Status of Pan-STARRS Image Data Simulator and Prototype Pipeline

Nick Kaiser

August 21, 2003

Abstract

This talk is an update on development of the image data simulator and prototype image processing pipeline. The prototype image processing pipeline is a straightforward generalization of the existing pipeline to be able to handle the the entire sky (as a set of overlapping tangent planes). It has been augmented by a crude data simulator that generates mock input data. I will list some of the physics that should be included in the simulations, describe what has been implemented so far and present some examples. I will show some tests of the performance of a simple image mapping algorithm.

The software described here is by no means complete. What is currently in place provides a framework within which we can experiment with different algorithmic choices, and others are strongly encouraged to join in, but it still needs considerable input.

Contents

1 Introduction **2**

2 Current Status of the Data Simulator **4**
2.1 Location of the Code 4
2.2 Division of the Sky into Cells 4
2.3 Parallelization 7
2.4 Input Objects 7
2.5 Telescope PSF 8
2.6 Pixel Mapping 13
2.7 Generation of Mock Sky Flats 13
2.8 Vignetting, Detector Response, Sky Background and Noise 13
2.9 Sky Subtraction and Flattening 13
2.10 Forward Mapping 16

3 Performance **19**
3.1 Photometric Precision 19
3.1.1 Aperture Magnitudes 19
3.1.2 PSF Magnitudes 20
3.2 Astrometric Precision 20
3.3 Timing Tests 20

1 Introduction

Development and testing of PS image pipeline (and other) software needs an image data simulator. The image data simulator should be able to generate realistic fake data to feed into the image processing pipeline and also generate some well understood approximation to the desired reduction pipeline output. In this way one can rigorously test the performance of the system. A data simulator will help address a number of issues:

- Development of algorithms.
  - What are the performance limiting steps?
  - What performance is expected? (So we will know if things are failing).
- Optimization of modules.
  - What are the pace-limiting modules?
  - What is the expected output given some standard test input?
- Testing the performance of different parallelization schemes and inter-process communication media.
- Testing the performance of different hardware options.
- Development of observing strategy.
  - How should we offset the relative telescope pointings?
  - Should we rotate the cameras with respect to each other?
  - Should we arrange for the diffraction spikes to be in different orientations with respect to the sky?

Many of these require an end-to-end simulation. Optimization of modules can be performed in isolation, but if we want to have example test input and output examples for each module then we also need end-to-end simulation.

There are two possible strategies here. One is to make the test comparison at the object catalog level:

```
input object catalogs        =====>          simulated focal plane data
        /\                                             ||
        || <-test                                      ||
        \/                                             \/
output object catalogs <== object finding <== image processing pipeline
```

An alternative is to make the test comparison between sky-plane images:

```
input object catalogs
        ||
        \/
simulated sky plane data         =====>          inverse pipeline
        /\                                              ||
        || <-test                                       ||
        \/                                              \/
processed sky plane data  <===  pipeline <==== simulated focal plane data
```

In the latter case, differences between the input and output sky plane images can arise from errors in both the inverse pipeline and in the pipeline, so errors will tend to be exaggerated. This talk will be mostly concerned with the latter type of simulation.

A partial flowchart for the planned data simulator is shown in figure 1.

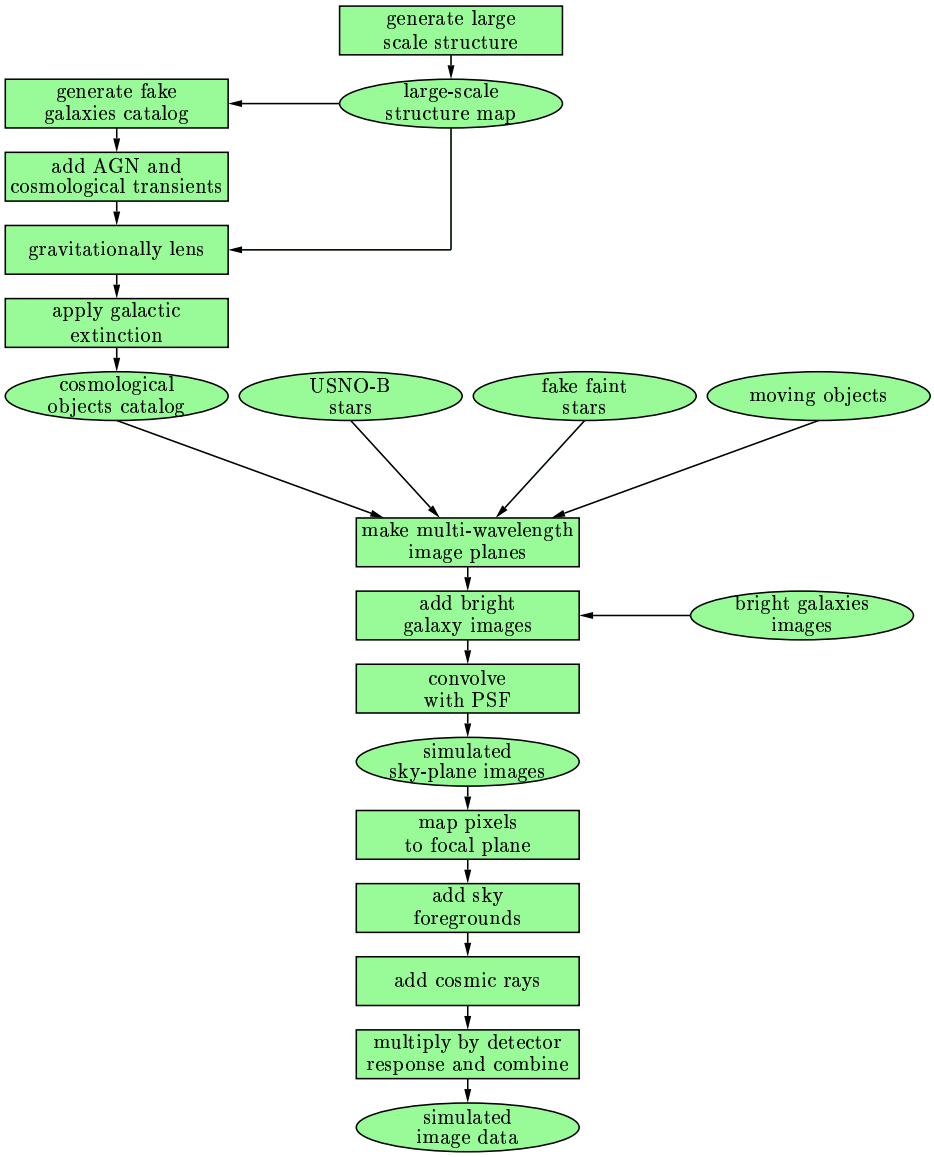

Figure 1: Data simulator flowchart. Currently only simple fake galaxies and real USNO stars are included and very crude models for the instrumental response, sky foregrounds etc are included. Other physics needed is patchy extinction models as well as models for galactic and extragalactic diffuse emission.

2 Current Status of the Data Simulator

2.1 Location of the Code

Raja Narayan has installed a read-only mirror of parts of my CVS repository on `surf.ifa.hawaii.edu` as an anonymous CVS server. It currently contains my `imcat` stuff as well as the code for the simulator and prototype pipeline. It is updated hourly.

To use this you should

```
setenv CVSROOT :pserver:anonymous@surf.ifa.hawaii.edu:/CVSROOT/kaiser/pan-starrs
```

and then use `cvs` to check stuff out (see the `cvs` man page, or get the "Cederquist" from `http://www.cvshome.org/do` for more info). For example, to checkout the prototype pipeline sources do

```
cvs checkout computing/tesselation
```

or to checkout the entire imcat distribution, do

```
cvs checkout imcat
```

See `http://pan-starrs.ifa.hawaii.edu/project/people/kaiser/cvsmirror/` for more information.

2.2 Division of the Sky into Cells

In the current implementation, the sky is divided up into a set of overlapping triangular tangent planes using a 3-level deep recursion starting from a base of 60 triangles obtained by triangulating a dodecahedron as shown in figure 2. A tangent plane has an identifier `tmmnnn`, where `mm` ranges from 0 through 59 and each `n` represents a digit in the range 0 through 3. These tangent planes are roughly 10 square degrees in area, and are covered with a set of overlapping 1024x1024 image planes. These are grouped together by a 5-level deep triangulation of the tangent planes as illustrated in figure 3. A group is identified by a string `gnnnnn`.

This is all highly programmable. The sky description parameters are all in `conf.d/sky.conf` and the two scripts that set up the sky cells are `makesky.pl` and `maketangentplanes.pl`. However, the sky division also interacts with the parallelization scheme (in the current implementation, the node ID for a sky cell is decoded from the low order digits of the group ID).

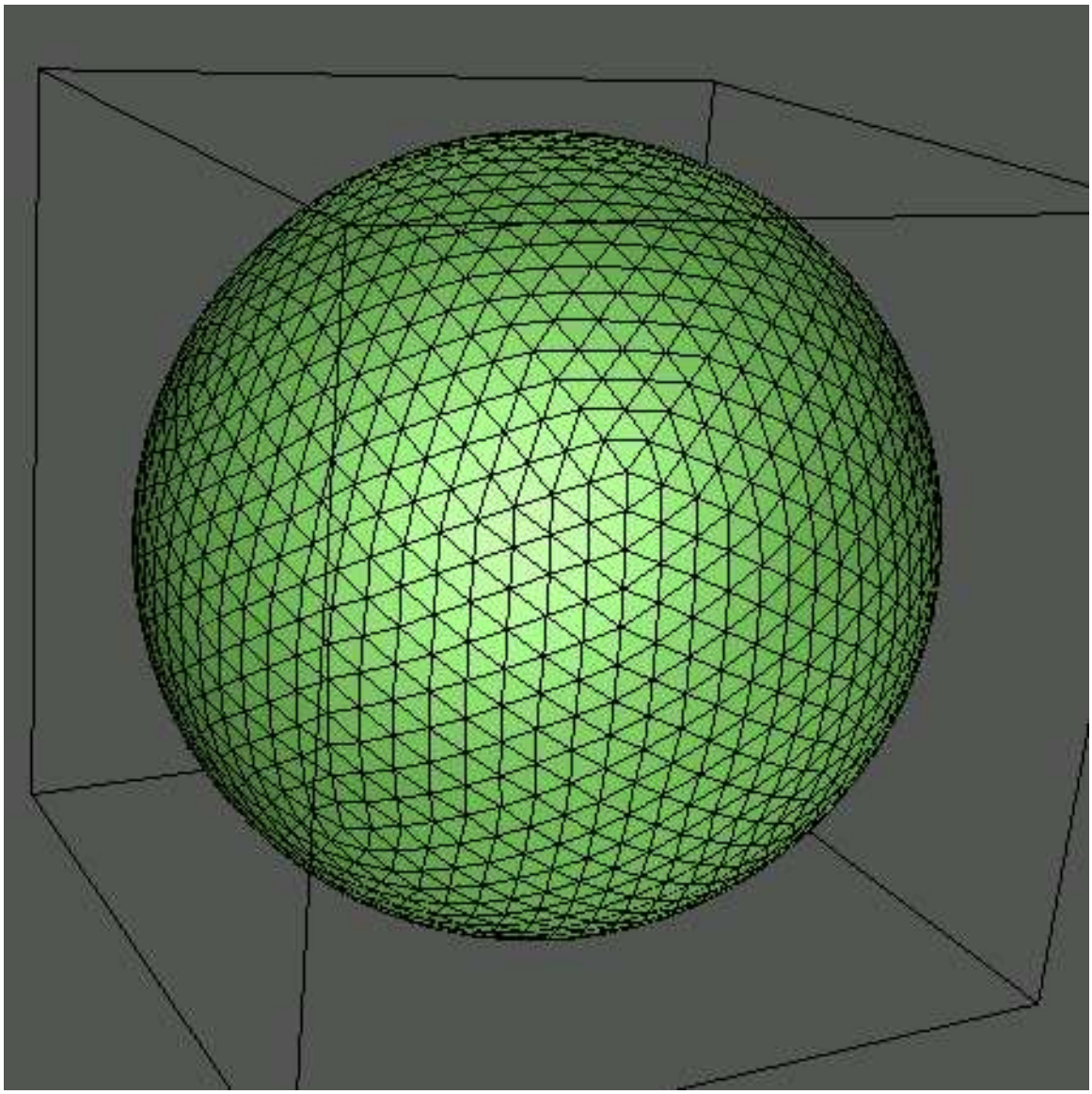

Figure 2: Sky tesselation.

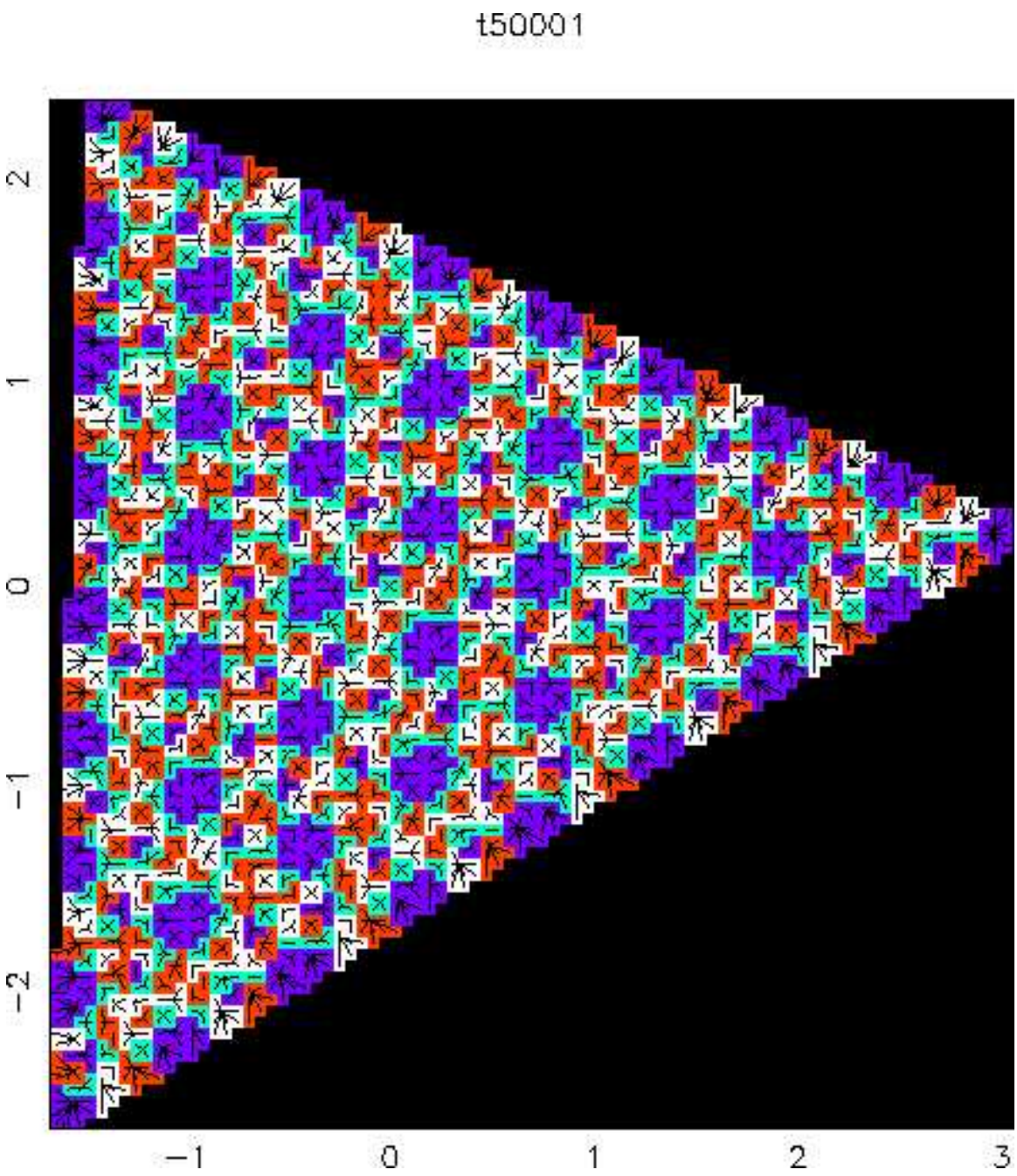

Figure 3: Tangent plane. A tangent plane consists of a large number of slightly overlapping sky-cells. These have been grouped together by recursively splitting the tangent plane into smaller triangles. This is done in order that the sky cells can be assigned to sky servers in such a way that the work is distributed evenly over the compute nodes.

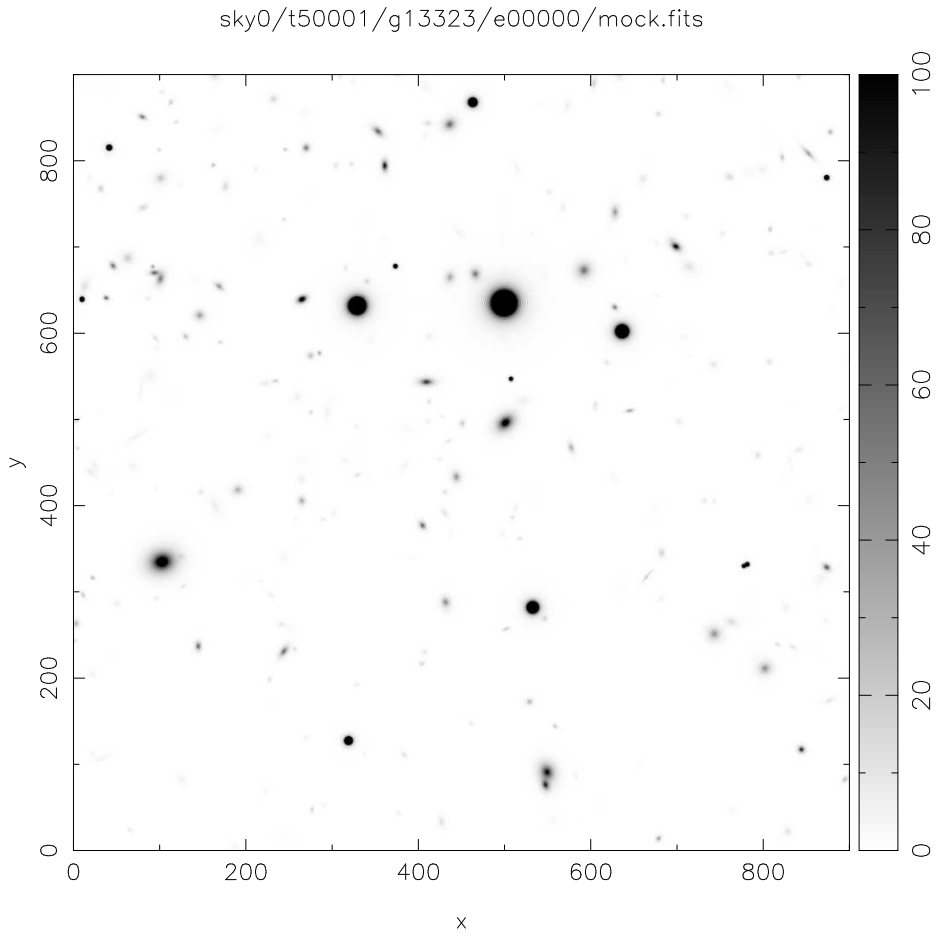

Figure 4: Mock sky cell.

2.3 Parallelization

The simulator/proto-pipeline system is implemented in a parallel manner. It is designed to be run on a set of similar, loosely coupled bricks. We define a set of detector processors (with the processor ID being decoded from the OTA ID) and a set of sky processors with processor ID encoded as above. Batches of tasks are executed using the imcat program `rshloop` (successor to `pvmloop` - we had lots of problems with PVM and it was slow at transmitting data, whereas `rsh` seems at least to achieve $\gtrsim 60\%$ of the theoretical maximum bandwidth). Customization of the system to allow it to run on different clusters is done by modifying the field `conf.d/slaves.conf`. This `perl` file contains arrays giving node names and node 'tags' that are used to generate scratch names, as well as defining functions that return node IDs given OTA IDs or sky cell group IDs.

2.4 Input Objects

Currently we include fake galaxies (drawn from a Schechter luminosity function and populating a cosmological model) and real stars from the USNO-B1 catalog. Galaxies are modeled as optically thick exponential disks. The mock sky cells are then convolved with a atmospheric seeing PSF. An example sky cell is shown in figure 4. This is a high galactic latitude field.

2.5 Telescope PSF

The telescope PSF was generated from the pupil shown in figure 5. The support struts were assumed to be rotated by multiples of 22.5 degrees for the four telescopes. A scale invariant random phase error with total rms value of 0.2 radians was included to simulate fine scale mirror roughness. Scale invariant (2-dimensional flicker noise) was chosen to match empirical $1/r^2$ aureoles (see figure 6). The logarithm of the PSF for one of the telescopes is shown in figure 7 and the profile is shown in figure 8.

The convolution of a mock sky cell (chosen to contain some particularly bright stars) with the PSF for telescope 0 is shown in figure 9.

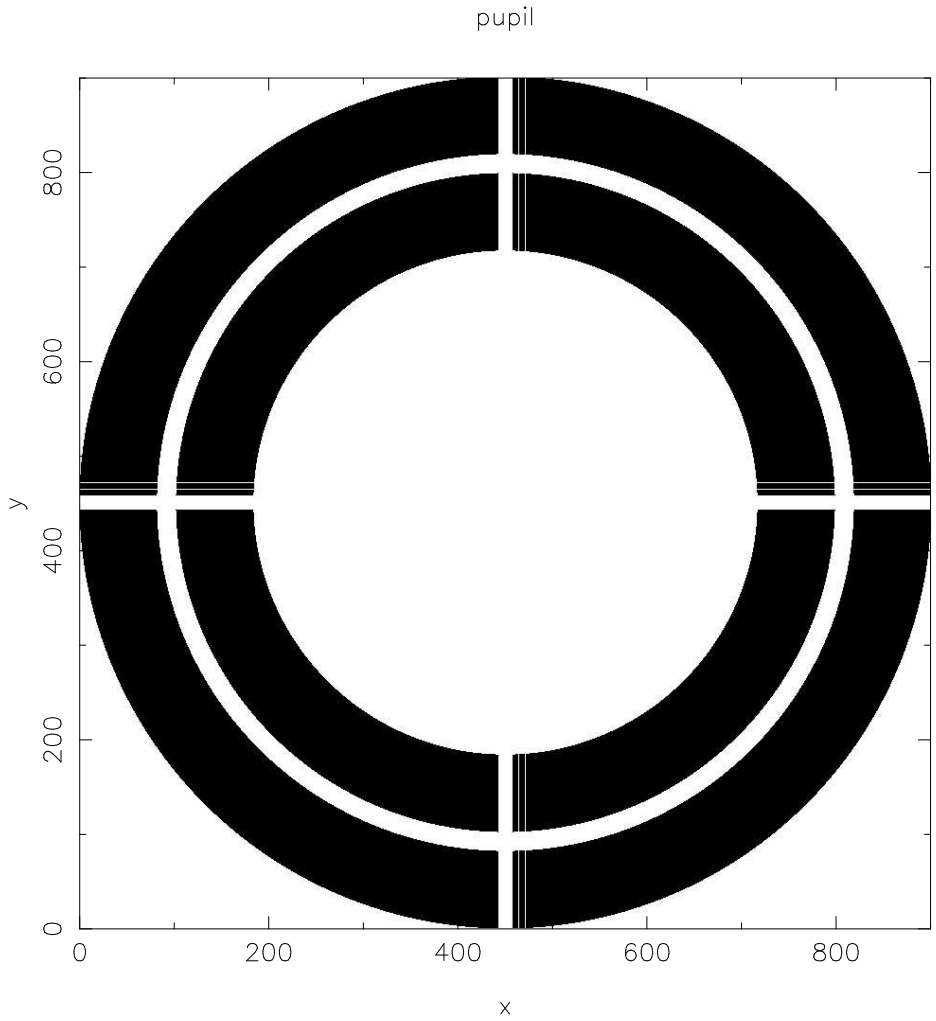

Figure 5: Model pupil function. The obstructing annulus represents the baffle. The thick and thin radial obstructions are the secondary support struts and conical baffle support wires.

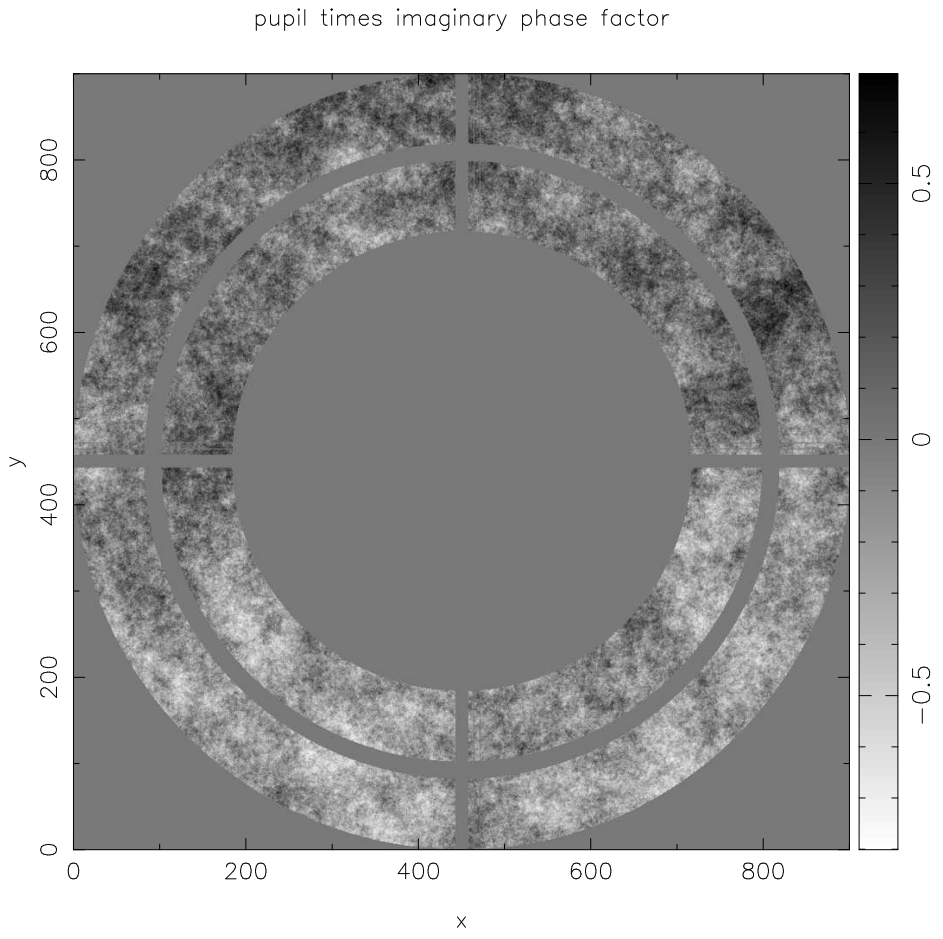

Figure 6: Model pupil function times the imaginary part of the fine-scale mirror roughness phase factor.

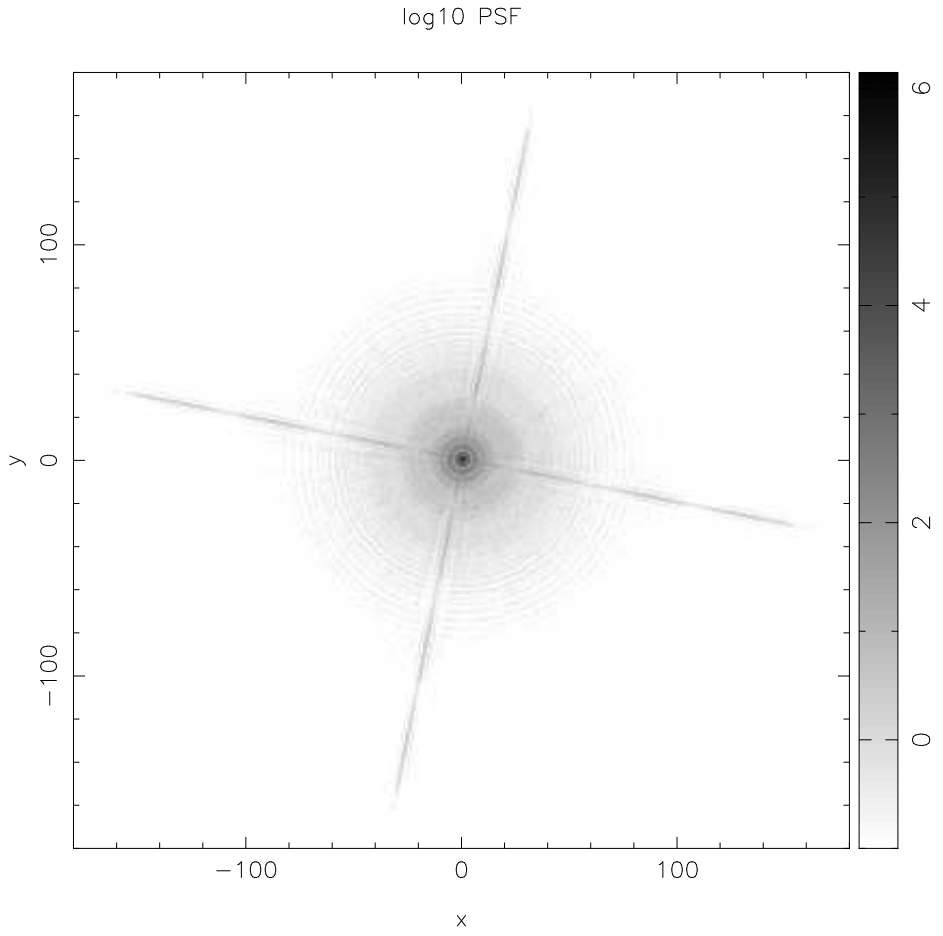

Figure 7: Logarithm of point spread function. The PSF was calculated using 5 wavelengths distributed over a bandwidth $\Delta\lambda/\lambda \simeq 0.25$.

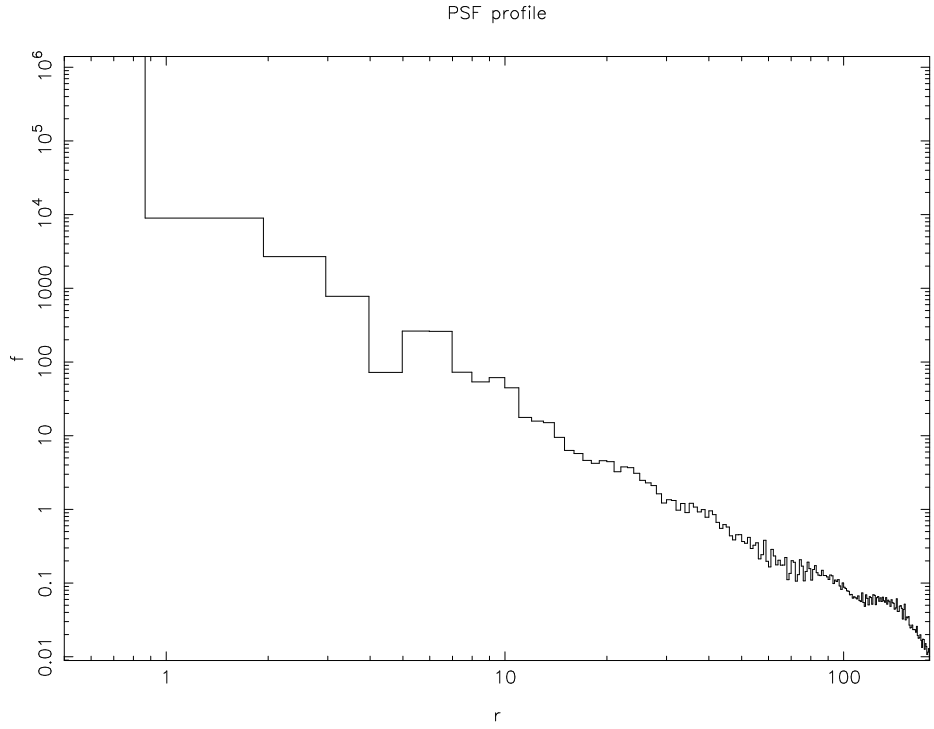

Figure 8: Telescope PSF profile *vs* radius in pixels. The inverse square aureole comes to dominate beyond about 50 pixels or around 10 arcsec.

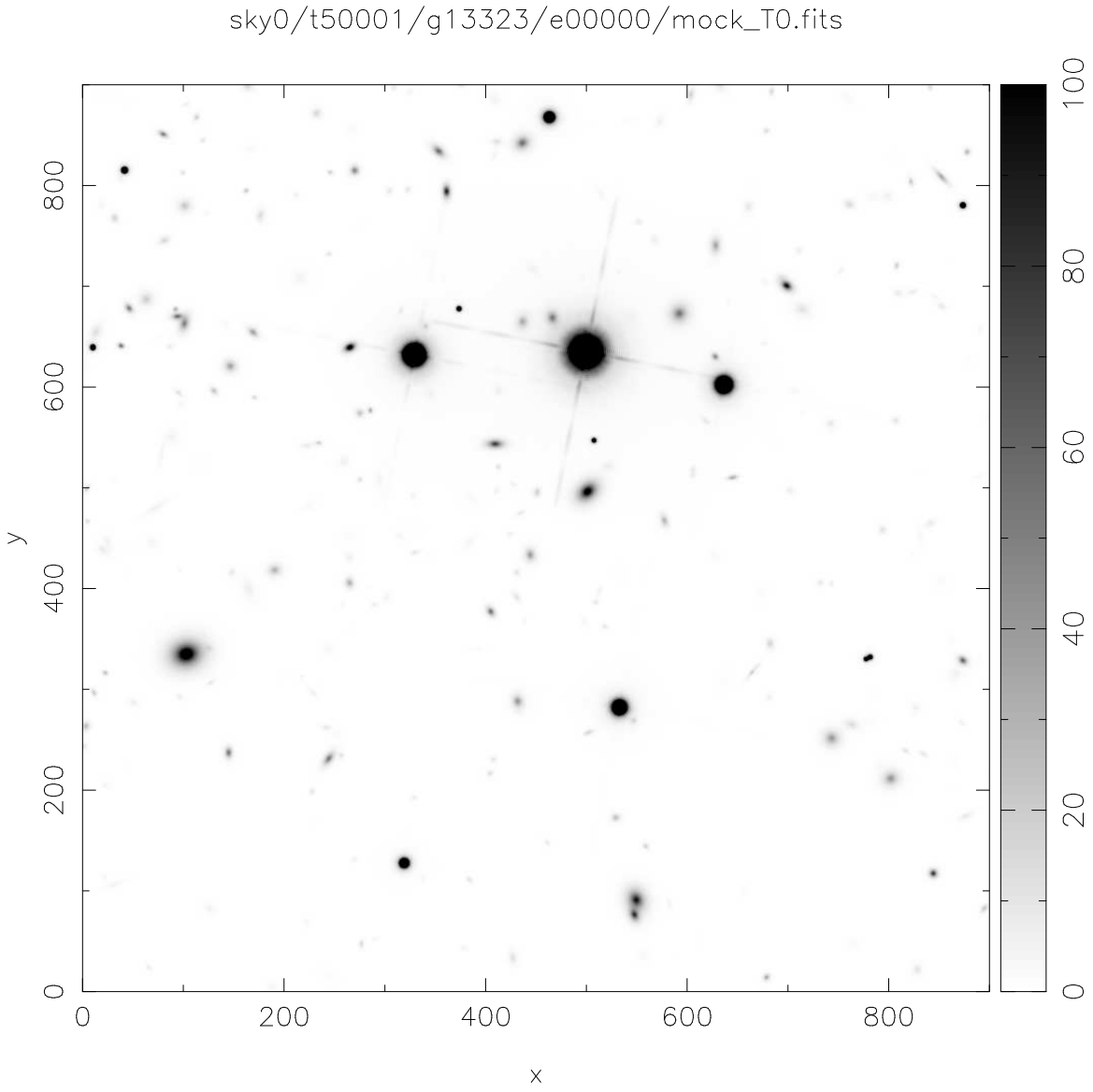

Figure 9: Mock sky cell image convolved with PSF for telescope 0 as well as atmospheric PSF.

2.6 Pixel Mapping

The mapping from sky coordinates to focal plane coordinates was obtained using a quintic model for the field distortion obtained by fitting a polynomial to the output of ZEMAX for Klaus Hodapp's design.

From the pointings for the four telescopes in an exposure we generate a set of cell-pair overlap files `ennnnn.omm.tppppp.gqqqqq.Tr.cells.cat` that give the sky-plane coordinates of the corners of the detector cells in OTA `omm` in exposure `ennnnn` that overlap sky cells in sky cell group `gqqqqq` in tangent plane `tppppp` for telescope `Tr`. These are used for 'forward mapping' from detector coordinates to sky plane coordinates. As well as defining the spatial transformation, these provide the essential index from sky cells back into the source image data base.

We also define a set of inverse overlap pairs `tppppp.gqqqqq.ennnnn.omm.Tr.cells.cat` that give the focal plane coordinates of the sky cells. These allow 'inverse mapping' from sky to detector coordinates of the mock sky images.

The mapping is performed by a pair of commands `sendpix` and `recvpix` (the latter being invoked by the former). The spatial transformation is a linear transformation. In this scheme a `sendpix` process is launched on each OTA server and it uses `rsh` to launch the `recvpix` process on the appropriate sky-server nodes. This 'pixel pushing' algorithm is probably inefficient and will be replaced by a command that runs on the sky-servers and launches remote commands on the OTA nodes to send the pixels.

The pixel mapping algorithm currently used is 'cloud-in-cell' charge assignment. We generate both a 'signal' and 'exposure' map. Detector plane images are generated by dividing the signal by the exposure map. A inverse mapped detector cell image is shown in figure 10

2.7 Generation of Mock Sky Flats

We generate, for each OTA

- A mock sky-flat: This has a quadratic roll off to mimic vignetting and a Gaussian random pixel sensitivity.
- A super-flat: A random Gaussian variable to represent the difference in the QE for the night sky as compared to that for a 'normal' SED.
- A mock fringe frame: The cosine of a smoothed Gaussian random field.
- A bad pixel mask: Some randomly chosen columns.

2.8 Vignetting, Detector Response, Sky Background and Noise

For each OTA we take a random multiple of the fringe frame and add to that the super-flat and multiply by an appropriate factor to give the sky background of desired level. We add to this the inverse mapped mock sky image and multiply by the sky flat and by the bad pixel mask. An example is shown in figure 11. At this point we add a Gaussian random variate with variance equal to the number of electrons in order to simulate photon counting noise. The examples shown here have sky noise appropriate for 30s integrations in the r-band.

2.9 Sky Subtraction and Flattening

The procedure for estimating and subtracting the sky is as follows:

- Divide by the sky flat.
- Set bad pixels to magic.
- Get the mode and sigma of all the sky cells in the OTA.
- Take the average.

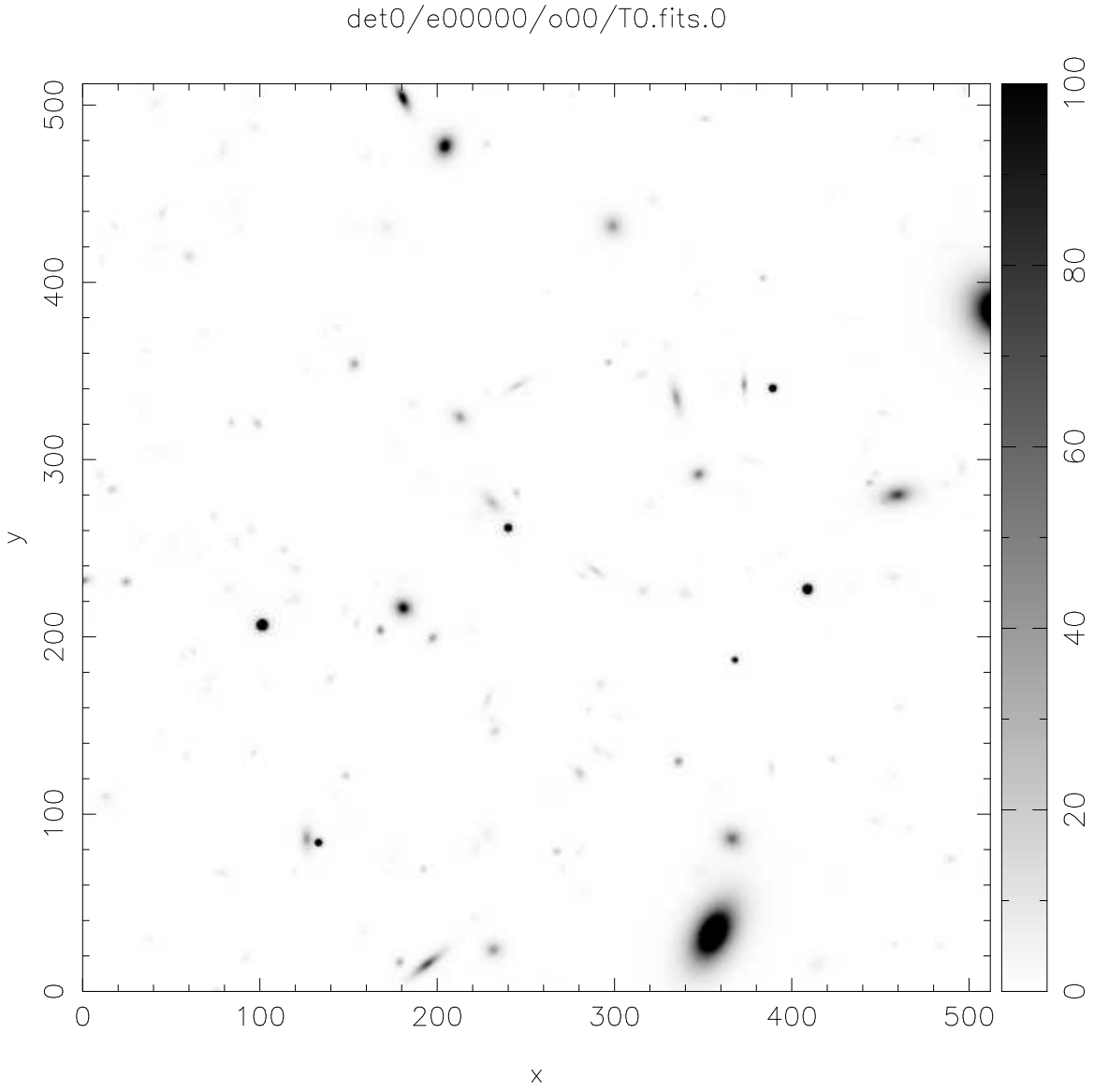

Figure 10: Example of detector cell image. No noise or instrumental response has been added yet.

- Clip the image at ± 3-sigma and replace clipped pixels with magic values.
- Set all neighbors of magic pixels to magic value and repeat.
- Generate a non-object mask (unity/zero for all non-magic/magic pixels).
- Dot the non-object masked image with the super-flat.
- Dot the non-object masked super-flat with itself.
- Divide the results of the last two steps to get the amplitude of the super-flat.
- Multiply super-flat by the amplitude and subtract.
- Compute amplitude for fringe pattern in a similar manner and subtract scaled fringe.

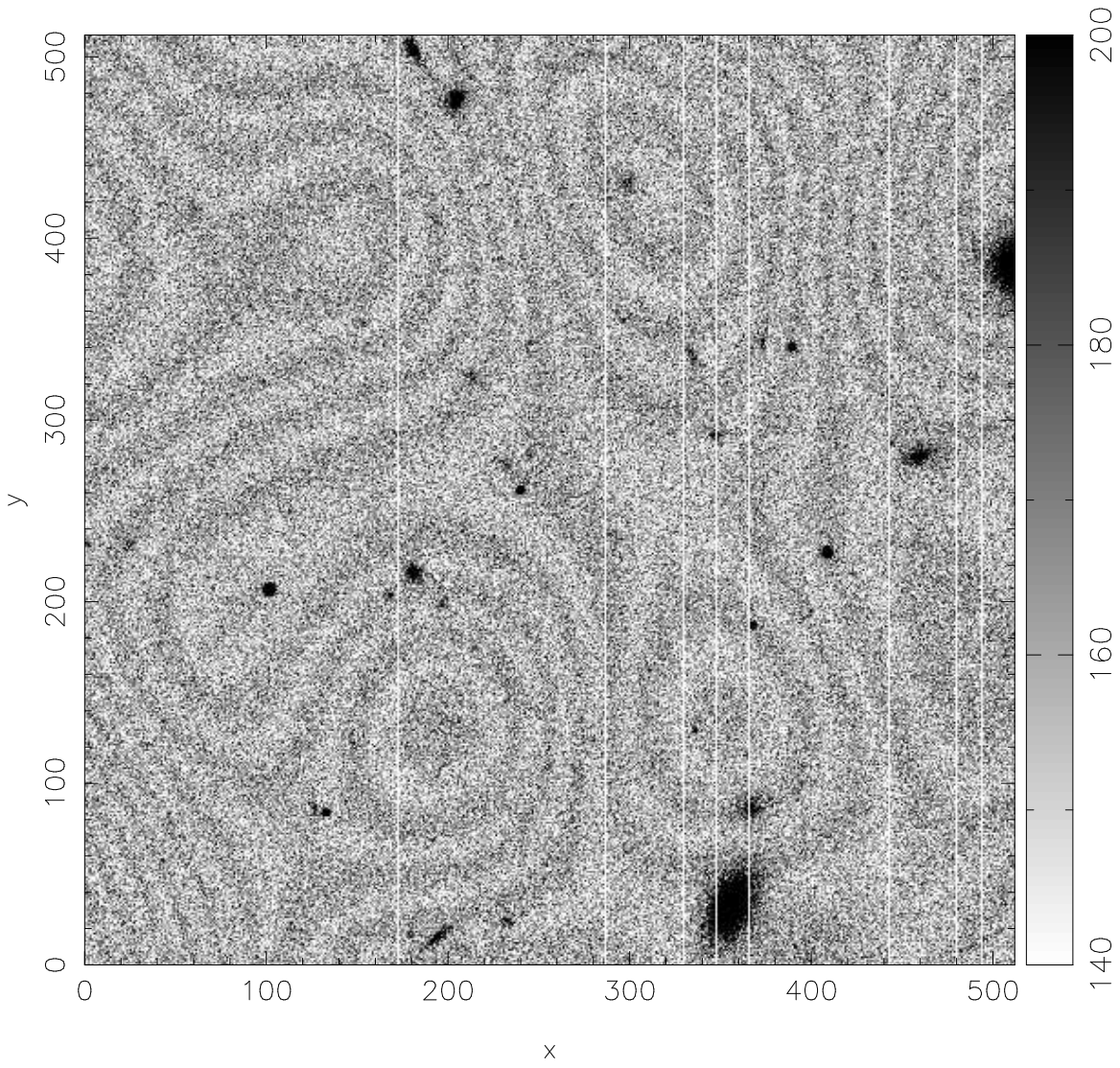

Figure 11: Example of detector cell image. This has been degraded to simulate the effect of vignetting, pixel to pixel sensitivity variation, air-glow and photon counting noise.

The result is shown in figure 12. Subtracting from this the original inverse-mapped image shows a negative bias of about 0.1 electrons, whereas the noise level is around 12.7 electrons rms.

This is very crude and simplistic. It is also a cheat, as we are using the true sky-flat, super-flat and fringe frame. The next step is to simulate a calibration run — a set of widely dithered blank sky images — and see how well one can recover these. Other avenues for further development include

- More careful job by masking of fainter objects.
- Global sky fitting (use all of the information from an exposure to determine the sky component amplitudes — the current implementation solves for each OTA independently).
- Extended galaxies, extragalactic diffuse emission.
- Relative line strength variations?

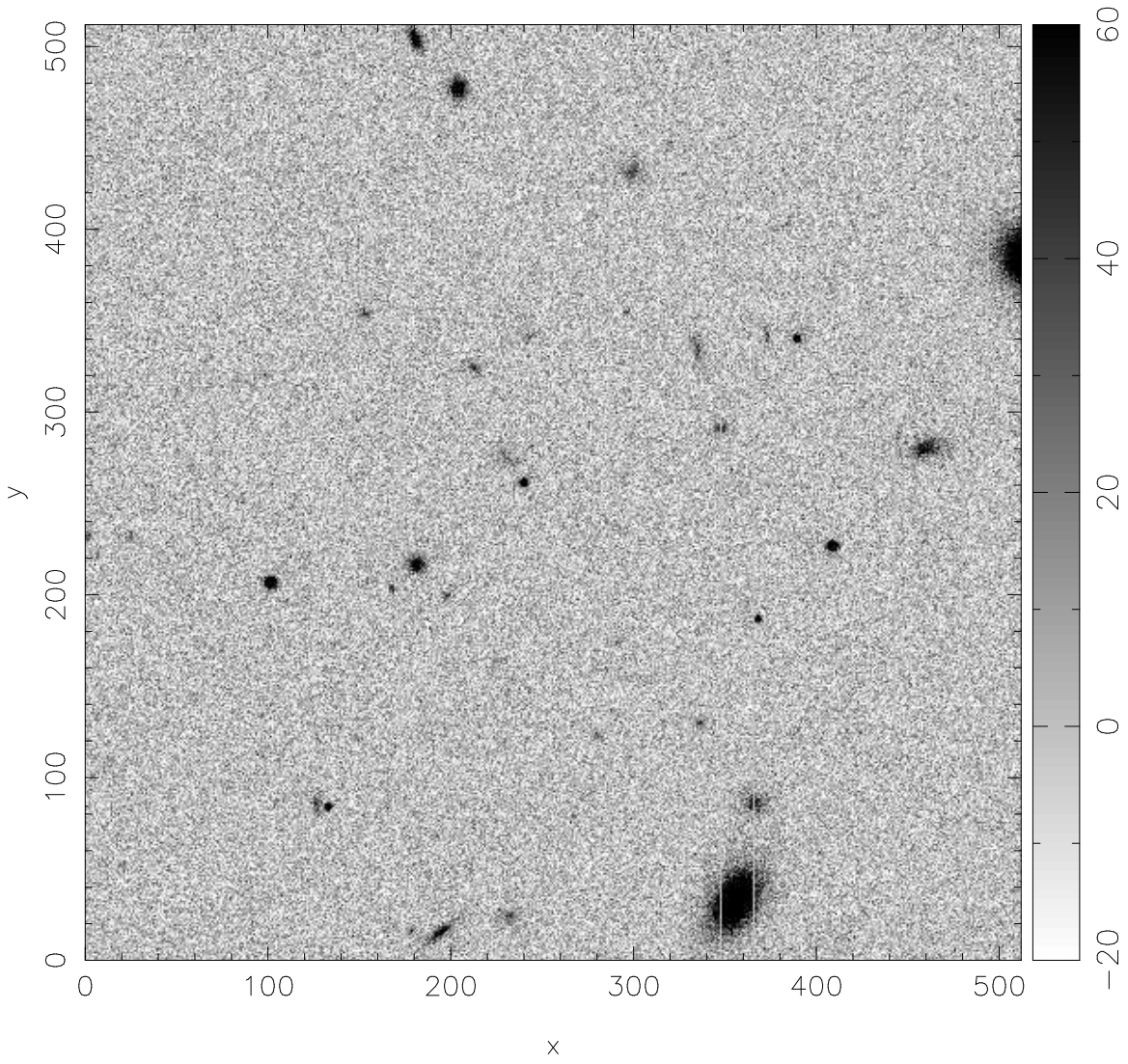

Figure 12: Sky subtracted and calibrated image.

- Zodiacal light, twilight, scattered moonlight.
- Ghosts, glints, light scattered off baffles and telecope structure.
- Cosmic rays.

Low galactic latitude observations will require different procedures.

2.10 Forward Mapping

The result of mapping from sky to detector plane and back is shown in figure 13. The original, mapped and difference images are shown in figure 14.

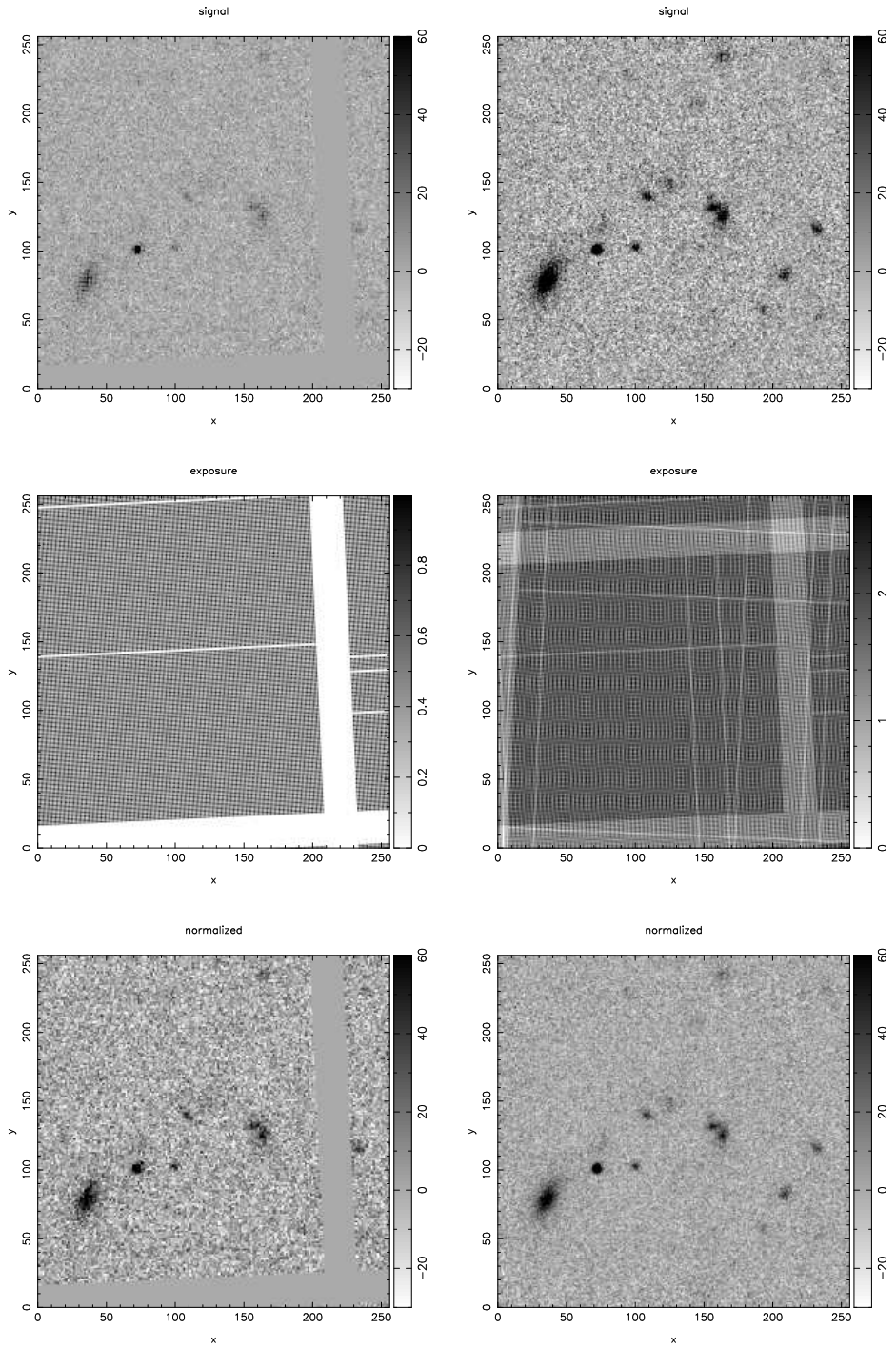

Figure 13: A small section of a sky cell showing the signal, exposure and normalized images. On the left is the result for a single telescope. On the right is shown the result for four telescopes combined.

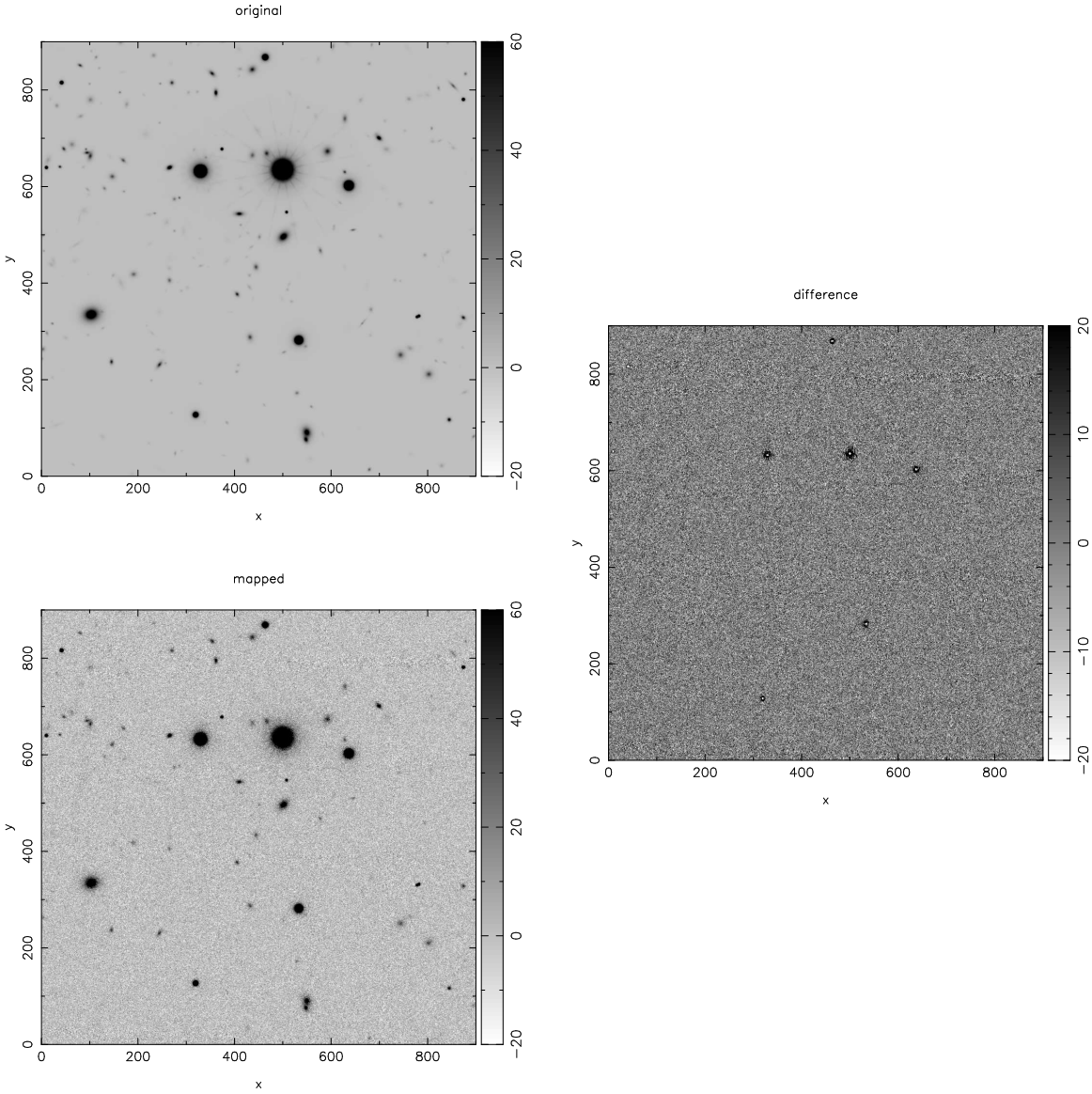

Figure 14: Original, (twice) mapped, and difference image. No PSF matching has been attempted.

3 Performance

3.1 Photometric Precision

3.1.1 Aperture Magnitudes

Cloud in cells charge assignment conserves zeroth and first moments of the charge distribution, but the normalization breaks exact charge conservation. We can measure this by looking at the bright USNO-B1 stars in the simulated images.

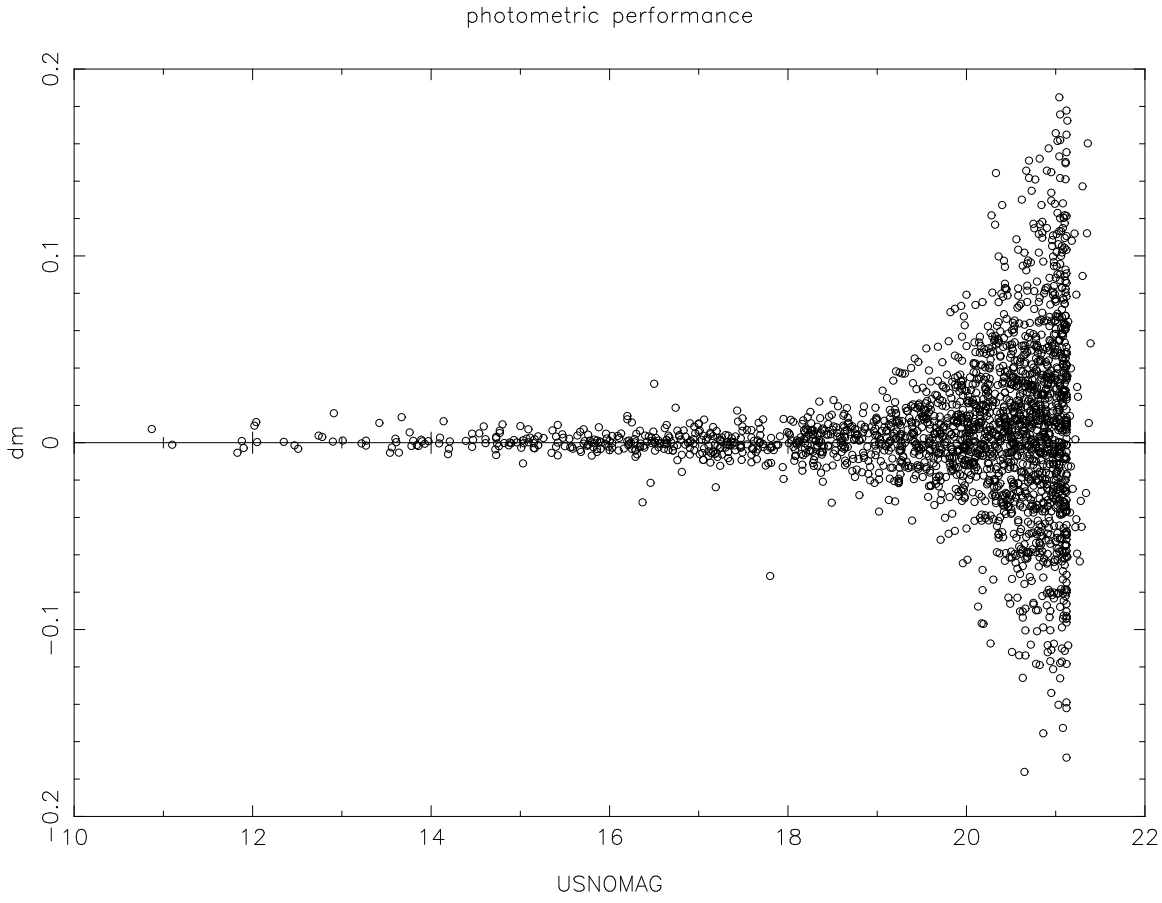

Figure 15: Difference in magnitudes measured from original and twice mapped images using USNO star centers and 10 pixel apertures. Faint objects are photon counting limited. For bright objects, the rms magnitude error bottoms out at around 3.6 milli-mag.

3.1.2 PSF Magnitudes

More precision is possible for faint point sources by performing likelihood analysis. Given the raw pixel positions and charge values, and in the absence of noise, this algorithm gives arbitrarily accurate positions and fluxes (subject to the sources being isolated and the validity of the pixel PSF model). It is interesting to ask, how accurate is this algorithm if applied to accumulated images mapped onto the sky?

To answer this, a grid of star positions was generated. This was rotated with respect to the sky coordinate frame so that the positions sampled all distances from the sky pixel centers. These point sources were mapped to a very fine-scale working image (16 times finer sampling than the detector pixels) and were convolved with atmospheric and pixel PSFs. These were then sampled at a set of 4 grids of detector pixel locations. The 4 grids were rotated slightly with respect to each other to minimize aliasing. The pixel PSFs were also rotated. It was then confirmed that likelihood analysis using a fine calculation grid gave essentially exact (few micro-mag) flux densities. The pixels were then mapped onto a 0.2 arc-second sky grid. The unsmoothed normalized image (signal divided by exposure) is shown in figure 16 and 17. PSF magnitudes determined from the latter image have rms scatter of about 3.5 milli-mag. This is very similar to that obtained from aperture magnitude analysis. We suspect that it will be hard to better this with this sky pixel size.

The choice of sky-pixel pitch is a trade-off between precision and cost. For the nominal 0.2 arc-second sampling rate, the pixel mapping algorithms used here result in irreversible loss of information for bright objects; for objects with photon limited photometric uncertainty worse than a few milli-mags this is negligible. For brighter objects, more information can be extracted from the detector plane pixel values.

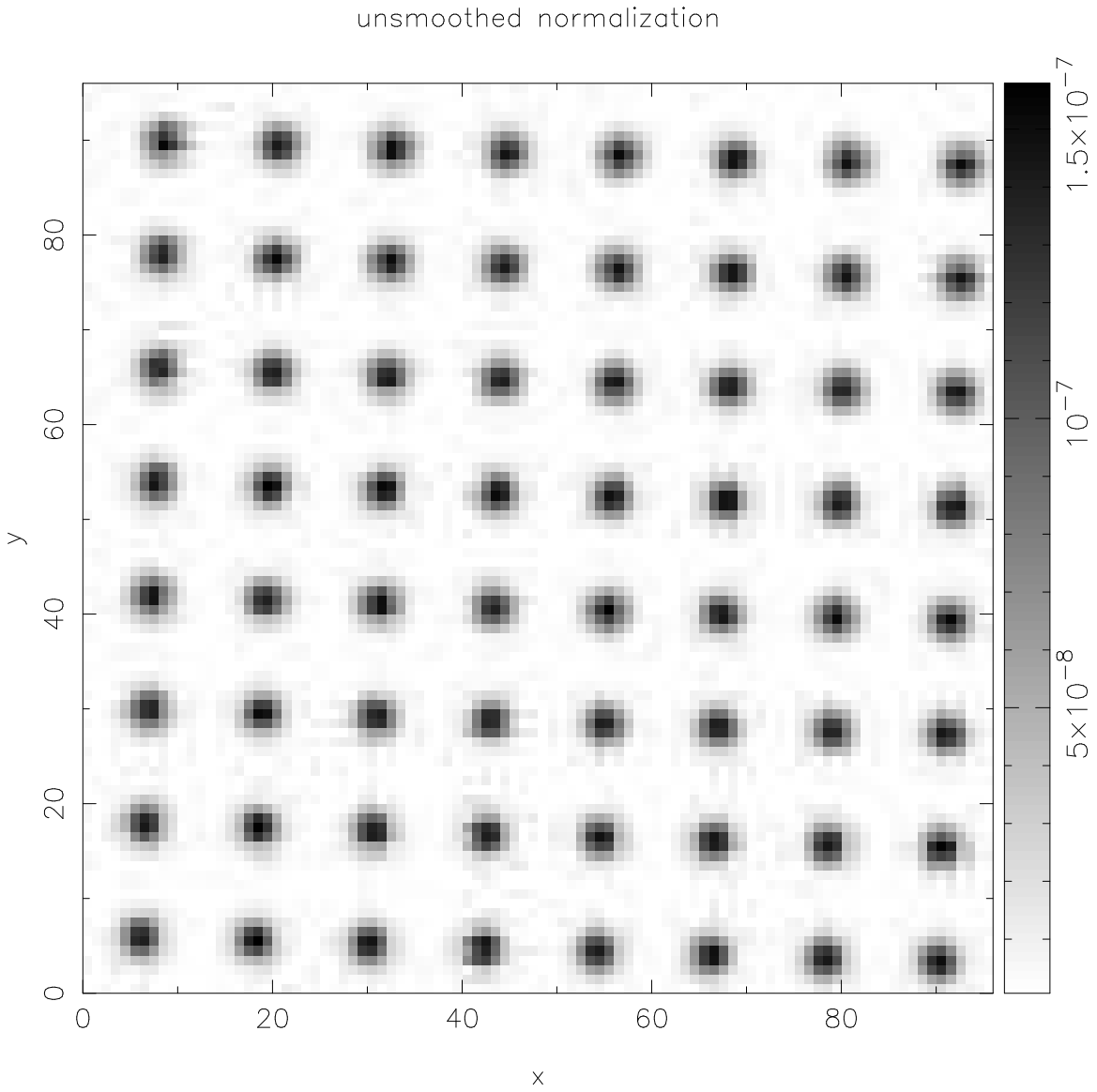

Figure 16: Mapped pixels for a field of artificial stars. Exact pixel values were generated as described in the text. Normalization was performed by dividing signal by exposure with no smoothing.

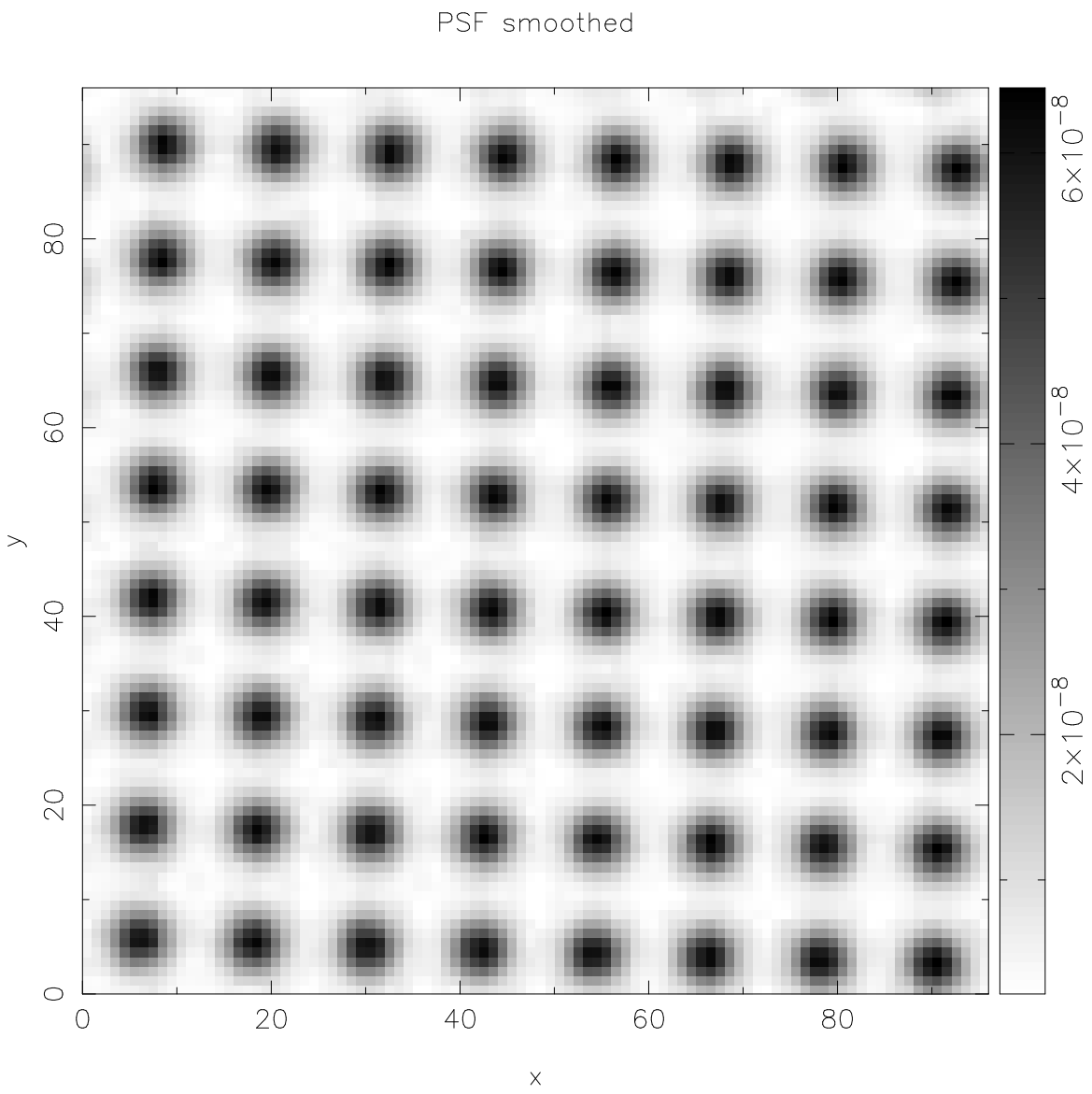

Figure 17: Mapped pixels for a field of artificial stars. Exact pixel values were generated as described in the text. Normalization was performed by dividing signal by exposure after smoothing with the PSF. Peaks of this image are uniform to about 2% (i.e.$\sim$ 20 millimag precision). However, errors correlate strongly with source position relative to the nearest pixel center. Modeling the residuals as quadratic and subtracting reduced the magnitude errors to approximately 3.5 milli-mag.

3.2 Astrometric Precision

The errors in positions of USNO stars recovered from the mapped images is shown in figure 18. The rms error seems to bottom out at about 6 milli-arcsec. This is similar to the precision obtained using the same object finding algorithm on CFHT images.

3.3 Timing Tests

Experiments on the `cmb` cluster (Dual 1.6?GHz Athlon, GigE switch) with 4 nodes assigned as OTA servers and 4 nodes assigned as sky servers was capable of mapping pixels in approximately 20s. However, the current prototype is grossly inefficient. A near term goal is to implement a sky-cell initiated pixel mapper.

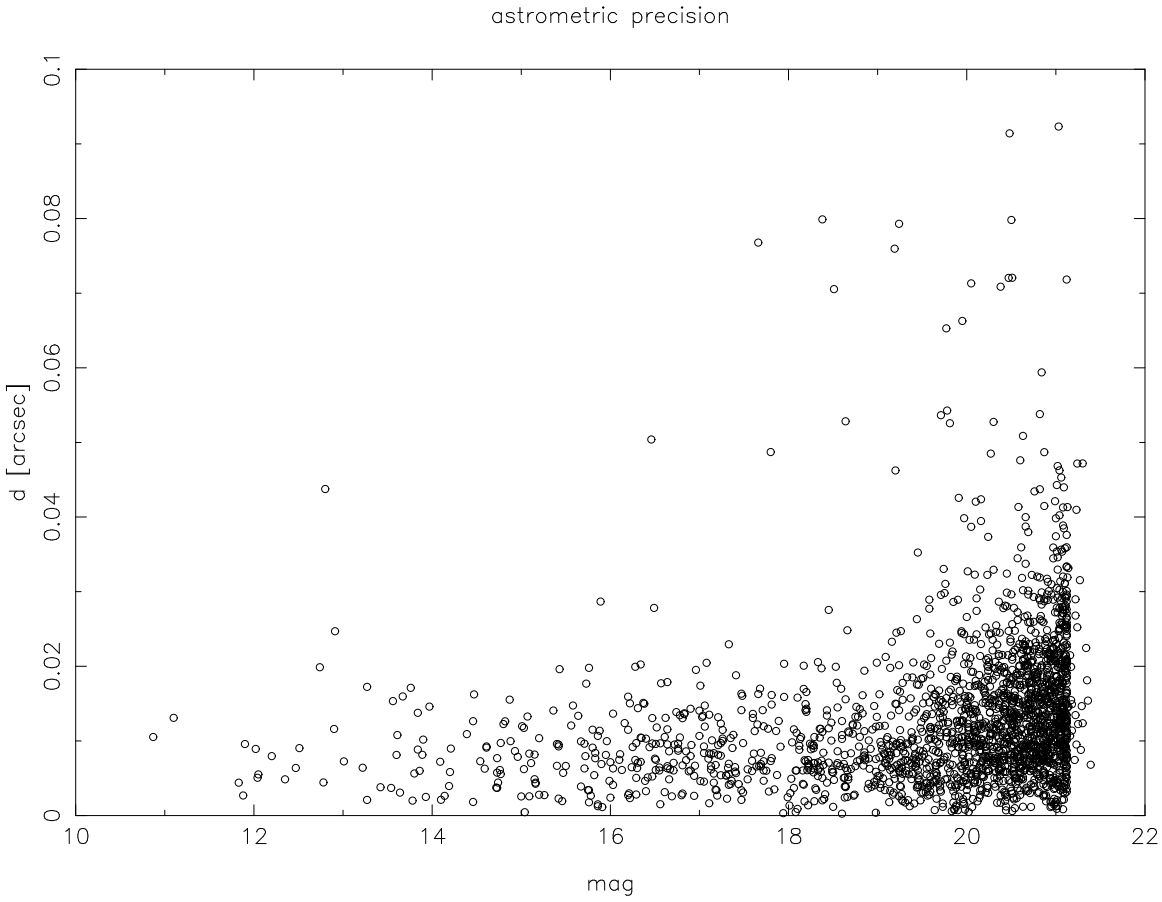

Figure 18: Errors in positions of USNO stars recovered from the mapped images. Rms error is $\simeq$ 6milli-arcsec for the brighter objects ($m < 16$ to be precise), rising to approximately 10 milli-arcsec for the faintest objects shown here.