Sensor testing and simulation meeting (Brighter-Fatter Study)

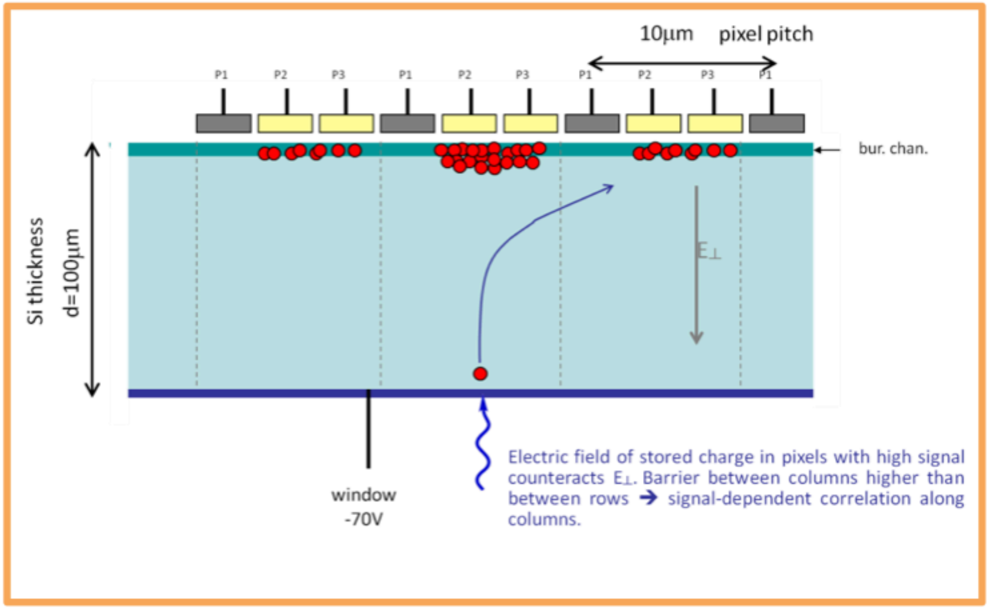
C. Walter

2/10/2014

Intro

- The goal is to simulate and compare the brighter-fatter effect between the Phosim output and data for various algorithms.
- Start by setting up analysis framework and sources in the MC.
- Start by generating a calibrated “perfect” spot source so we can predict and understand the performance.
- Start by trying to compare with the measurements made by P. Doerty used in Astier’s study using the model in Phosim v3.3.2.

Reminder: Spot data



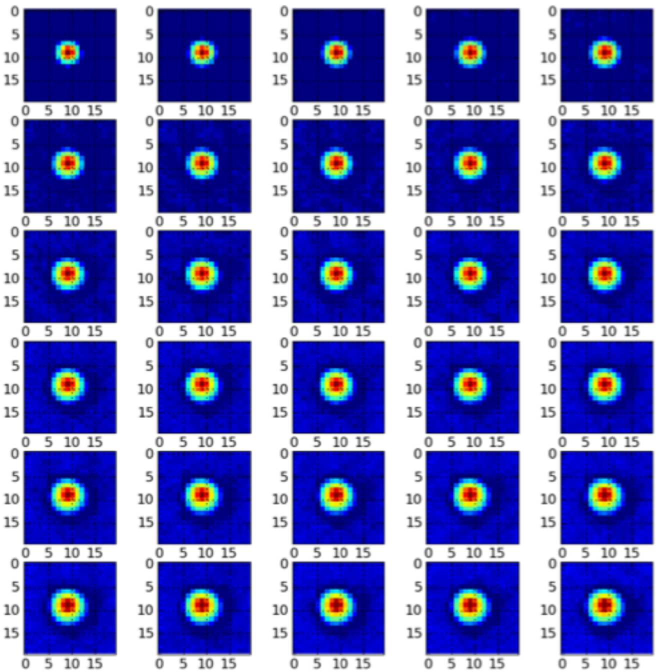
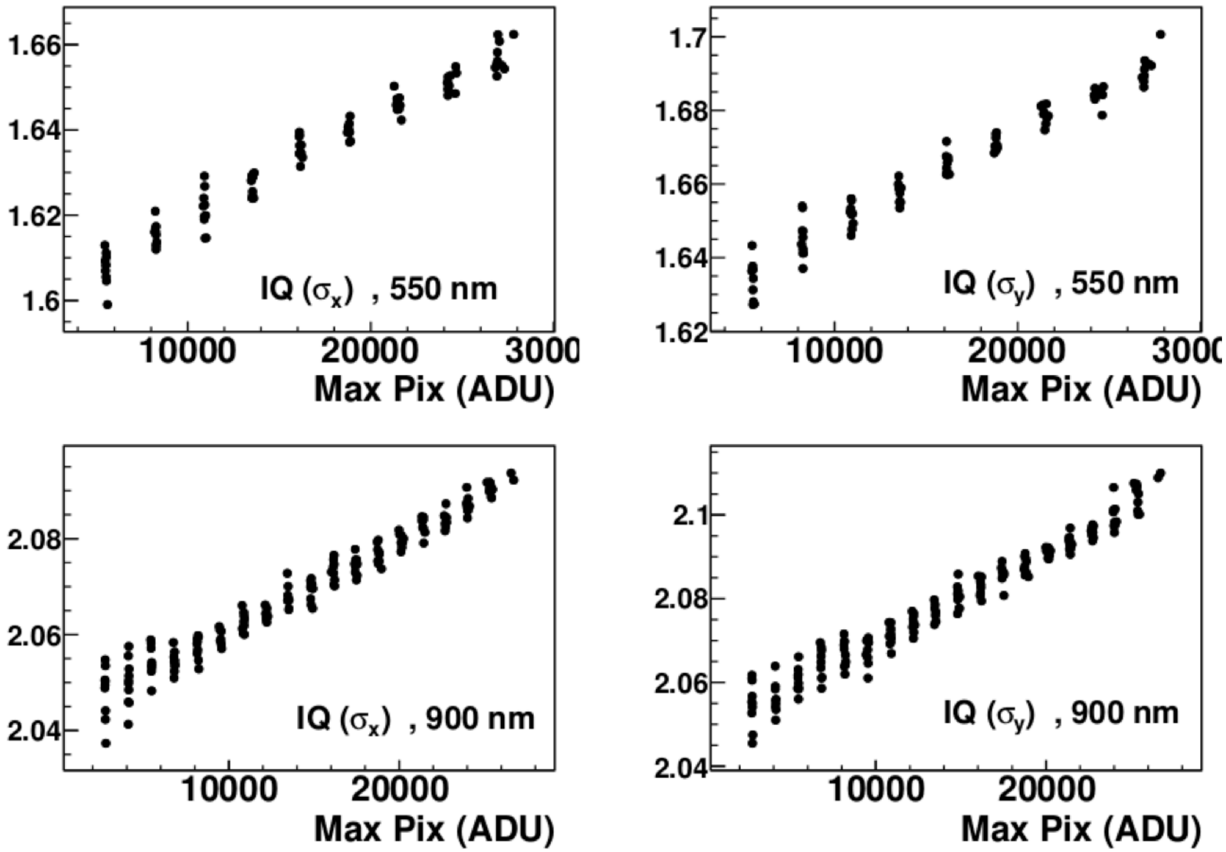


Figure from Andrei:
Drift should happen in the bulk.

Data taken by Paul.

Data taken by P. Doherty @ Harvard for e2v



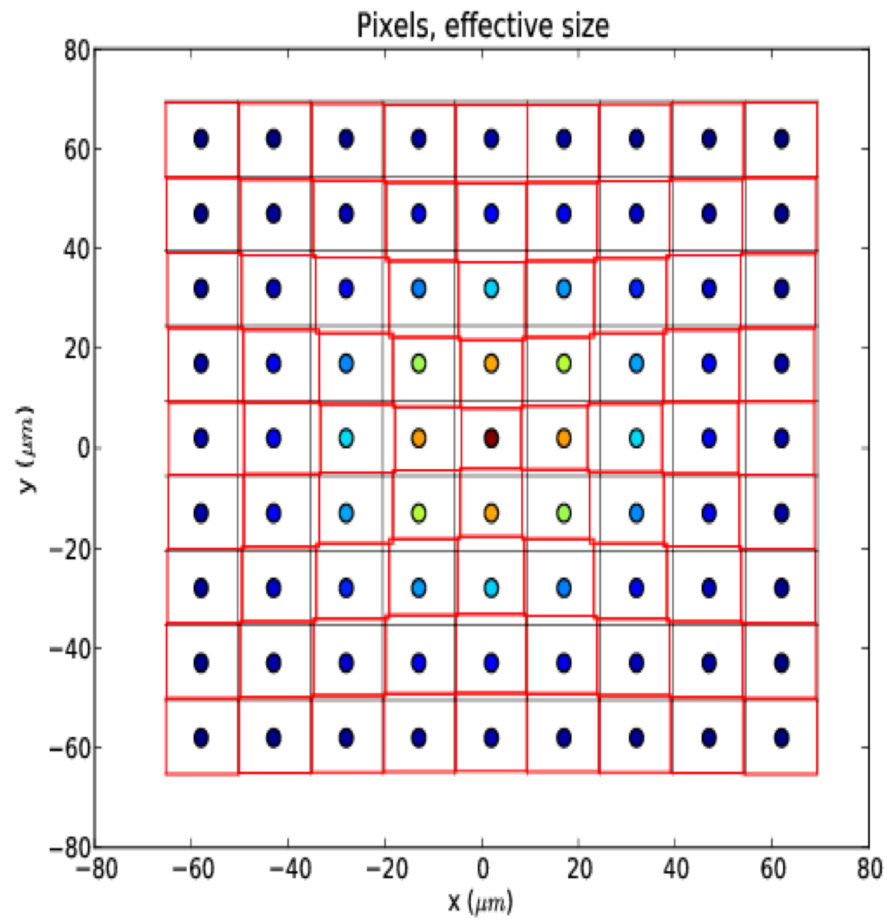
2-3% change over range. Notice comparable changes in both X and Y.

<u>What is range and gain Here?</u>

From Astier et al paper/talk from BNL conference.

Astier et all model

- In this model we view the distortion of edges as an effective change of pixels size as the collecting area is modified by the field for each pixel.



From Astier et al talk from BNL conference.

Make a "perfect source"


```
# Make a subpixel test (if pixelsize < 10.0)
# pixelsize 0.1

# Set pixel depth and parameters
# well_depth 1000000
blooming 0
saturation 0

# Turn on/off the sharing to other pixels
chargesharing 0

# Turn off other effects
diffractionmode 0
telescopemode 0
lascatprob 0.0
detectormode 0
atmosphericdispersion 0

rotationjitter 0.0
elevationjitter 0.0
azimuthjitter 0.0

clearperturbations
```

Need to calibrate source in electrons.

A magnitude 20 star with these settings
Results in 507967 electrons (no saturation)

We can calculate the magnitude for a given electron level:

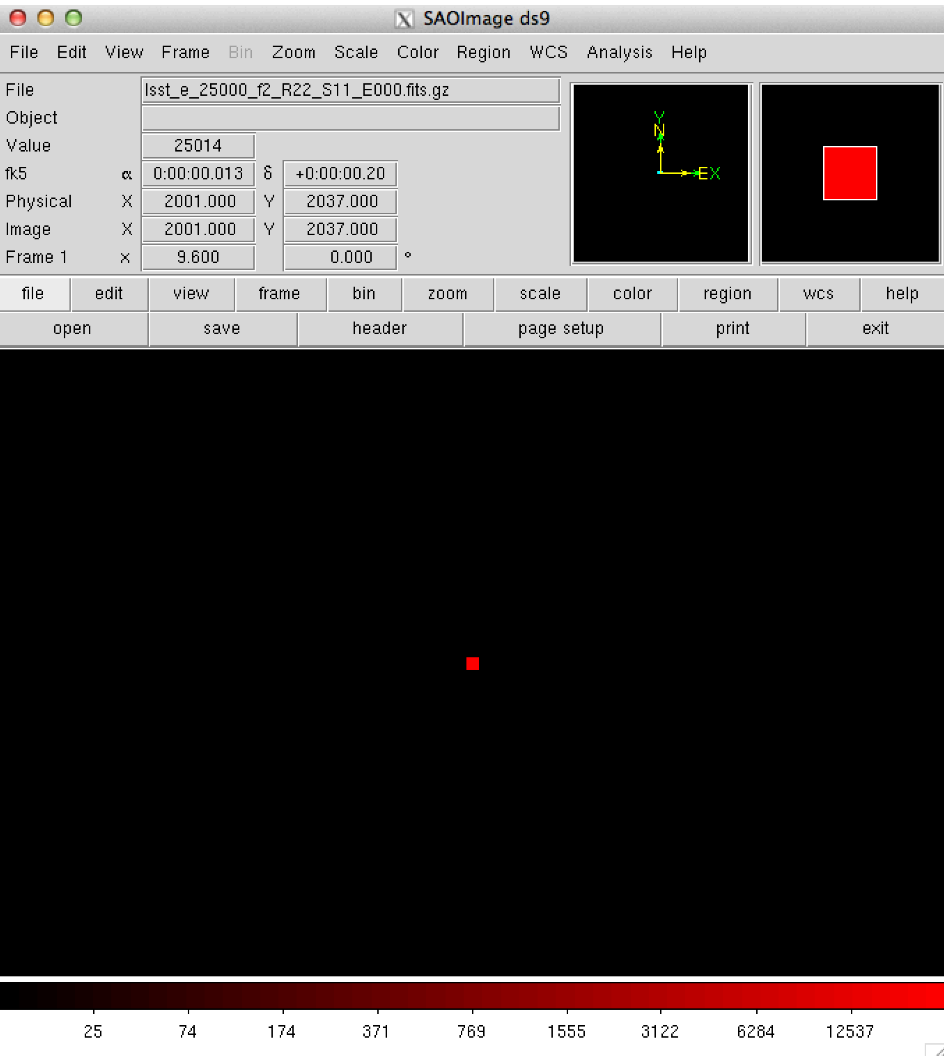
m1 = 2.5 log_10(Num_e/507967) - 20

Which gives

# electrons	Magnitude
5000	25.017
10000	24.264
15000	23.824
20000	23.512
25000	23.269
30000	23.072
Etc.	

Calibrated charge sharing in 3.3.2

For 25,000 electrons shown with and w/o the charge sharing



3.3.2 Algorithm

Note: model only works if saturation and blooming are turned on since this effect is simulated in the saturation code.

```
    rebloom:;
        *(focal_plane+location)+=leftover;
        if (*(focal_plane+location) > well_depth) {
            leftover=(long) (*(focal_plane+location)-well_depth);
            *(focal_plane+location)=(float)well_depth;
            if (blooming==1) {
                if (bloom(1)) goto fullysat;
                location=nampx*(yPos-miny)+(xPos-minx);
                goto rebloom;
            }
        }
        if (leftover==1 && blooming==1 && chargesharing==1) {
            if (RngDouble() < 0.04*2.0/((double)well_depth)*(*(focal_plane+location))) {
                *(focal_plane+location)-=1;
                if (bloom(0)) goto fullysat;
                location=nampx*(yPos-miny)+(xPos-minx);
                *(focal_plane+location)+=1;
            }
        }

    fullysat:;

        if (*(focal_plane+origlocation) >= well_depth) {

            if (ghostFlag==0 && sources.spatialtype[source]!=4) {
                minrad=(long)fabs((largeAngle->y)/DEGREE*platescale/pixsize)-4;
                // minrad=(long)fabs(deltaY)-4;
                if (minrad == (long)(fabs(sourceSaturationRadius)+1)) {
                    sourceSaturationRadius=(double)minrad;
                }
            }
        }
```

3.3.2 Algorithm continued

```
if (leftover==1 && blooming==1 && chargesharing==1) {
    if (RngDouble() < 0.04*2.0/((double)well_depth)*(*(focal_plane+location))) {
        *(focal_plane+location)-=1;
        if (bloom(0)) goto fullysat;
        location=nampx*(yPos-miny)+(xPos-minx);
        *(focal_plane+location)+=1;
    }
}
```

For these photons divert:

0.04*2.0*(contents/well_depth)

of them to another pixel.

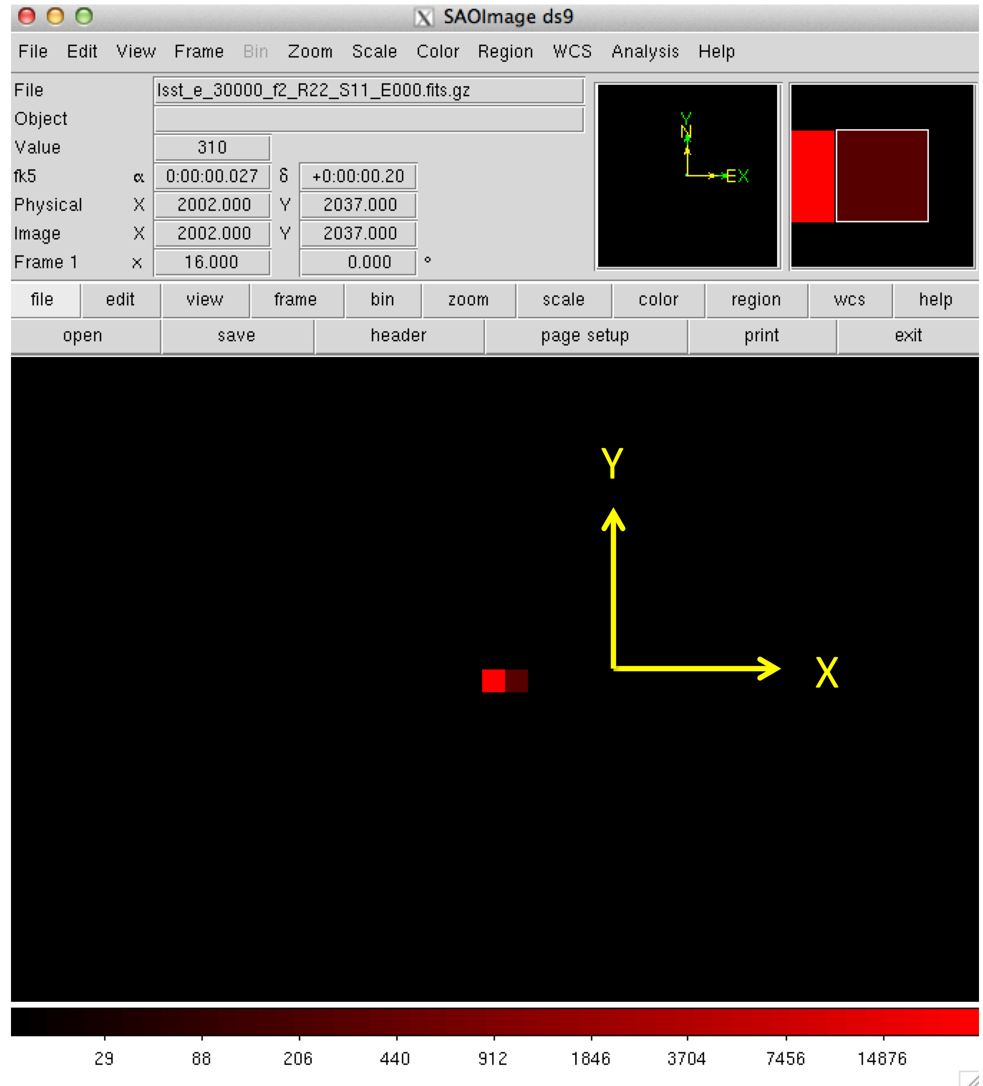
= 4% * 2 * (current fraction of well depth)

At the end of the process the average diverted will be approximately ½ of this (Hence the 2).

Charge Sharing Algorithm

```
# Set pixel depth and parameters
# well_depth 1000000
blooming 1
saturation 1

# Turn on/off the sharing to other pixels
chargesharing 1
```



For "30,000":
310 moved 29681 left

→ 1.03 % moved

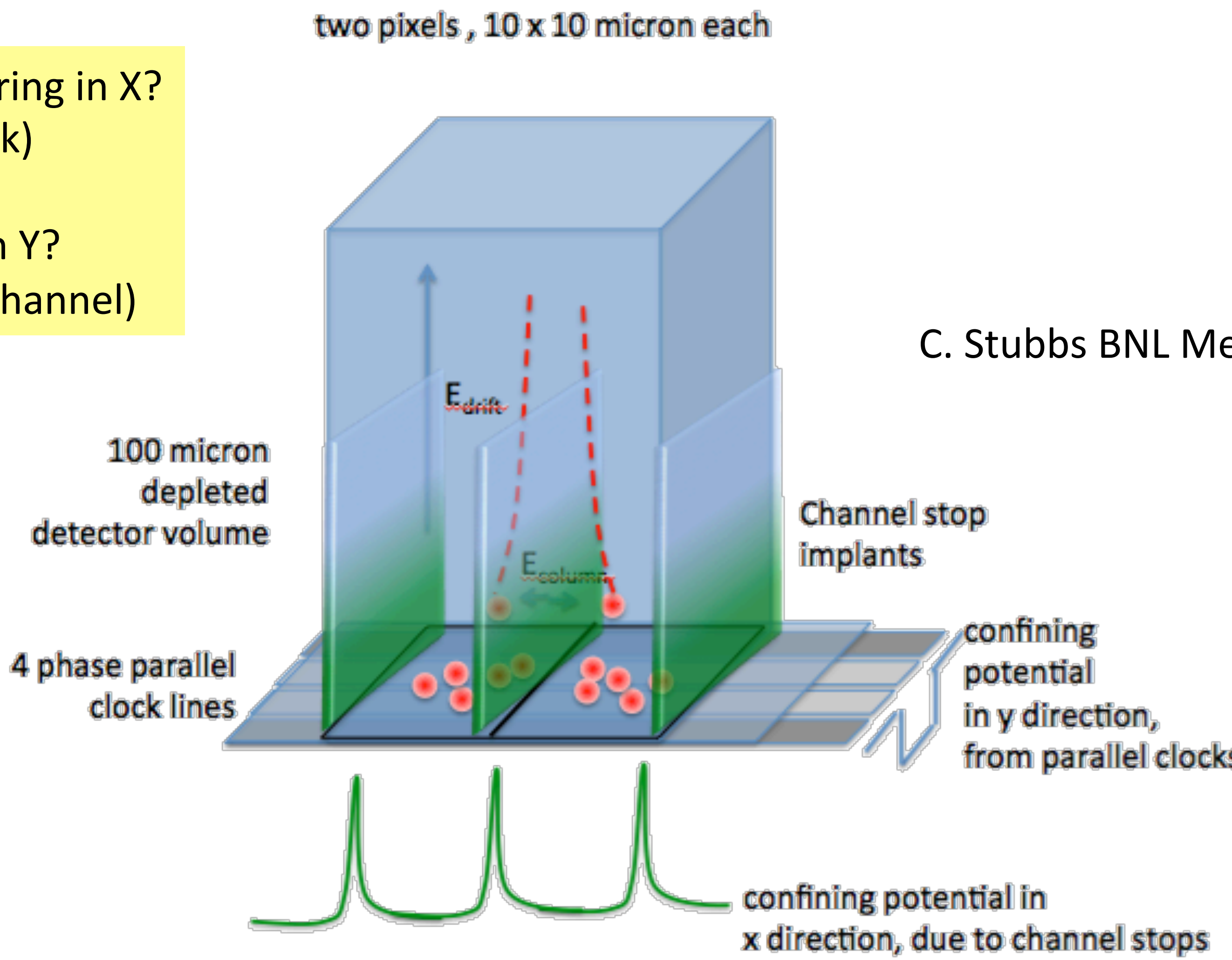
We expect 30000/100000*.04 = 1.2%

Note: only moves in X.

Expected charge movement

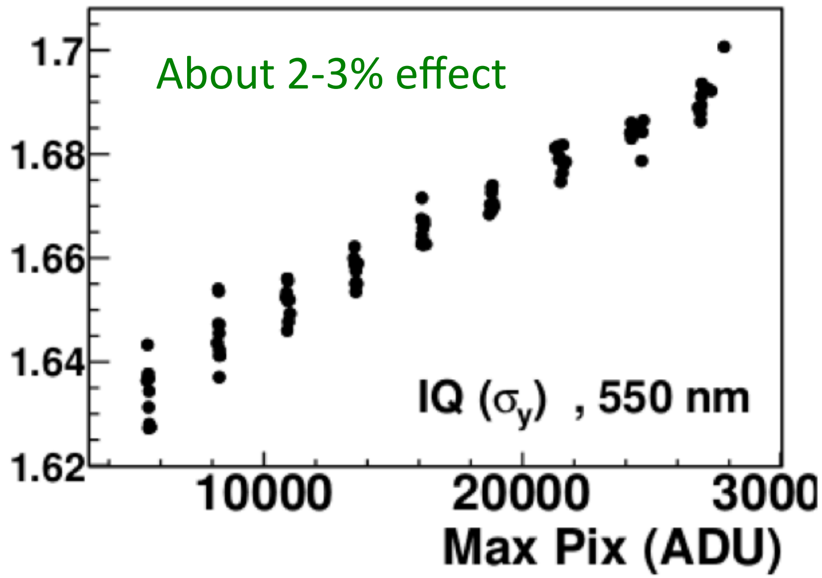
Charge sharing in X?
(Drift in bulk)

Spill-over in Y?
(in buried channel)



C. Stubbs BNL Meeting

Now: want to measure the sigma as a function of electrons



I wrote a short DM program to analyze the output.

The SDSS Shape algorithm failed on my reference 1 pixel “Peak”.

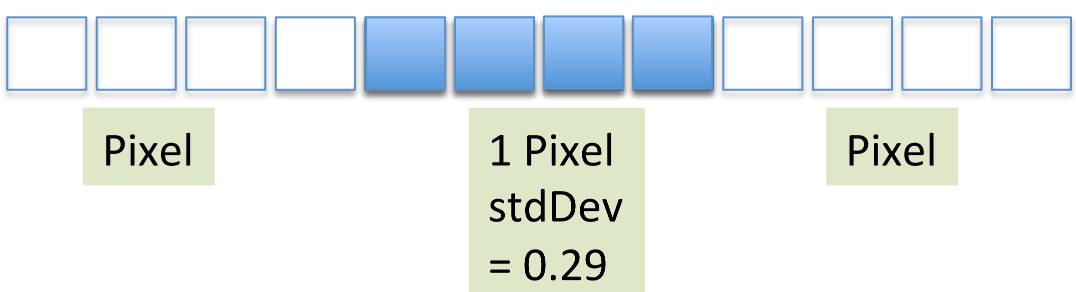
Remember: Variance of a square is 1/12 = .083
stdDev = sqrt(1/12) = 0.29

Calculating the StdDev

For now I calculated the variance along the x and y column of the source. I made each pixel 10 "sub-pixels" in order to be able to calculate a width (only for the calculation).

→ <u>In future use this to probe the model of pixel edge effects</u>.

X Case:



Same for Y:

Results
v3.3.2 “perfect” spot

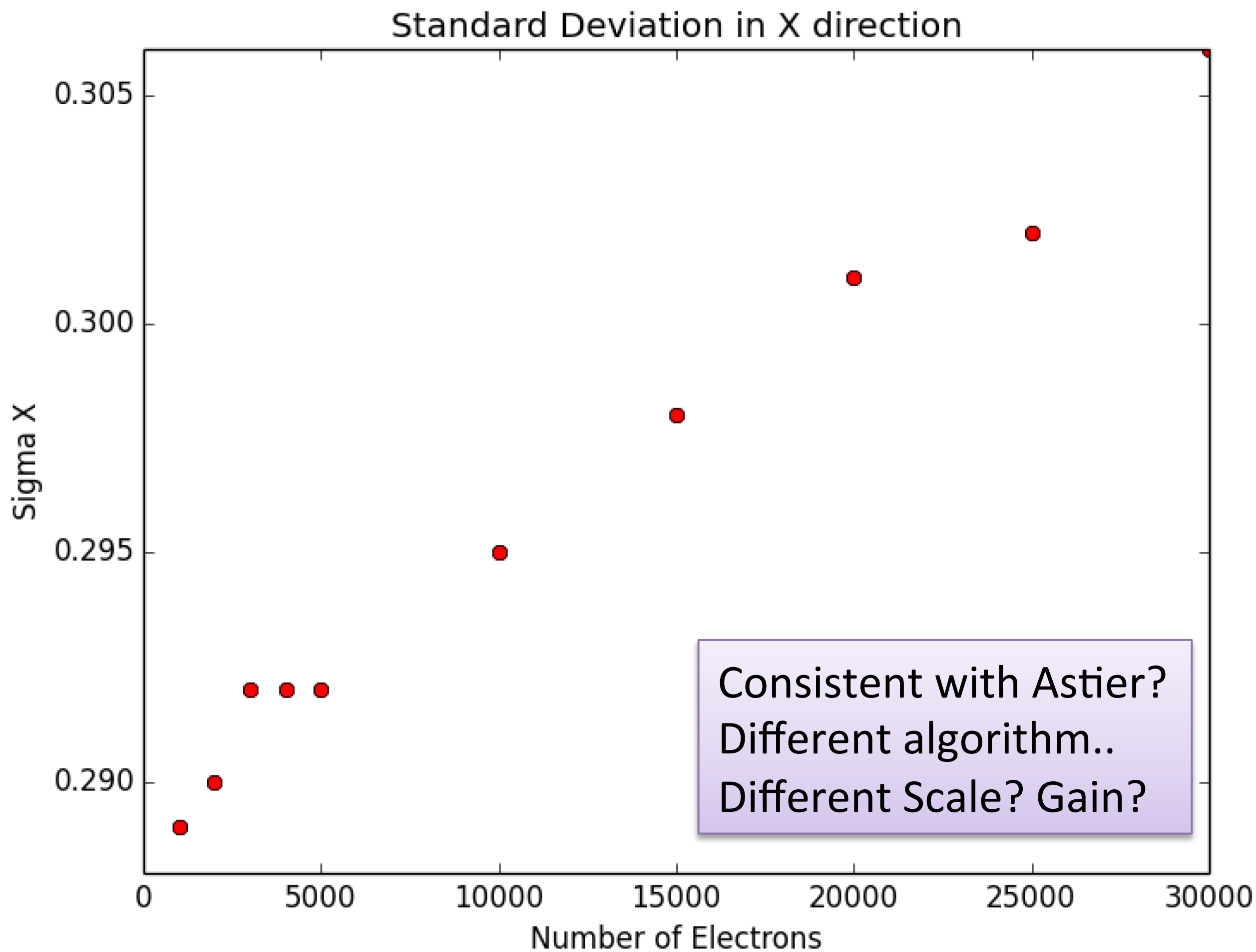
```
Electrons = 1000 STDX = 0.289 STDY = 0.289
Electrons = 2000 STDX = 0.289 STDY = 0.289
Electrons = 3000 STDX = 0.289 STDY = 0.289
Electrons = 4000 STDX = 0.289 STDY = 0.289
Electrons = 5000 STDX = 0.289 STDY = 0.289
Electrons = 10000 STDX = 0.289 STDY = 0.289
Electrons = 15000 STDX = 0.289 STDY = 0.289
Electrons = 20000 STDX = 0.289 STDY = 0.289
Electrons = 25000 STDX = 0.289 STDY = 0.289
Electrons = 30000 STDX = 0.289 STDY = 0.289
```

No Charge Sharing:
All “1 pixel”

```
Electrons = 1000 STDX = 0.289 STDY = 0.289
Electrons = 2000 STDX = 0.290 STDY = 0.289
Electrons = 3000 STDX = 0.292 STDY = 0.289
Electrons = 4000 STDX = 0.292 STDY = 0.289
Electrons = 5000 STDX = 0.292 STDY = 0.289
Electrons = 10000 STDX = 0.295 STDY = 0.289
Electrons = 15000 STDX = 0.298 STDY = 0.289
Electrons = 20000 STDX = 0.301 STDY = 0.289
Electrons = 25000 STDX = 0.302 STDY = 0.289
Electrons = 30000 STDX = 0.306 STDY = 0.289
```

Charge Sharing:
X direction increasing

Width increase about ~6%



Make flats to check for auto-correlations

```
-- Submitter: condor.internal.phy.duke.edu : <10.136.82.10:9129> : condor.intern
al.phy.duke.edu
 ID       OWNER            SUBMITTED     RUN_TIME HOST(S)
413146.0   cwalter          2/8  14:47   1+00:06:03 slot1@qgp06
condor:brighter-fatter $
```

I tried again to turn off all effects and then generate a flat with

SIM_TELCONFIG 2

But clearly it didn't work…. It is taking forever.

Is there a fast way to generate a flat with no detector effects?

Aside: Phosim enviroment

- Not yet so optimized for running in a centralized installed model with many simultaneous runs in condor etc (or maybe I am missing options..)

Condor example script:

```
Initialdir        = $(phosimdir)
Executable      = $(phosimdir)/phosim
Arguments      = $(workdir)/$(SOURCE_FILE) -c $(workdir)/perfect_seeing -w $(workdir)/work -o $(workdir)/output
Output        = $(workdir)/logs/$(SOURCE_FILE).log
Error        = $(workdir)/logs/$(SOURCE_FILE).log

SOURCE_FILE = 1000e
Queue
..
```

What would be good would be to have a $PHOSIMDIR environment variable that would be used for the installation and then default to local directories for run files.
Also: autogenerated output directories for file collision reduction.

Next Steps

- Generate flats and calculate auto-correlations
- Explore Andy R's model in the trunk
- Try to implement model from Astier?
- Improve measurement algorithm/variable
- Compare w/real data
  - Make a "realistic" spot including size and noise.
  - Explore sub-pixel edge effects.

Conclusions

- Framework for exploring sensor effects with a perfect spot now running including simulation and analysis.
- Have done simple simulations and DM analysis of the BF effect in 3.3.2
- Will expand analysis and models next.