Integration of LSST stack elements on DESDM

Felipe Menanteau
NCSA, University of Illinois
The Problem
The Problem
The Problem

We need robust (clean) catalogs to achieve the Dark Energy Science goals. However,

There are still several masking issues on single-epoch DECam images from. These include:

• Streaks/trails form satellites
• Cosmic Rays
• Bleed Trails from bright stars
• Scattered light
The Problem

We need robust (clean) catalogs to achieve the Dark Energy Science goals. However,

There are still several masking issues on single-epoch DECam images from. These include:

• Streaks/trails form satellites

• Cosmic Rays

• Bleed Trails from bright stars

• Scattered light
The Problem

We need robust (clean) catalogs to achieve the Dark Energy Science goals. However,

There are still several masking issues on single-epoch DECam images from. These include:

• Streaks/trails form satellites
• Cosmic Rays
• Bleed Trails from bright stars
• Scattered light
The Problem

We need robust (clean) catalogs to achieve the Dark Energy Science goals. However,

There are still several masking issues on single-epoch DECam images from. These include:

• Streaks/trails form satellites
• Cosmic Rays
• Bleed Trails from bright stars
• Scattered light
Reinventing the wheel. Knowing when and when not!
Reinventing the wheel.
Knowing *when* and *when not*!
1. Satellite Trails

- Occasionally (~6% of CCDs in single epoch exposures) have bright objects streaking across them (satellites, meteors, etc.)

- Streaks can impact photometry in both single epoch and co-added images.
1. Satellite Trails

- Occasionally (~6% of CCDs in single epoch exposures) have bright objects streaking across them (satellites, meteors, etc.)
- Streaks can impact photometry in both single epoch and co-added images.
The Solution
(by Alex Drlica-Wagner and Eli Rykoff)

Pre-Processing

1. Subtract background
2. Apply existing mask
3. Bin image (speed up)
4. Estimate sky-noise level
5. Threshold based on sky noise
The Solution
(by Alex Drlica-Wagner and Eli Rykoff)

**Pre-Processing**

1. Subtract background
2. Apply existing mask
3. Bin image (speed up)
4. Estimate sky-noise level
5. Threshold based on sky noise
Hough Transform

(Eli Rykoff’s pyhough  http://github.com/erykoff/pyhough)

- Iterate through each pixel of the thresholded image and count how many pixels lie at each possible angle
- Create a 2D histogram in “Hough-space” where lines accumulate as localized over-densities
Hough Transform

(Eli Rykoff’s pyhough  http://github.com/erykoff/pyhough)

• Iterate through each pixel of the thresholded image and count how many pixels lie at each possible angle

• Create a 2D histogram in “Hough-space” where lines accumulate as localized over-densities
Get angle and distance to define a line

• Iterate through each pixel of the thresholded image and count how many pixels lie at each possible angle

• Create a 2D histogram in “Hough-space” where lines accumulate as localized overdensities

Hough Transform
(Eli Rykoff’s pyhough http://github.com/erykoff/pyhough)
Hough Transform

(Eli Rykoff’s pyhough  http://github.com/erykoff/pyhough)

- Iterate through each pixel of the thresholded image and count how many pixels lie at each possible angle
- Create a 2D histogram in “Hough-space” where lines accumulate as localized over-densities

<table>
<thead>
<tr>
<th>Angle (deg)</th>
<th>Dist. (pix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>30</td>
<td>69.6</td>
</tr>
<tr>
<td>60</td>
<td>81.2</td>
</tr>
<tr>
<td>90</td>
<td>70</td>
</tr>
<tr>
<td>120</td>
<td>40.6</td>
</tr>
<tr>
<td>150</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Angle (deg)</th>
<th>Dist. (pix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>57.1</td>
</tr>
<tr>
<td>30</td>
<td>79.5</td>
</tr>
<tr>
<td>60</td>
<td>80.5</td>
</tr>
<tr>
<td>90</td>
<td>60</td>
</tr>
<tr>
<td>120</td>
<td>23.4</td>
</tr>
<tr>
<td>150</td>
<td>-19.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Angle (deg)</th>
<th>Dist. (pix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>74.6</td>
</tr>
<tr>
<td>30</td>
<td>89.8</td>
</tr>
<tr>
<td>60</td>
<td>80.6</td>
</tr>
<tr>
<td>90</td>
<td>50</td>
</tr>
<tr>
<td>120</td>
<td>5.0</td>
</tr>
<tr>
<td>150</td>
<td>-39.6</td>
</tr>
</tbody>
</table>

Hough Transform Diagram
Detection and Masking of Streaks
Detection and Masking of Streaks

Performance:
~7 sec on 2.4 Ghz Core i7 MacBook Pro (includes I/O time)
2. Cosmic Rays

- Cosmic Rays have long and curly shapes on DECam CCDs (and likely on LSST)
- Present in virtually all images
- Particularly problem in long SN exposures
- Easy to mask on multi-epoch coadds, need a single-epoch application
- Currently DESDM has an implementation, but only work for bad seeing >0.8"

[Image of Cosmic Rays]
Finds Cosmic Rays on a Science Image using: \texttt{lsst.meas.algorithms.findCosmicRays}

The algorithm searches for all pixels which satisfy a series of conditions (from Photo-Lite in prep.):

1. That the candidate bad pixel \( p \) not be adjacent to a saturated pixel.

2. That \( p \)'s intensity \( I \) exceed the locally-determined background by \( n\sigma \) \((n=6)\) where \( \sigma^2 \) is the sky variance.

3. We require that no pixel \( I \) be part of a peak which is sharper than the centre of a star centered in a pixel. This condition becomes:

\[
I - c \ast N(I) > P(d)(\bar{I} + cN(\bar{I}))
\]

- \( I \) pixel Intensity
- \( c=3 \) constant
- \( N(I) \) std-deviation of \( I \)
- \( P(d) \) PSF at distance \( d \)
- \( \bar{I} \) average of two pixels a distance \( d \) away

4. Conditions applied sequentially to the pixel in using the four pairs of neighboring pixels (NS, EW, NW-SE, and NE-SW, \( d=1, \sqrt{2}, \sqrt{2} \) ). The candidate cosmic ray must exceed condition 2 for all four pairs of neighbors, and condition 3 for at least one pair.
The Call

#!/usr/bin/env python
import immask

args   = immask.cmdline()
desobj = immask.DESIMA(args.fileName,args.outName)

# CR/Streak Masking
desobj.CRs(**args.__dict__)
desobj.mask_streaks(**args.__dict__)
desobj.write(compress=args.compress)

%>setup immask
%>immask DECam_00226647_47.fits.fz clean_DECam_00226647_47.fits.fz
   --bkgfile DECam_00226647_47_bkg.fits.fz
   --compress

Performance:
~5 sec on 2.4 Ghz Core i7 MacBook Pro (includes I/O time)
We married Streak Finder with LSST/CRfinder into: Immask
DECam de-trended image
DECam de-trended image

Cosmic Rays
Immask CR and Streak image
DECam de-trended mask plane
Immask CR and Streak image

Satellite Trail
Immask CR and Streak image

Cosmic Rays

Satellite Trail
Final Remarks

- Immask should be included in DESDM production for Y1A1 (summer 2014)

- Plans to include DESDM bleedtrail algorithm into Immask.

- Immask is an eups (DESDM) deliverable package, i.e.: %>eups distrib install immask 0.1.1