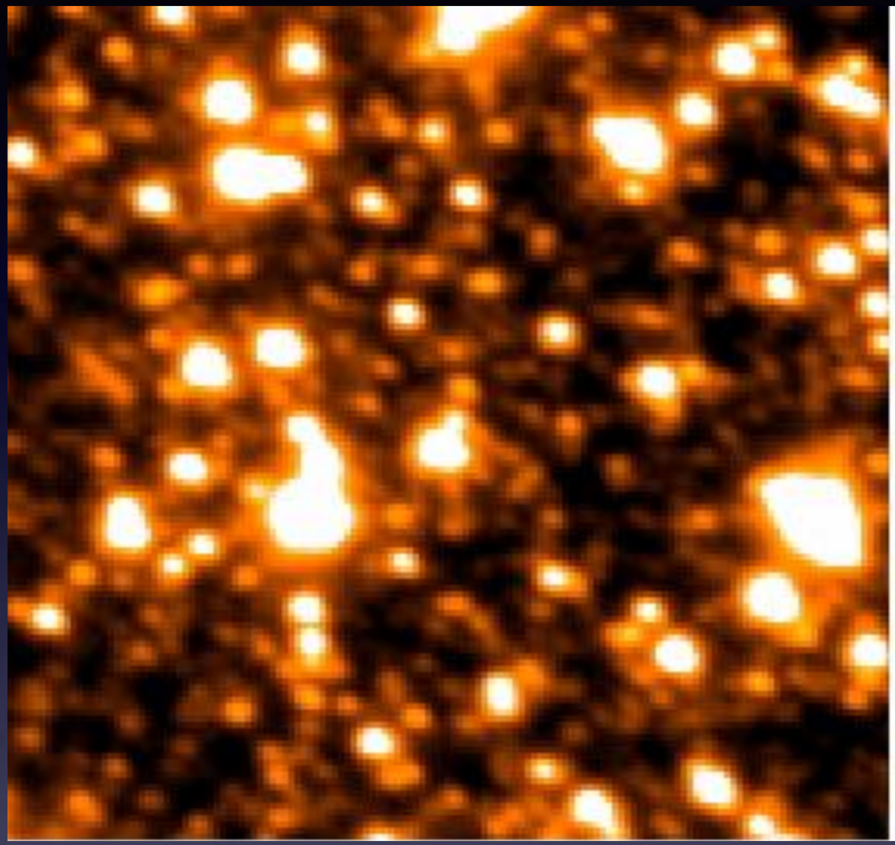
Difference imaging

If we have a reference image, R, and a series of target images, T^α, then we define the difference image

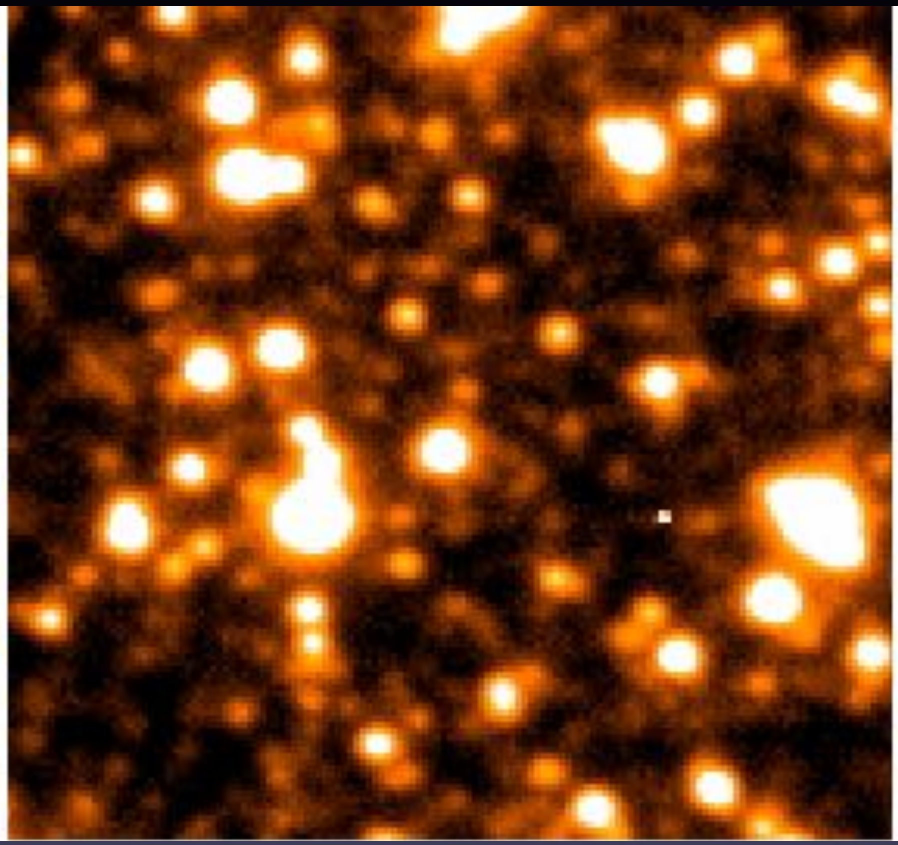
$$D^\alpha = T^\alpha - R \otimes K^\alpha$$

K^α is a convolution kernel computed to minimize

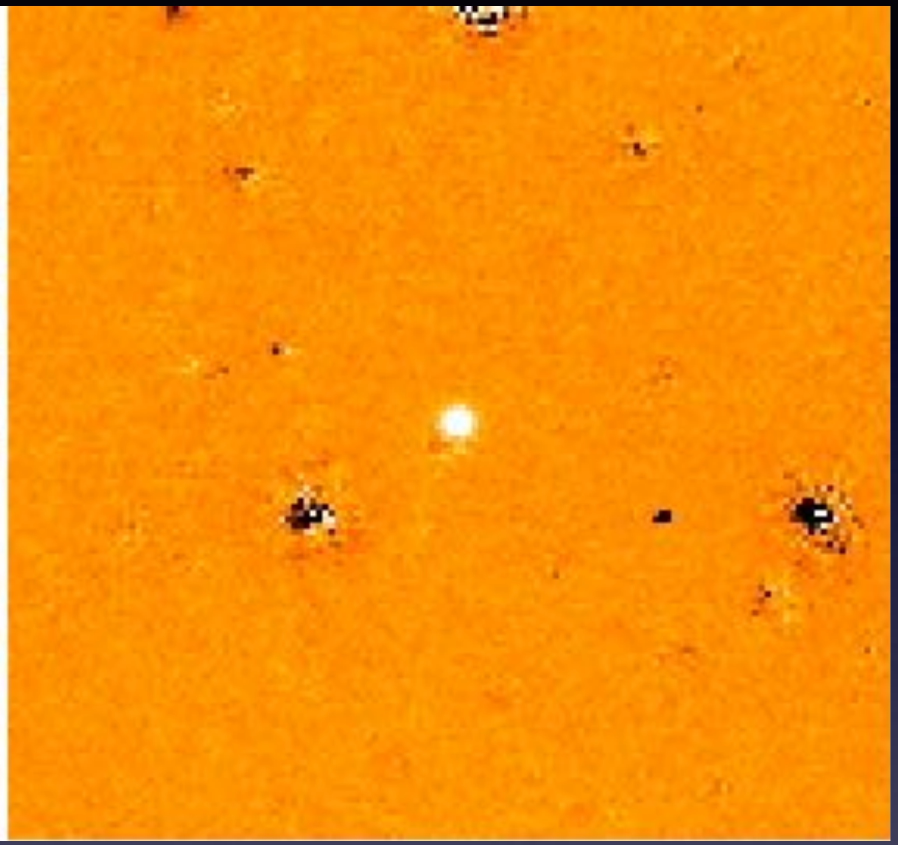
$$\chi^2 = \sum_{ij} \left(\frac{D_{ij}}{\sigma_{ij}} \right)^2$$



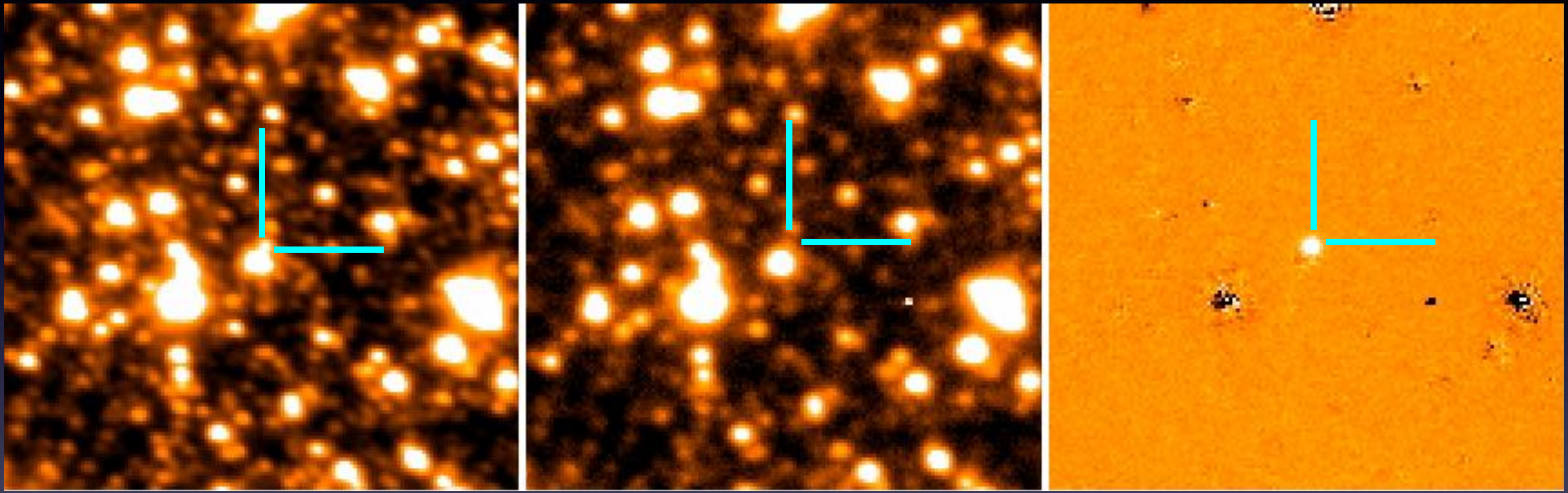
R



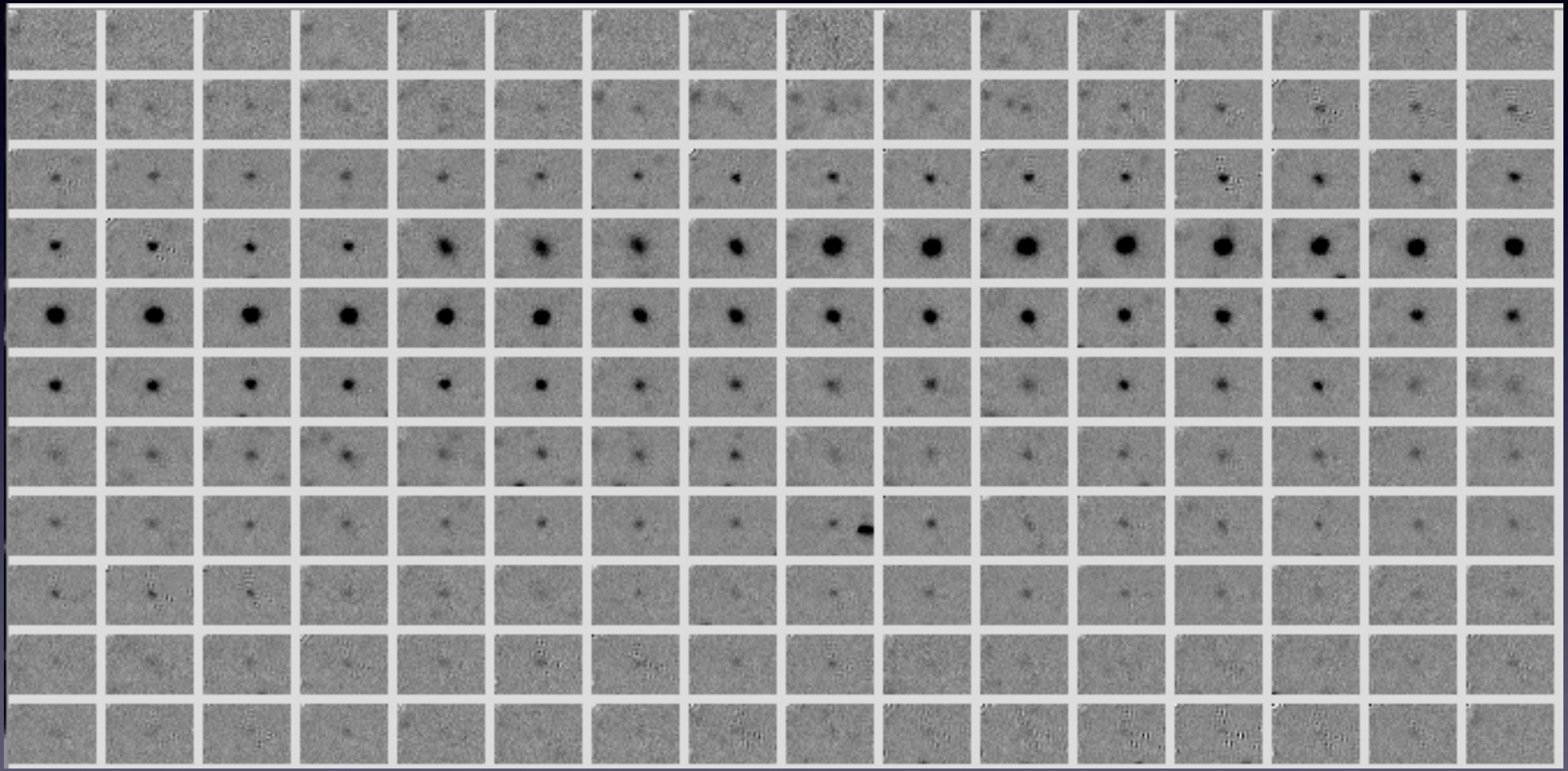
T



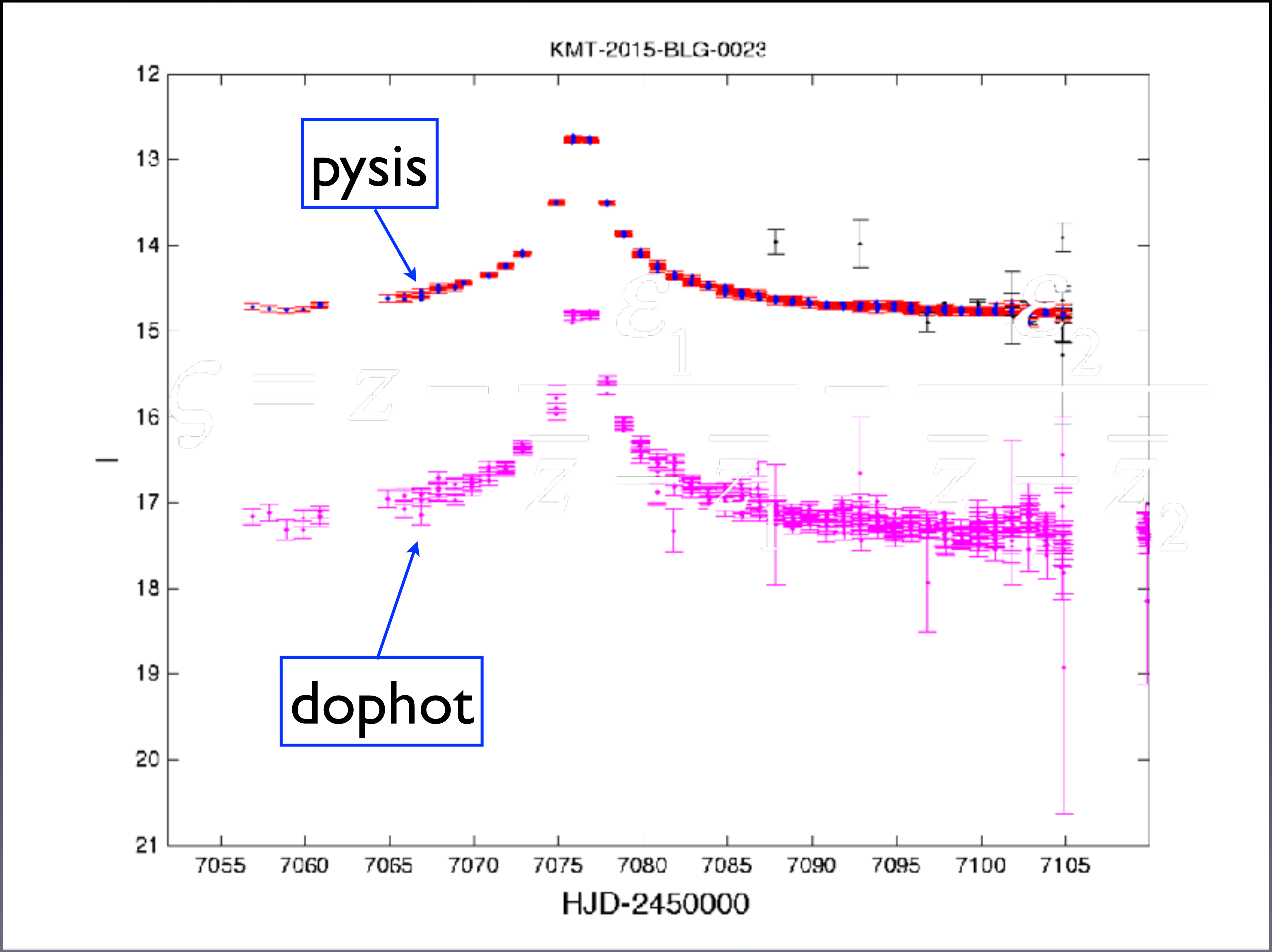
D


R
T
D

Example: KMT-2015-BLG-023



Time-series of difference images


KMT-2015-BLG-0023
I
HJD-2450000
pysis
dophot

pySIS ➡ pyDIA

pySIS: single star, python + C, CPU

pyDIA: all stars, python + (CUDA) C, CPU or GPU

pyDIA philosophy

Don't reinvent the wheel.

Use standard python packages if possible for higher level tasks. Make things easy to customize.

Use C/CUDA only for the lowest-level routines.

Make it all open source.

pyDIA capabilities

Alignment

Masking

Reference image construction

Star detection

Reference photometry

Difference imaging

pyDIA capabilities

Difference image photometry

Variable star detection

Coordinate refinement

Calibration, CMDs, etc

Alignment

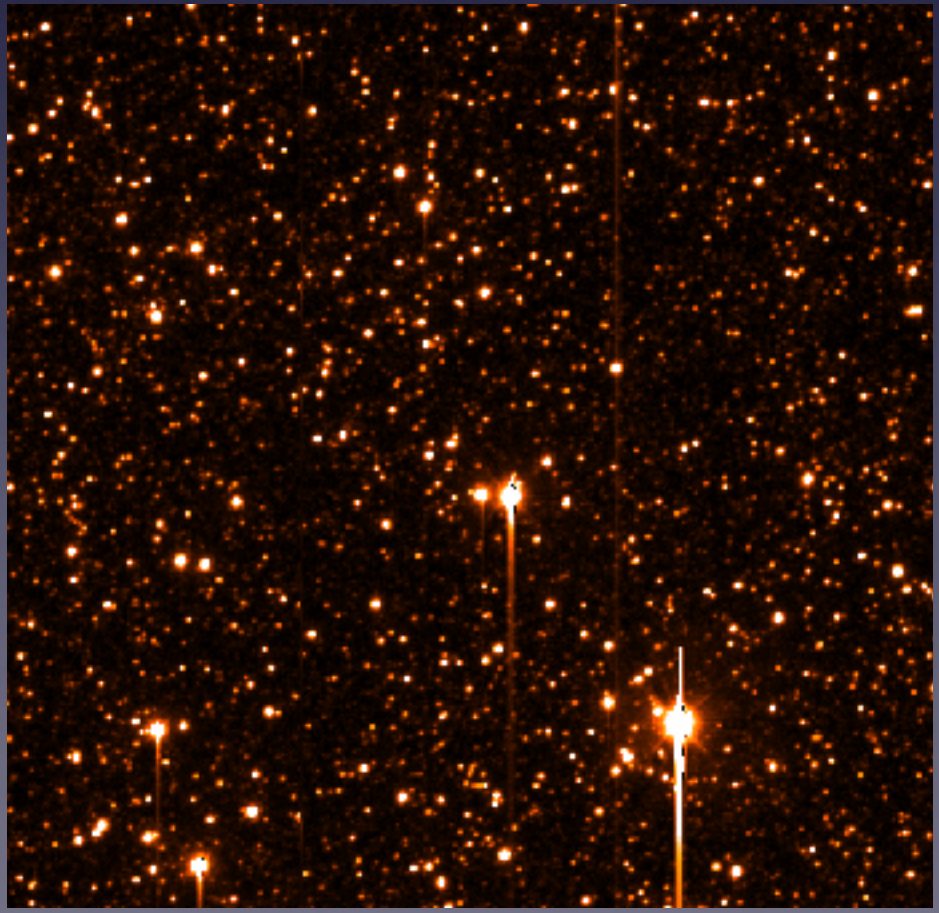
Offset determined by cross-correlation.

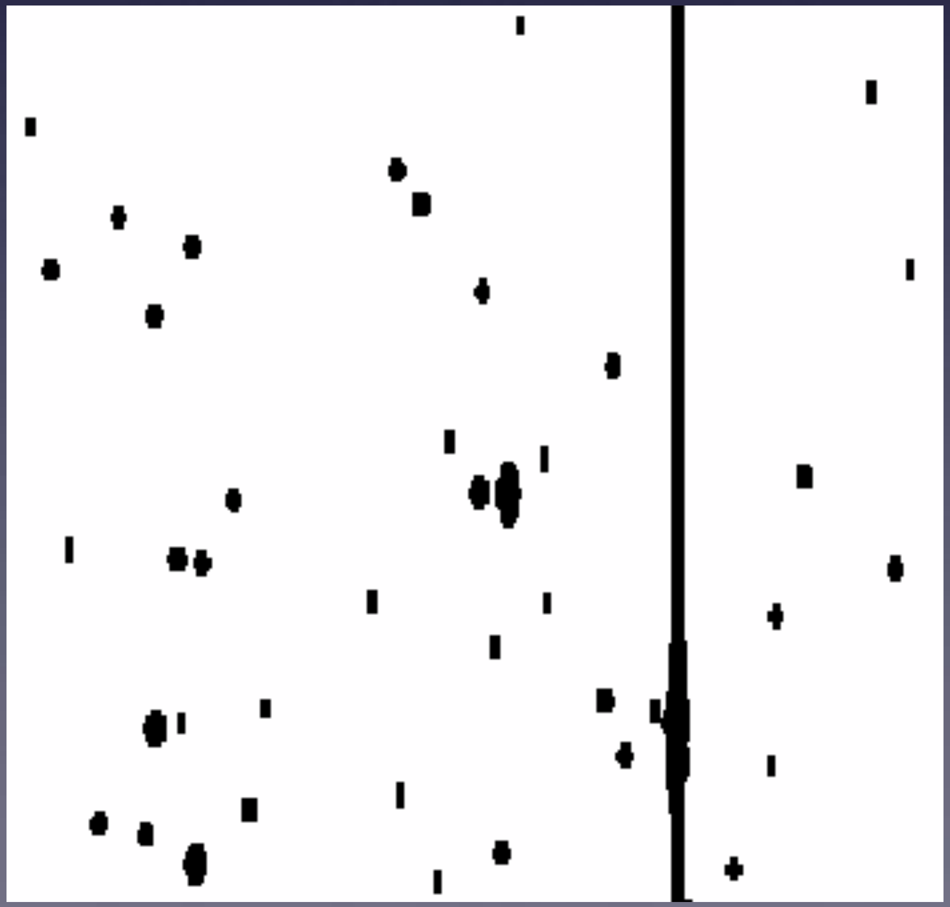
Integer pixel translation, preserves pixel independence and noise characteristics.

Masking

Circular areas around saturated stars.

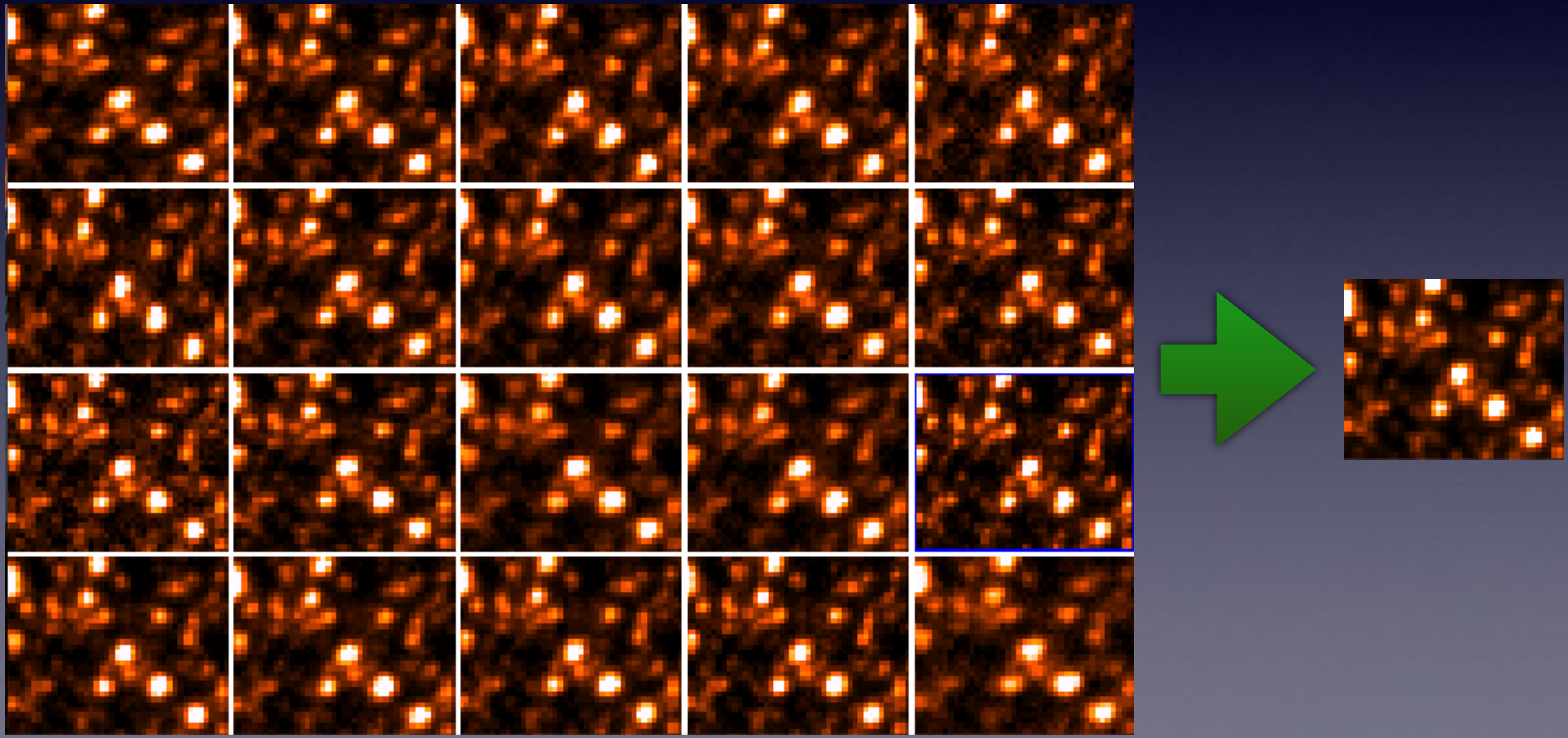
Bleeding trails detected with an extended Prewitt filter.





Reference image

Automated or manual selection of images to stack. Generally based on images with smallest FWHM and lowest background.



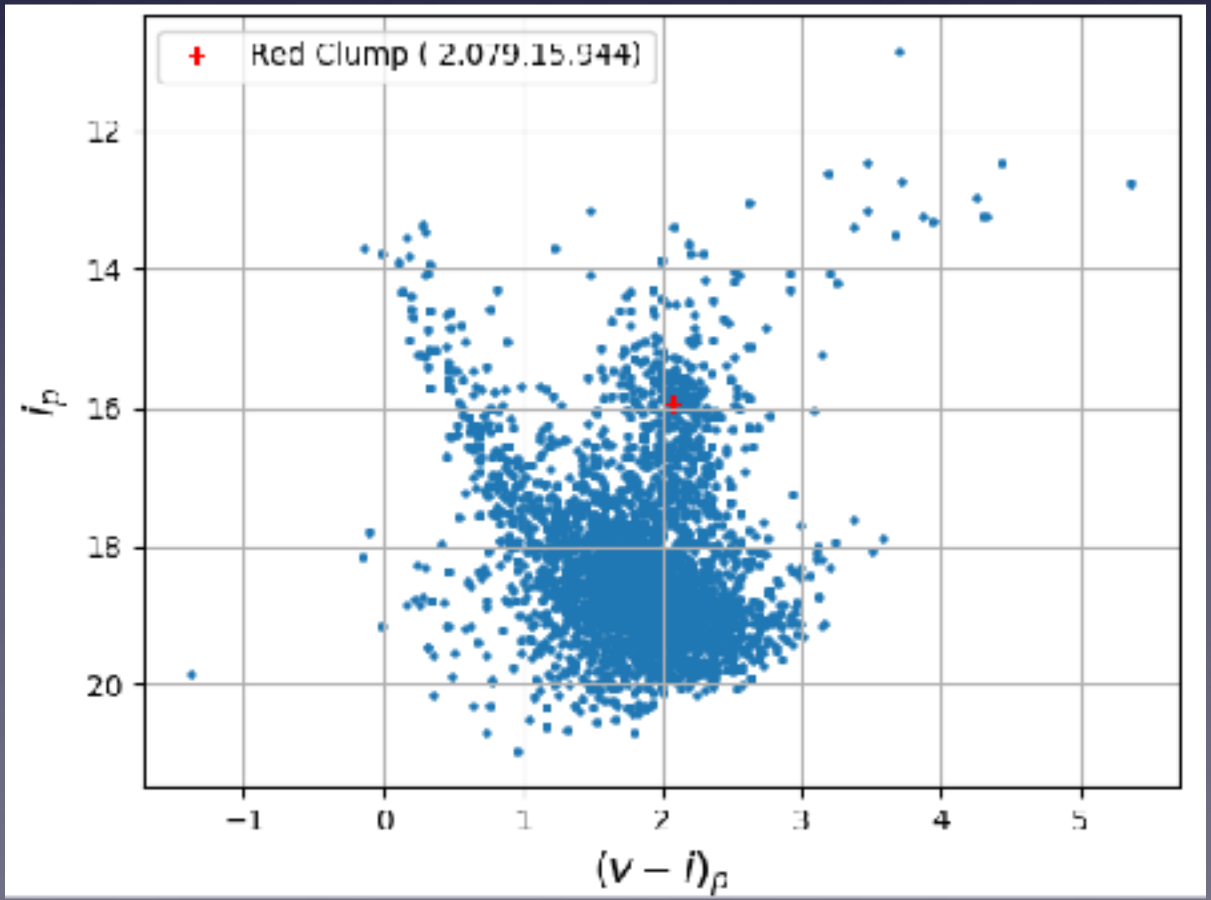
Object catalogue

Uses daofind from (pyraf) DAOphot package.

Option of using existing catalogue. Code will align catalogue to reference image.

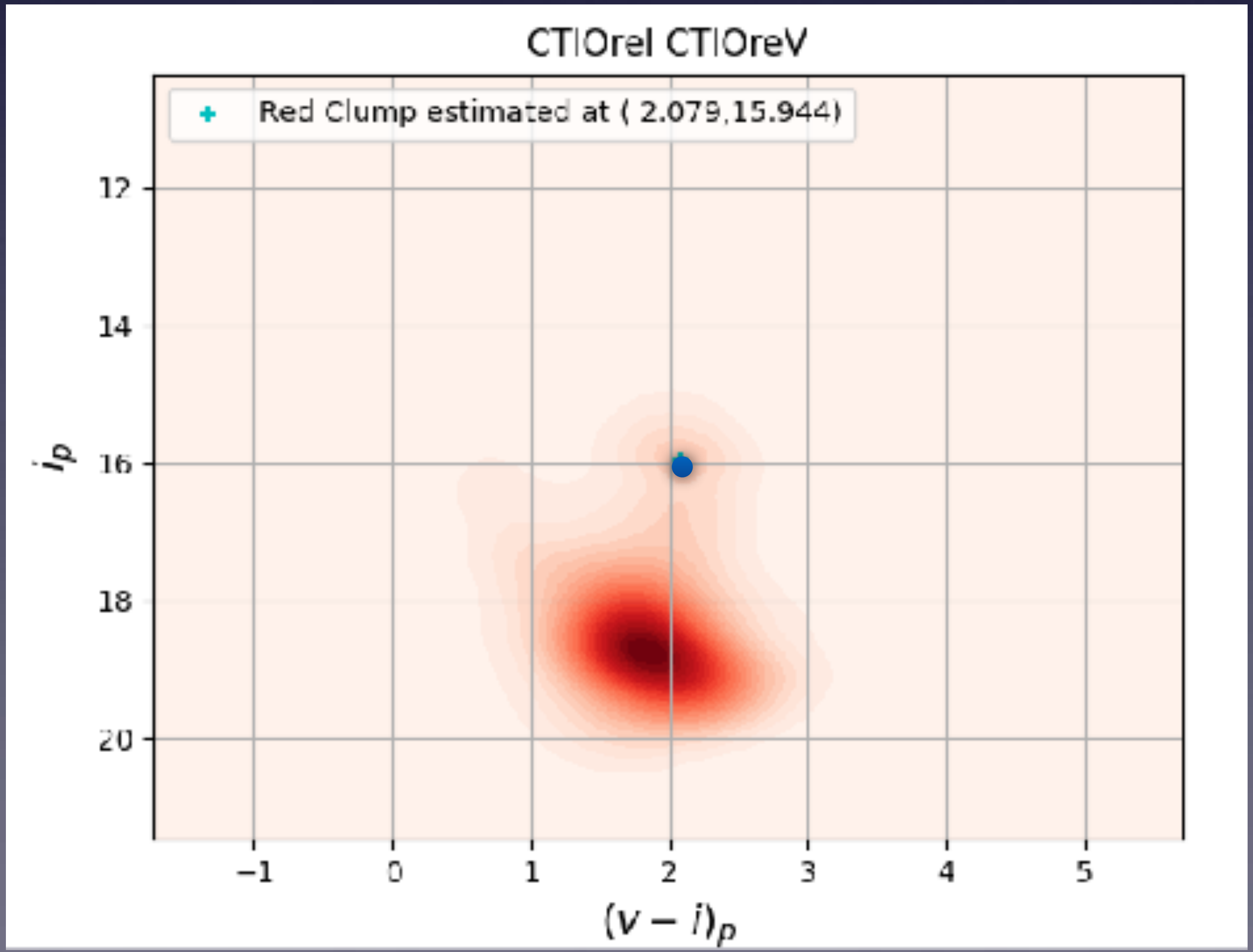
Reference image photometry

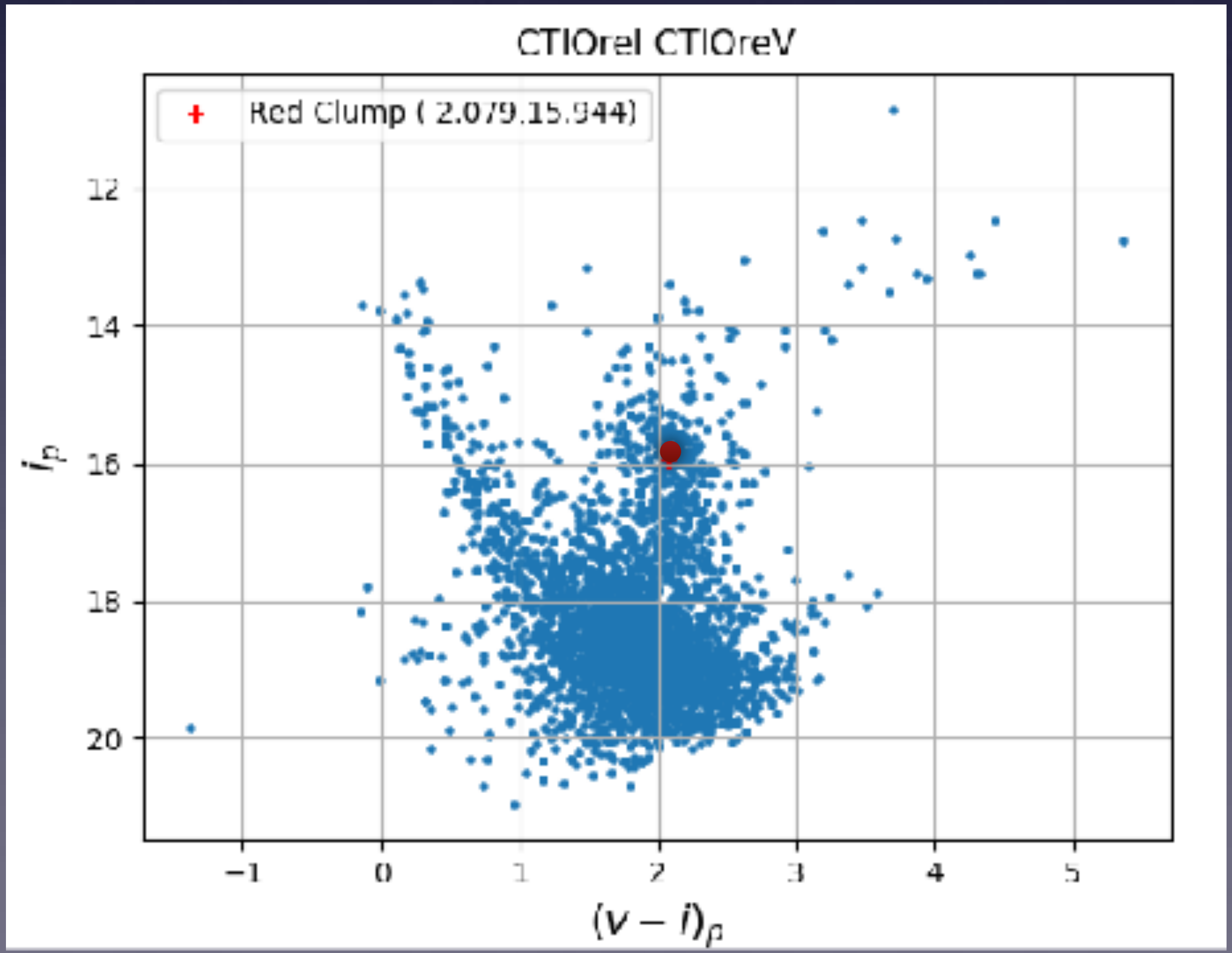
Uses routines from the DAOphot package (in python) to compute PSF and do photometry of the reference image (psf/allstar).



Reference image photometry

Automated kernel density estimation of red clump centroid.





Difference imaging

Uses extended modified-delta-basis functions for kernel representation.

User can set independent spatial degrees for the kernel intensity, kernel shape and background.

Difference imaging

If we have a reference image, R, and a series of target images, T$^\alpha$, then we define the difference image

$$D^\alpha = T^\alpha - R \otimes K^\alpha$$

K$^\alpha$ is a convolution kernel computed to minimize

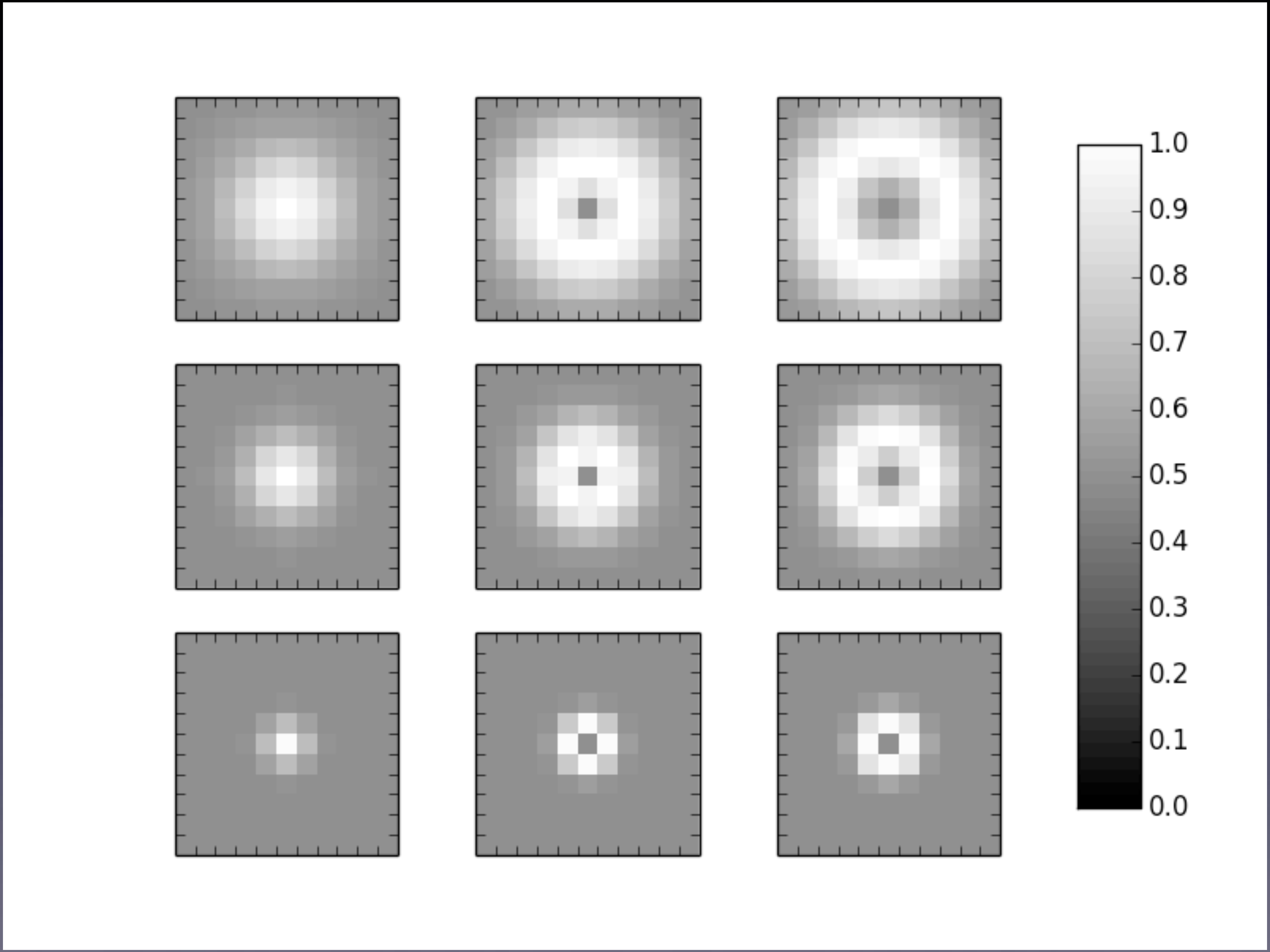
$$\chi^2 = \sum_{ij} \left(\frac{D_{ij}}{\sigma_{ij}} \right)^2$$

Kernel representation

For a constant kernel, K can be decomposed into a linear sum of pre-defined basis functions

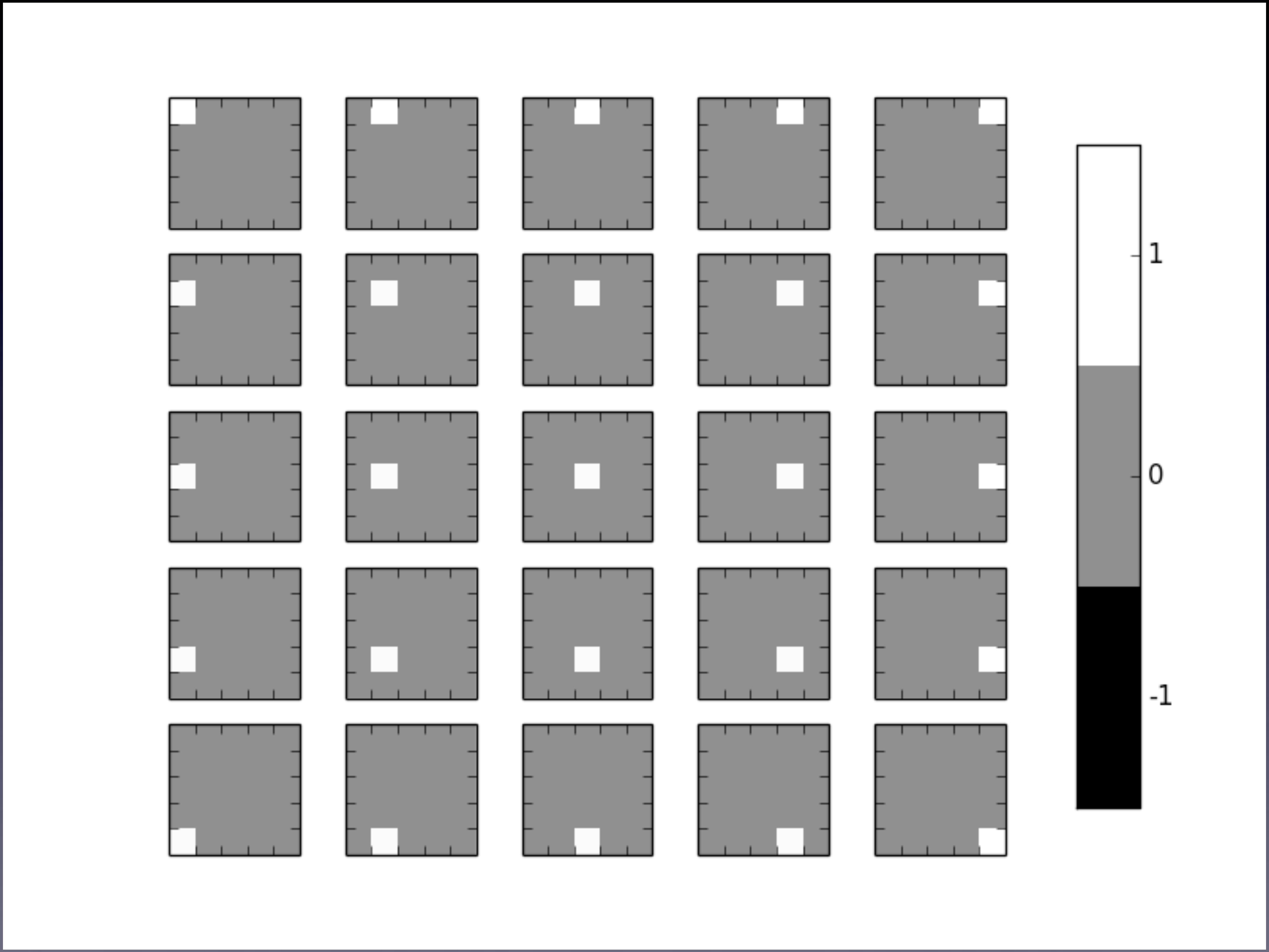
$$K(u,v) = \sum_k c_k b_k(u,v)$$

Gaussian-polynomial basis functions



Alard & Lupton (1998)

Delta basis functions



Bramich (2008)

Spatial dependence

If the mapping from reference to target image has a spatial dependence, the kernel representation becomes

$$K(u,v,x,y) = \sum_k \sum_i \sum_j c_{kij} b_k(u,v) x^i y^j$$

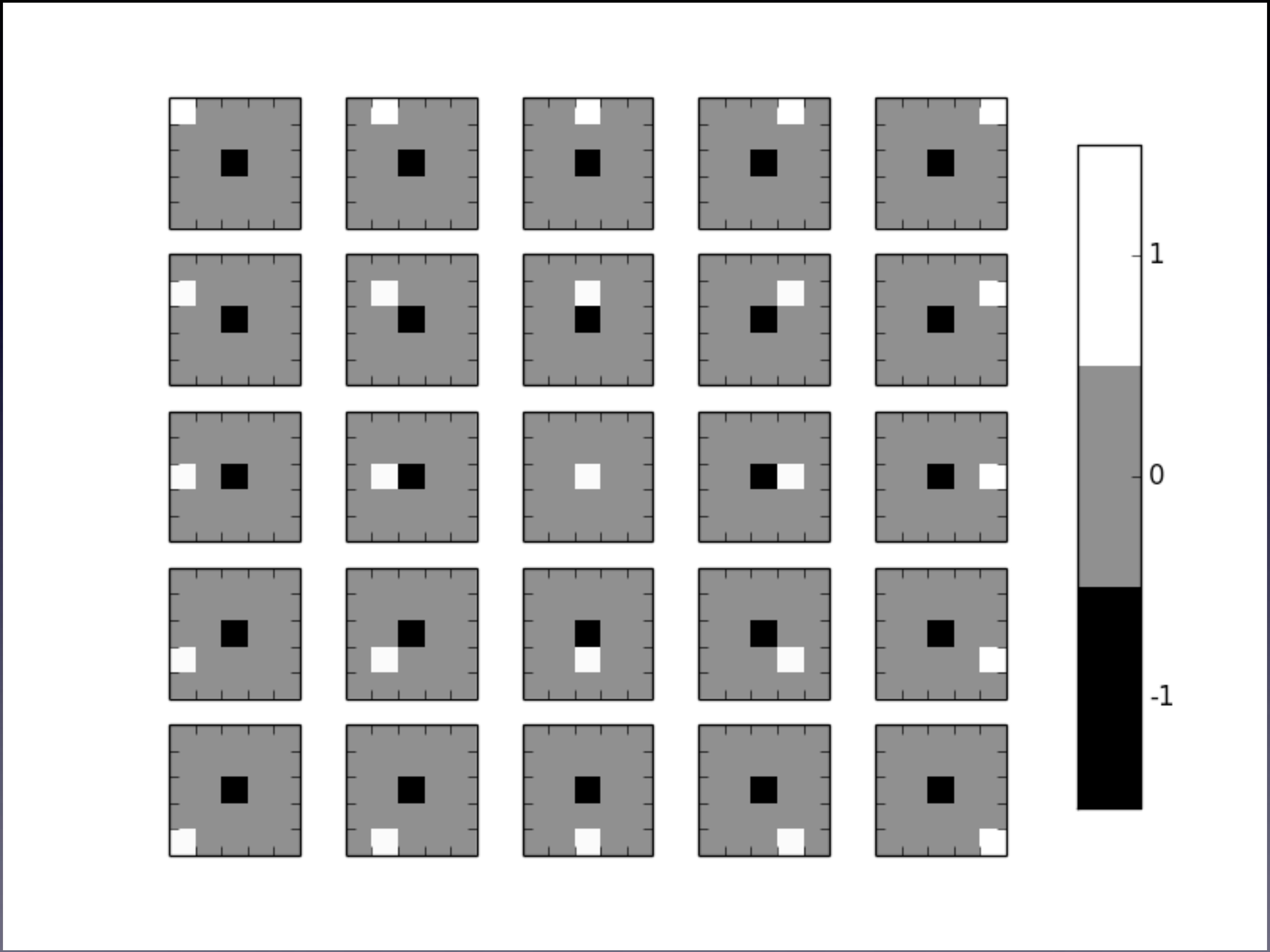
Photometric scale

The kernel sum

$$S = \sum_{uv} K_{uv}$$

gives the photometric scale for the mapping.

Modified delta basis functions



Bramich, Horne, Albrow et al. (2013)

Modified delta basis functions

Since all the kernel basis functions except one have a zero sum, the photometric scaling is controlled by the coefficients corresponding to a single kernel element. The remaining coefficients only describe shape-changes.

This decomposition allows us to set the spatial degree of the photometric scaling separately from the spatial degree of the shape changes.

Difference imaging

Kernel evaluated in CUDA (GPU) or C (CPU).

Can use all image pixels or small stamps.

Can iteratively reject residuals from kernel calculation.

Computing the kernel coefficients

Computing the kernel involves filling and inverting a large matrix, where the matrix elements are sums of products of the reference and target image pixels. This is computationally intensive.

GPU’s for computation

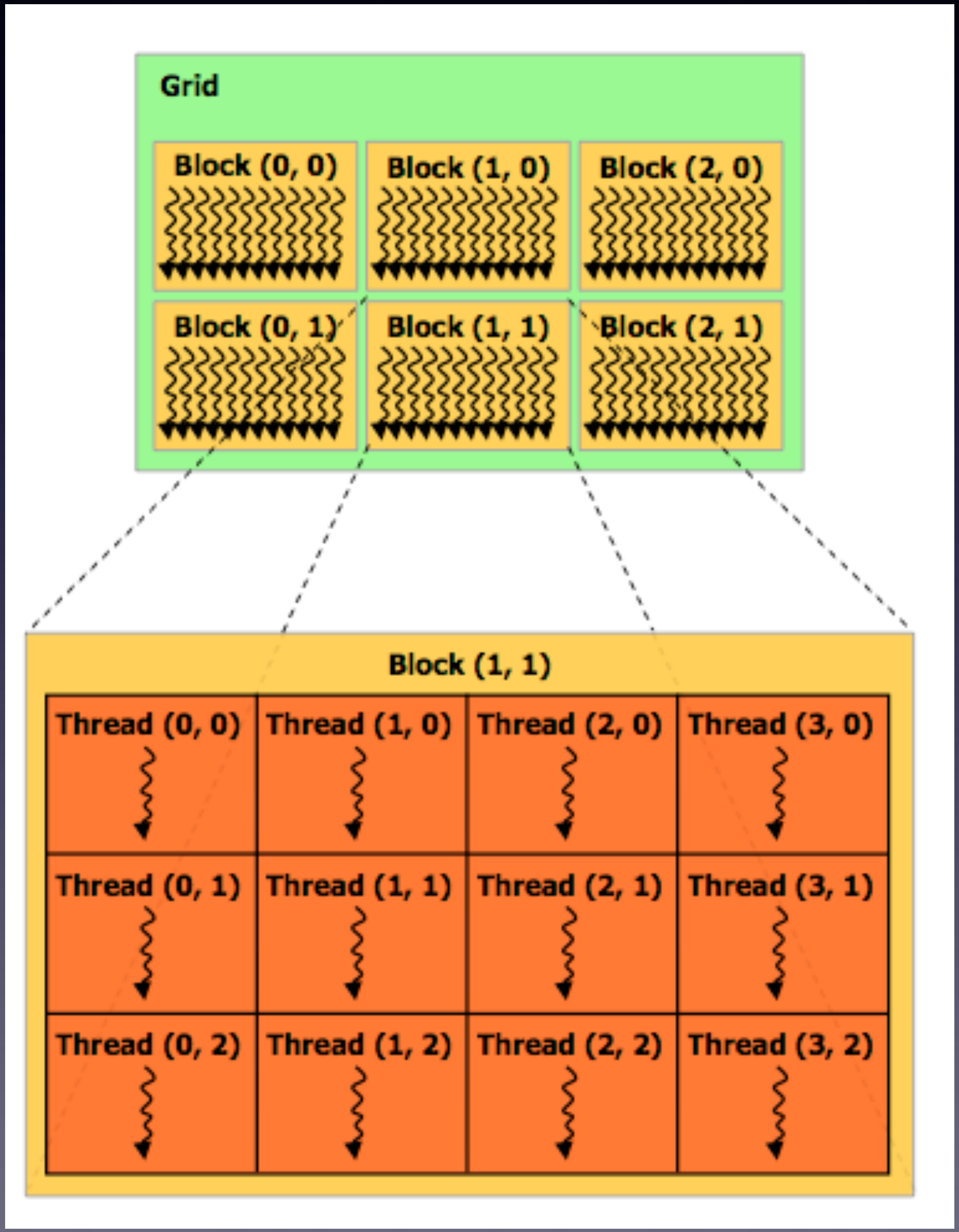
GP100
3584 cores

5.3 TFLOPS
Double Precision

GPU solution

We can use a GPU, with thousands of processing cores, to compute the matrix elements in parallel.

Each matrix element is computed by a CUDA block, threads are used to multiply and add image pixels.



NVIDIA CUDA

Difference-Image Photometry

Use reference-image PSF computed by DAOphot (gaussian + LUT).

Convolve PSF with kernel.

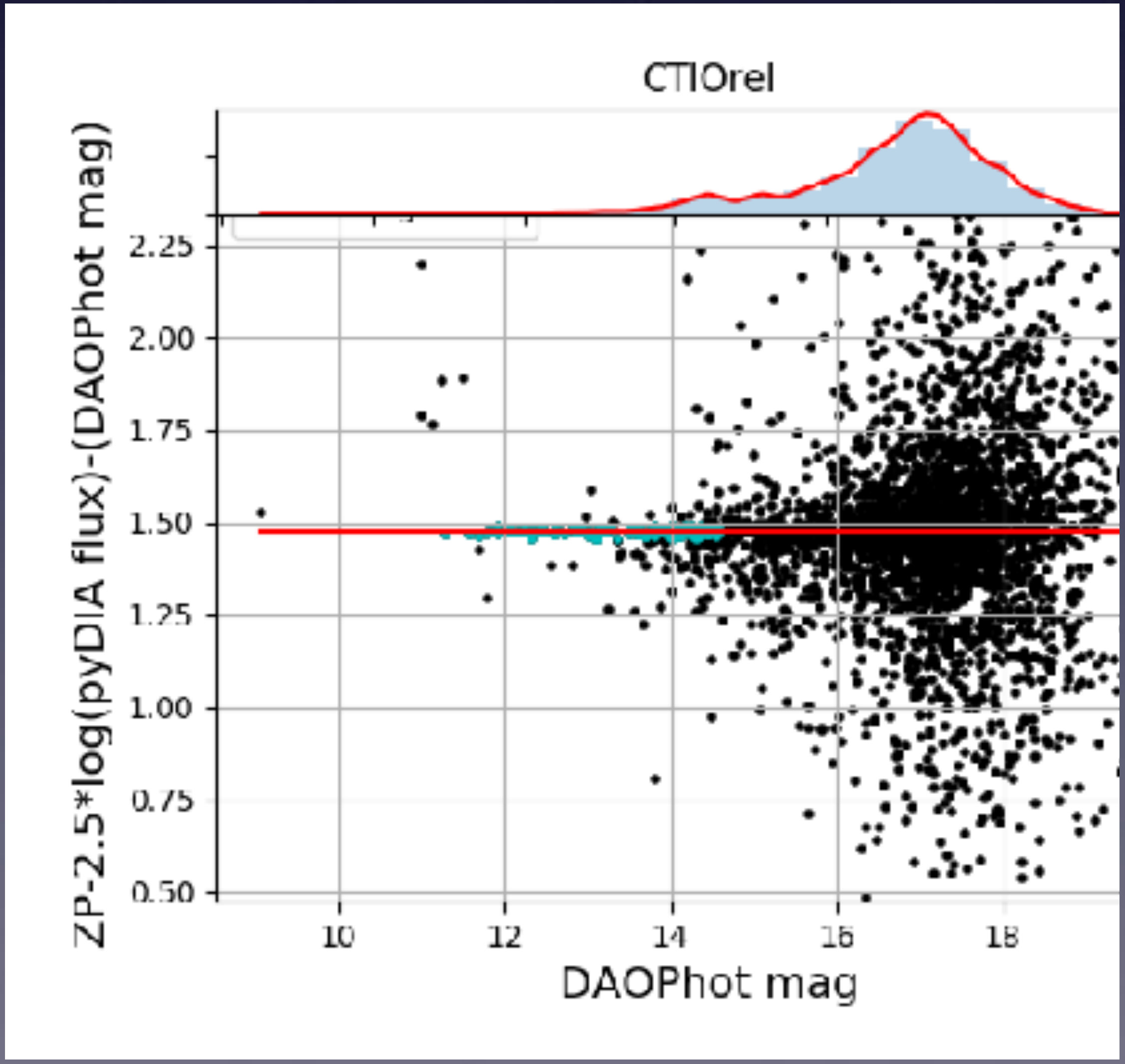
For each star in catalogue:

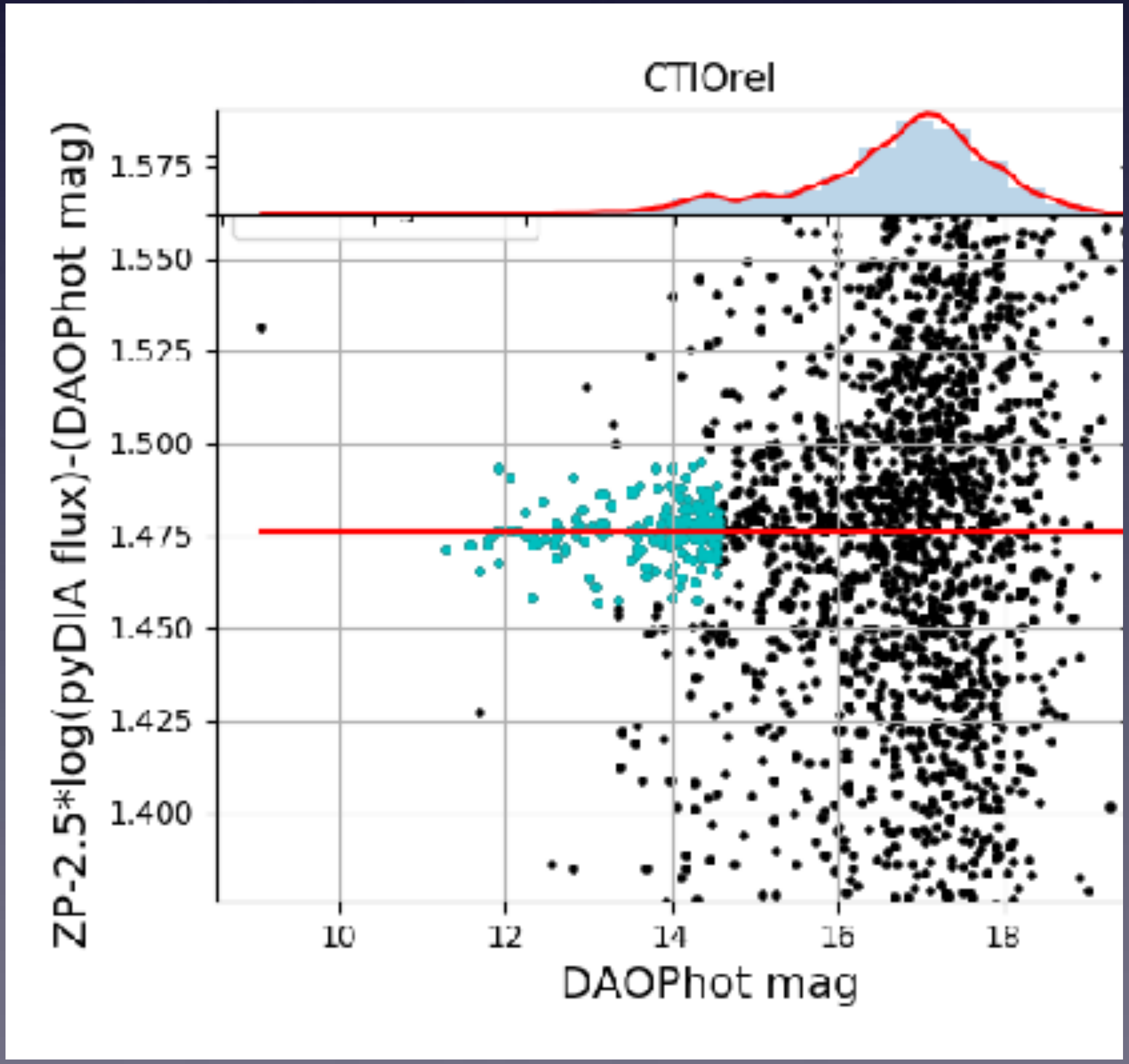
- evaluate PSF at star pixels
- evaluate flux by fitting PSF to difference image

Photometry

- CUDA photometry code
  - One CUDA block of 16 x 16 threads per star.
  - Each thread corresponds to an image/PSF pixel.

Calibration of lightcurves to reference photometry scale

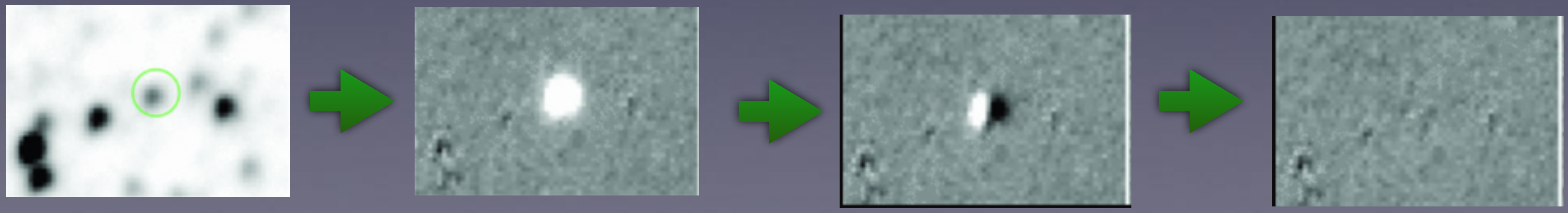




Blended-target photometry

Refined photometry for individual targets can be performed. This uses residuals in the difference images (after subtraction of the stellar PSF) to refine the coordinates.

The algorithm used is the same as in pySIS - see Albrow et al. (2009), MNRAS, 397, 2099



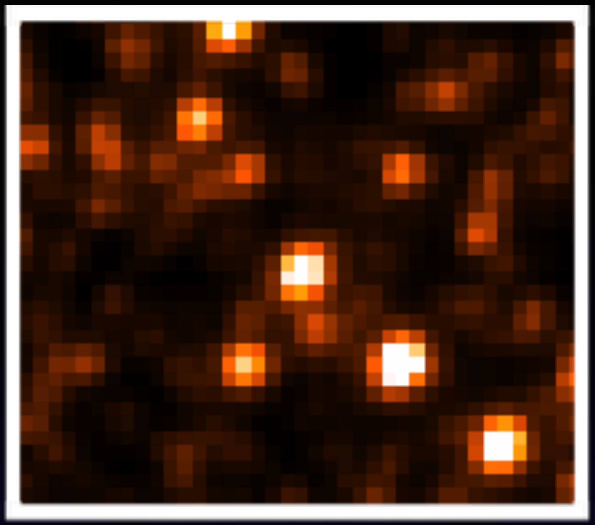
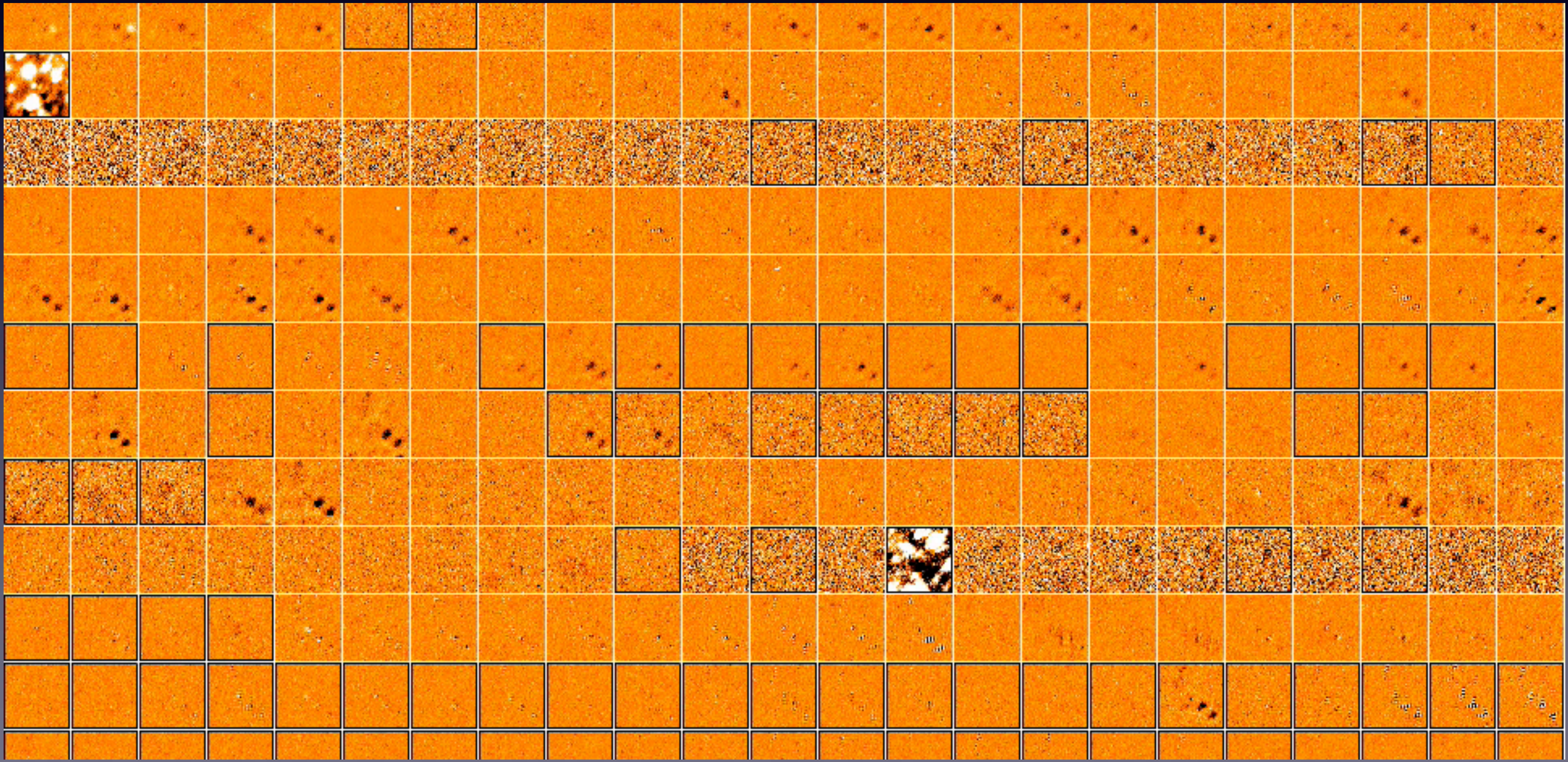


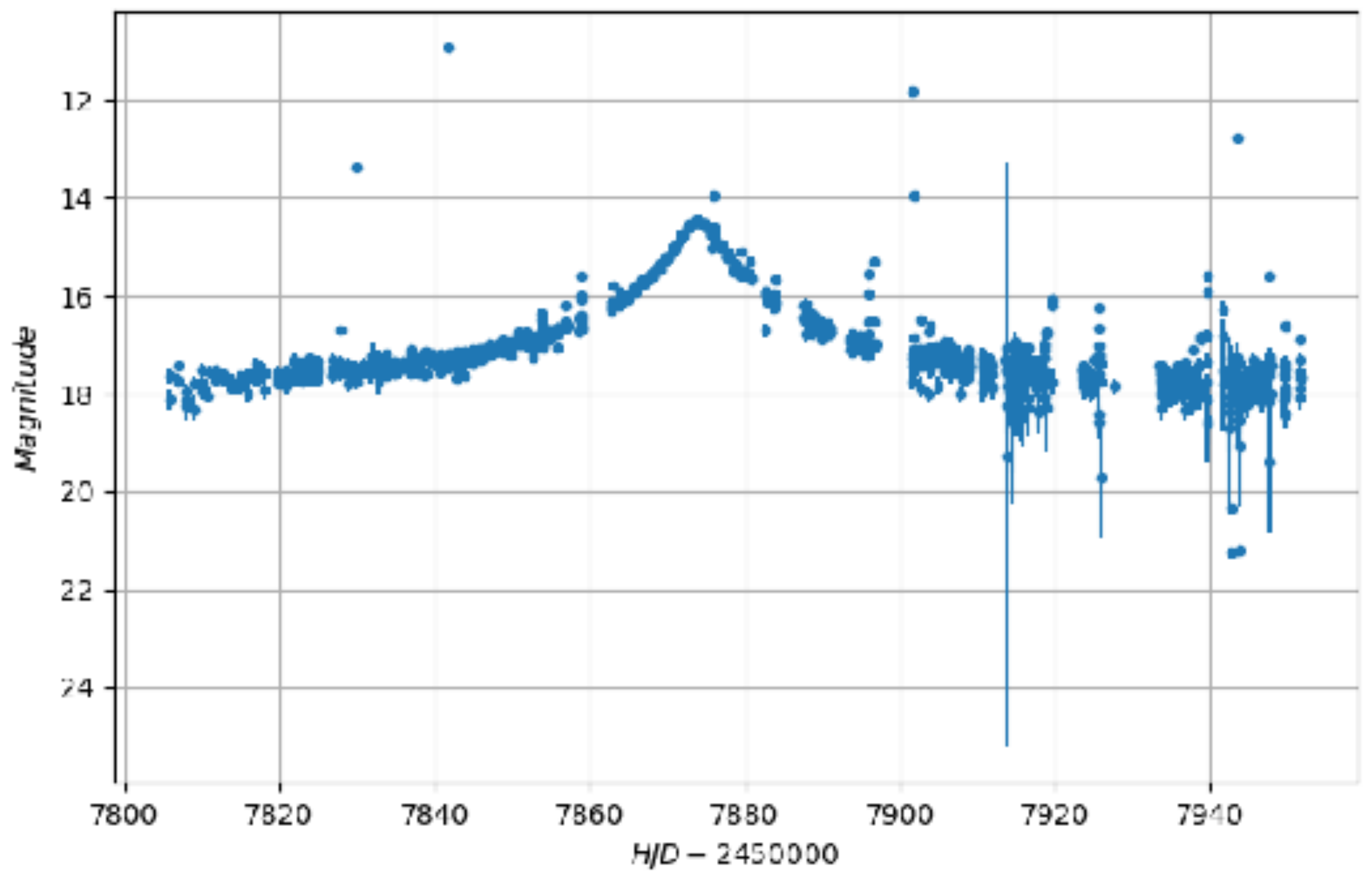
Image-level checks



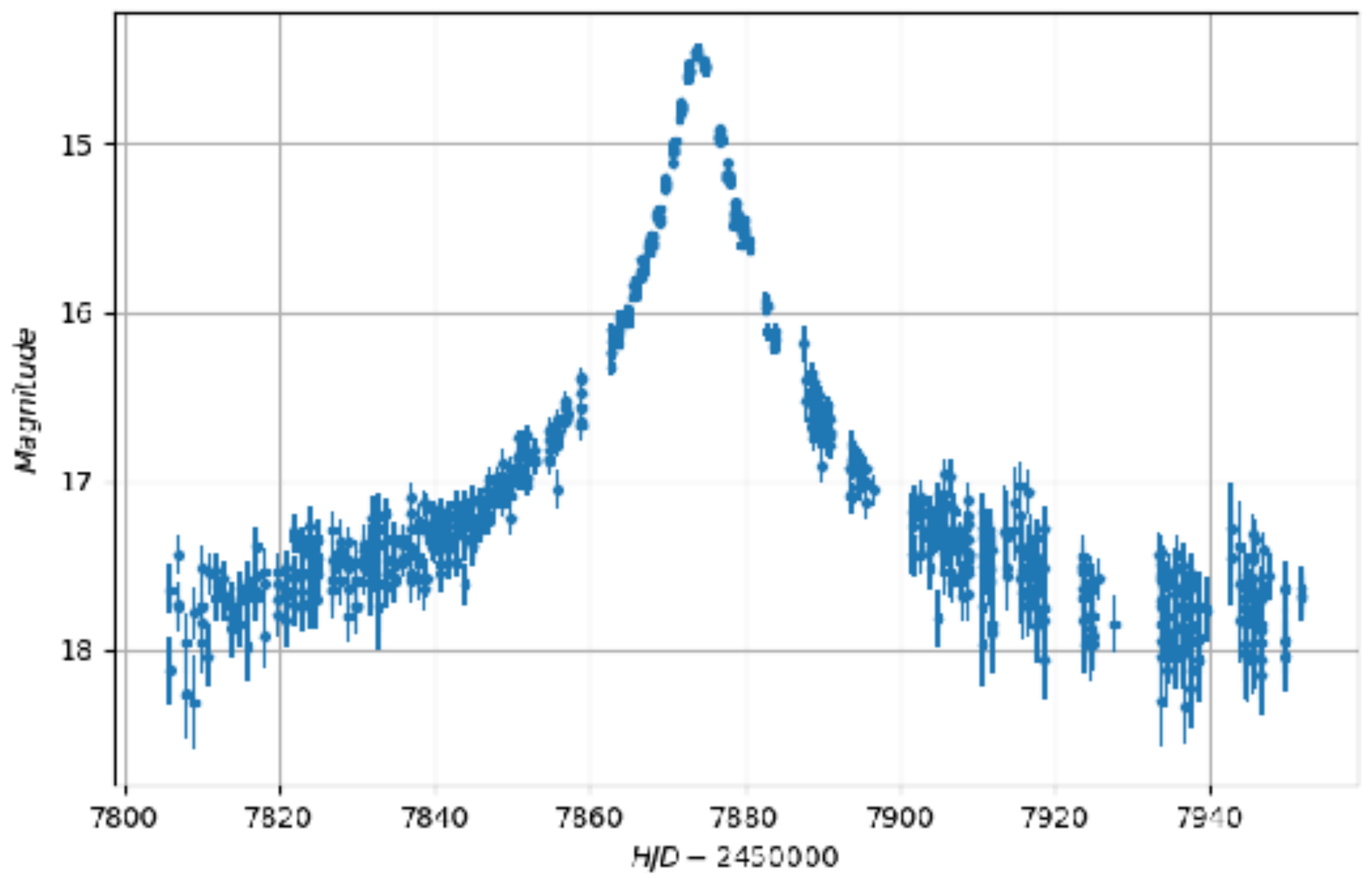
OB170482

KMT - CTIO

raw



filtered



Hardware requirements for GPU version

PC or Mac with an NVIDIA GPU available for computation (i.e. not being used to drive a display).

Software requirements

Linux or OSX/MacOS

NVIDIA CUDA drivers (for GPU version)

Python 2.7 with

- numpy / scipy / astropy / pyraf / scikit-learn
- pyCUDA (for GPU version)

```
#
#  Set the package install location
#
import sys
sys.path.append('/home/mda45/PythonPackages')

#
# Import the high-level pipeline routines
#
from pyDIA import DIA_GPU as DIA

#
# Load parameter defaults and override if necessary
#
params = DIA.DS.Parameters()
params.gain = 1.55
params.readnoise = 5.0
params.loc_data = 'Images'
params.loc_output = 'Output'


#
# Perform the difference imaging and photometry
#
DIA.imsub_all_fits(params)
```

Typical timings

For 512 x 512 pixel KMT subimages:

> 10,000 stars, 3 s per image for difference imaging (with degree 1 kernel) and photometry (using a GP100)

Availability

Open source.

Download from:

https://github.com/MichaelDAlbrow/pyDIA