Computational Astrophotography (in near real time)

Alternate title 1: "how to do astrophotography processing with python"

Alternate title 2: "how to approach astrophotography with a scientific imaging approach"

Alternate title 3: "how to automate and accelerate astrophotography processing so that it can be done at the same rate the data is arriving"

Alternate title 4: "how to push the limits of EAA with python and a GPU"

Daniel Champion

Introductions and Outline

- Thank You!

  Its an honor to have the opportunity to speak today about this fascinating and exciting topic.

- About Me:
  - 2002-2012: started with Astrophotography (film, DSLR, all the usual tools), then moved to Bortle 9 skies on the east coast near D.C. 😱
  - …Became a scientist and industry mathematician… (Govt., Bio-Tech, …) performing scientific imaging, data science, math, physics.
  - ~3years ago resumed astrophotography, blown away with the advances in software and automation.
  - …unsatisfied that processing was still predominantly done with manual processing tools/GUIs. Started work on AstroComPYute!

Lagoon & Trifid nebula, ~2008

100's hours

TAIC Inspiration:

OUTLINE

1. **Introducing: AstroComPYute**
2. **Computational astrophotography, software, GPUs, workflows,…**
3. **Key components of a computational astrophotography approach**
4. **Examples**
5. **Deep dive on gradient estimation/removal using AstroComPYute**
6. **Demonstration**
7. **References, where to get stuff**

The Vision and Challenge

- Imagine EAA taken to the furthest extent….
- Hold up a device pointing towards any part of the sky, and through it you can view in real-time a finished astrophotography grade image.

- **Underlying physics and math behind astrophotography:**
  - **Consider a single pixel in a camera, and the arriving photons that were emitted from faint extended nebula (ex: Barnard's Loop, Rosette nebula), @ 1-second exposure.**
  - **The arrival rate of photons, μ, can be in the single digits.**
  - **The random arrival time of the photons results in noisy observed photon detections, X**
    - Poisson process results in the observations X.
  - **A fundamental property of Poisson processes is that $var(X) = \mu$, $\sigma_X = \sqrt{\mu}$, thus the relative uncertainty in our ability to estimate μ using X is**

$$rel.uncertainty = r = \frac{\sqrt{\mu}}{\mu} = \frac{1}{\sqrt{\mu}}$$

  - **After averaging N sub frames, uncertainty decreases approx. according to the standard error of the mean:**

$$r_N \approx \frac{r}{\sqrt{N}} = \frac{1}{\sqrt{N} \cdot \sqrt{\mu}}$$

  - ***This is the uncertainty in the linear state, stretching can amplify this uncertainty!***

The Vision and Challenge

- Imagine EAA taken to the furthest extent….
- Hold up a device pointing towards any part of the sky, and through it you can view in real-time a finished astrophotography grade image.

$$r_N = \frac{r}{\sqrt{N}} = \frac{\sqrt{\mu}}{\sqrt{N} \cdot \mu}$$

- ***What can we do about r, r_N?***
- ***We need stretched uncertainty less than ~0.01 for good images***
  - ***Increase N:***
    - ***<u>More sub frames</u> – Longer data acquisition, parallel imaging***
    - ***<u>Use nearby pixels</u> – Computation (noise reduction, star separation,…), binning,***
  - ***Increase μ:***
    - ***<u>Fast optics</u> (at equivalent focal length)***
    - ***<u>Longer eposures</u> – Longer data acquisition***

Can combined with compute and heterogenous data acq. to mitigate clipping

- ***<u>Detectors</u>***: we are already at 80+% QE,
- ***<u>Optics</u>***: Need f/ratios <2, ideally < 1 to move the needle here
  - Option: Dragonfly telescope strategy https://www.dragonflytelescope.org/
- All is not lost: **<u>Fast optics</u> + <u>modern detector</u> + <u>instrument characterization</u> + gargantuan amount of <u>compute</u> MIGHT get us close!**
- **We are going to explore this with what can be done with tech of today.**
- …Also need to address all of the non-automated and processing operations in astrophotogrphy.

AstroComPYute: Overview

- AstroComPYute is a Python + CUDA + OpenCV based astrophotography processing toolkit
  - Its not a GUI, you write a script that preforms a reproducible processing workflow
- It can run as fast* as the data is arriving (usually),
  - it can run in a live-stacking manner or after-the-fact
- Built to produce near-real-time fully processed astrophotography products
- Built to reproduce a full processing workflow, so that you can mod, add, sub, tweak the process and easily compare the results.

- Requirements:
  - Hardware: Nivida GPU and a late model CPU
  - Core software: Python, OpenCV, CUDA Toolkit 11+
    - numpy, scipy, scikit-image, astropy,
  - Helper software: ASTAP, Starnet++
  - Reference materials from NASA, WHAM

- Telescope/Optics/Mount/Camera/Acq.:
  - Star Tracker or most any mount, typically very fast optics (f/2.0 or faster),
  - Modern cooled Astro camera or DSLR
  - Data Acq software (whatever you like, e.g. N.I.N.A.)

GitHub github.com/Daniel-Champion/AstroComPYute

* 10 second exposures or longer on a 26 MP imager, but your results may vary

AstroComPYute and computational astrophotography strategy, goals, themes…

- **Scientific imaging approach**: reproducible, scripted, procedural, astrophotography processing sequence.
- Leverage **open source software** (OSS) solutions for astrophotography processing
- **Solve** as many **hardware problems with software** and computing as we can
- When possible solve astrophotography challenges with **a-priori apparatus or data characterization** (instrument and camera and target).

Astrophotography processing is hard: Modern compute hardware compared to compute requirements for astro processing

FLOP = floating point operation (fp32)
1 GFLOPs = 1 billion floating point operations per second

Hardware (compute and astro):

- Modern CPU: 100-200 GFLOPs/processor
- Modern GPU: 10000-83000 GFLOPs/device
- Modern APS or full-frame astro cameras: 20e6+ pixels @ 3 channels?

Processing:

- Simple processing steps: 1→10 GFLOP
  - Usually sub-second processing on a single core (~10-100 FLOP/pixel/channel)
  - FF/DF correction, stretching, color correction, HSV adjustments
- Complex processing steps 100→1000 GFLOP
  - Processing time in the minutes (~1000-10000 FLOP/pixel/channel)
  - Registration, noise reduction filters, star locations, gradient estimation.
- Total compute for the above: $\mathbf{1000} \rightarrow \mathbf{10000} + \quad \boldsymbol{GFLOP}$

CPU performance has doubled in 4 years

GPU performance has more than doubled in 4 years, 3x or 4x by end of year.

If we can make use of a fraction of the compute available on the GPU to perform the complex processing steps we can keep processing time in the seconds.

Python image processing and astronomy resources

- Within the python ecosystem (or accessible to it) there is a vast amount of leading-edge capabilities that can assist image and astronomy related processing.
- Some of these projects contain processing capabilities well beyond those contained in any GUI astrophotography application
- AI tools and AI-based applications are the exception, these seem to hit proprietary and higher-level processing environments first (for now).
- These are the tools the scientific imaging community uses (also MATLAB).
- 1000's of contemporary developers and contributors across these libraries.

python

NumPy

SciPy

scikit-image
image processing in python

OpenCV

matplotlib

TensorFlow

PyTorch

CuPy

NVIDIA CUDA

Numba

ASTROALIGN

The Astropy Project

Python image processing and astronomy resources

- What these libraries have in common:
  - Perform an operation, do it well, make its outputs usable by downstream operations.

python

NumPy

SciPy

scikit-image
image processing in python

OpenCV

matplotlib

TensorFlow

PyTorch

CuPy

NVIDIA CUDA

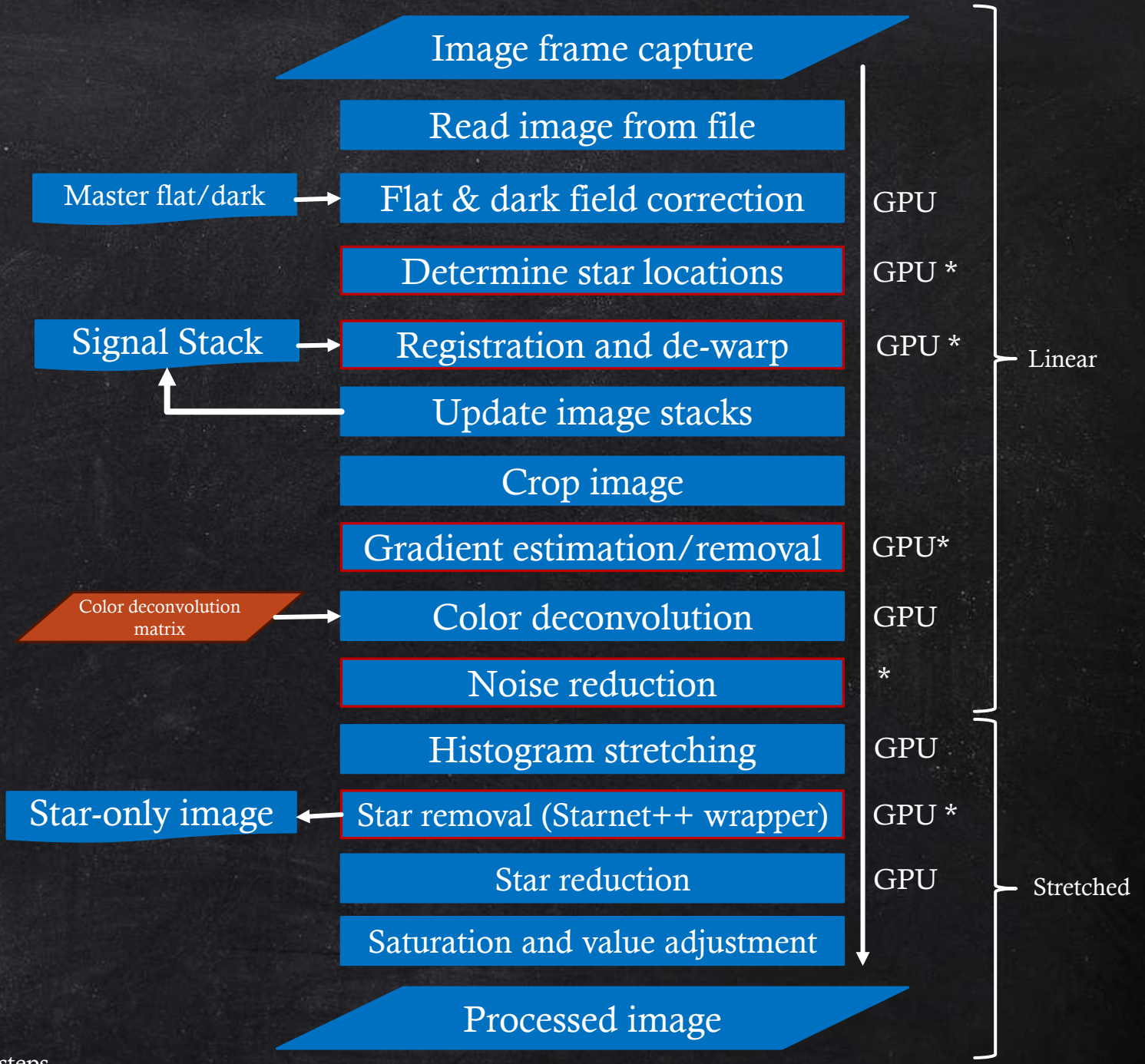
Numba

ASTROALIGN

The Astropy Project

Example astrophotography workflow:

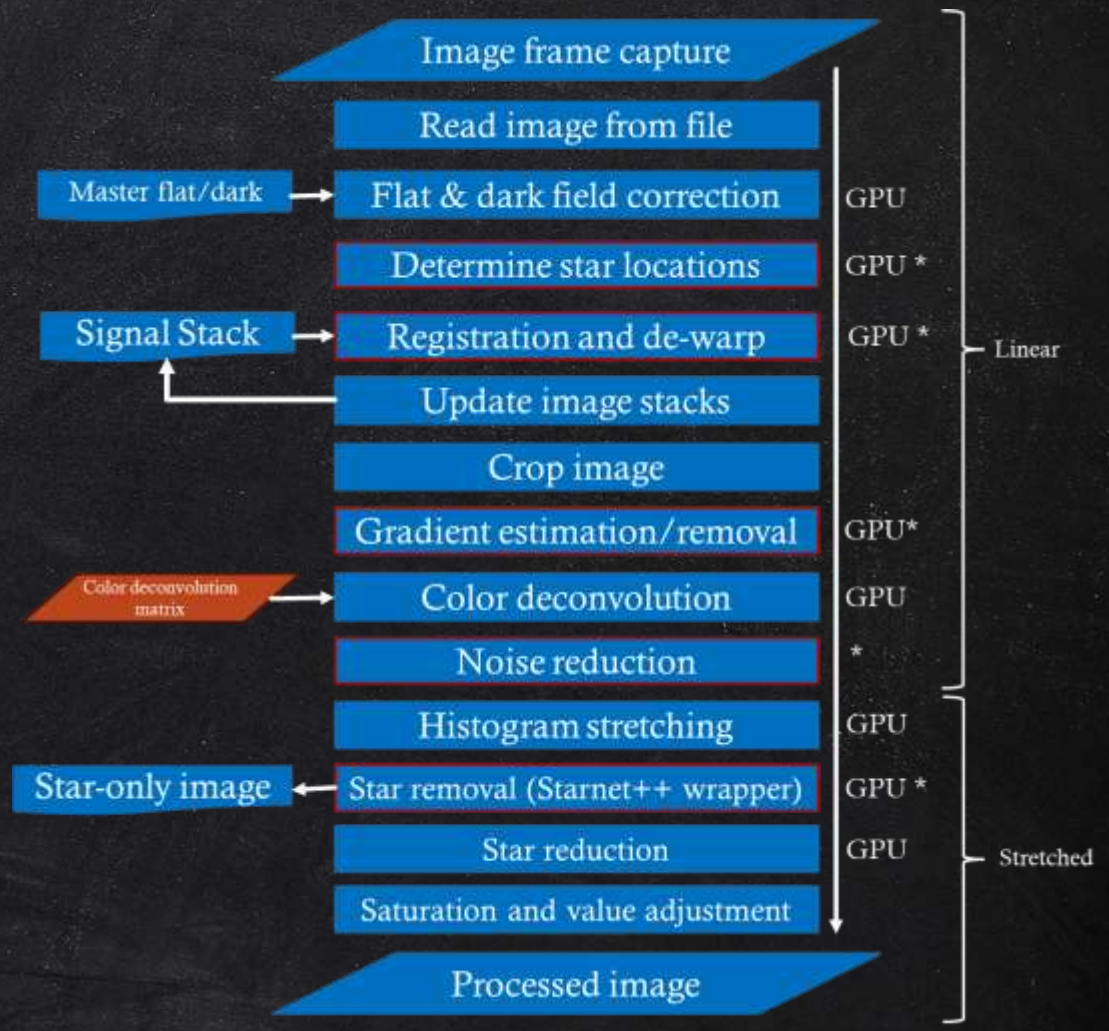
- The workflow at right is one possible end-to-end processing path that results in a fully-processed astrophotography product.
- All of the processing steps at right have been implemented in either CPU or GPU versions using python functions or array arithmetic.
- In EAA workflows it is common to omit gradient removal, color correction, and star separation related processing due to the computational cost.
  - Noise reduction?



* = larger computational steps

Unique/critical components of the computational astrophotography approach:

1. **Plate Solving [ASTAP CLI]**
   - Determines the celestial coordinates of the image (every pixel)
   - Also provides annotation services to further identify Deep Sky Objects (DSOs) in the FOV.
   - Python process converts ASTAP tangent plane coordinates to spherical coordinates (typically < 20" error across the FOV)
2. **Interpolation of the NASA Deep Star Map at camera image coordinates.**
   - Informs gradient & background estimation/removal
3. **Interpolation of the H-alpha full sky map at camera image coordinates.**
   - Informs gradient & background estimation/removal
4. **Pre-computed color calibration**
   - Determined using daytime experiments
5. **Fast noise reduction**
   - Assists histogram stretching, star reduction/separation
6. **Starnet++ python wrapper**



Unique/critical components of the computational astrophotography approach:

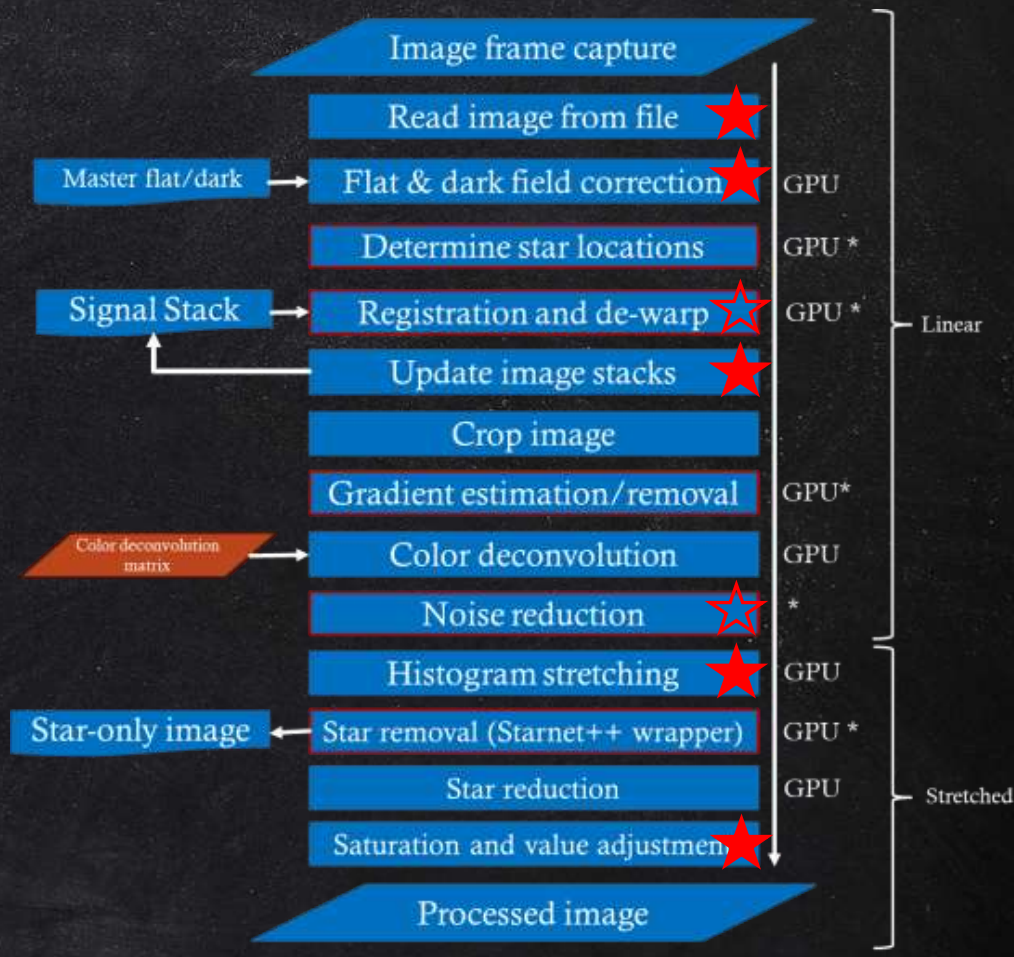
SharpCap PRO

Here's a list of the great features that you get when you upgrade from the free version of SharpCap to SharpCap Pro.

- Polar Alignment Adjustment Guidance
- **Planetary Live Stacking** – includes live stacking and wavelet sharpening, also works for solar and lunar images.
- **Live image processing** – Flat frame correction, Dark subtraction, Background subtraction, Banding suppression and Satellite trail removal
- **Advanced Live Stacking** – Sigma clipped stacking, comet alignment, automatic dithering, colour balance adjustments, sharpening and noise reduction
- **Automated Imaging** – Deep Sky sequence planner, Lunar/Solar Mosaic planner, Solar System sequence planner, advanced sequence editor, seeing triggered captures
- **Focus Assistance** – Automated focus scans, fully automatic refocus
- **Target finding and tracking** – Click to Recenter, Feature Tracking, custom deep sky catalogs, Push-To assistance
- **Usability** – Image auto-stretch, Two monitor view
- **Camera analysis and settings guidance** – Custom deep sky exposure setting guidance, export data from Sensor Analysis,
- Other – ADC alignment from image features, live stellar photometry estimation, Python scripting language, increased memory usage limits

- SharpCap is great,easy, accessible,…
- AstroComPYute is not a SharpCap competitor, its for Pythonistas who want to AUTOMATE ALL THE THINGS
- Difference is in the processing steps these tools perform and have available to them, and in the approach:

  AstroComPYute is about scripting astro processing and leveraging all of the image processing libraries and astro tools out there
- AstroComPYute is not alone, check out David Strickland's AstroPhotography project on GitHub, and https://yager.io/Astro.html



Examples of Processed Images

1000s

- 100 sub-frames
- 10s exposure each frame
- Unguided

- 105mm F1.4 Sigma Art lens
- ZWO ASI2600MC Pro
- Data Acq.: N.I.N.A.
- Processing: python

- Ocean View HI, Bortle 1-2

- Zodiacal lights were “annoying”

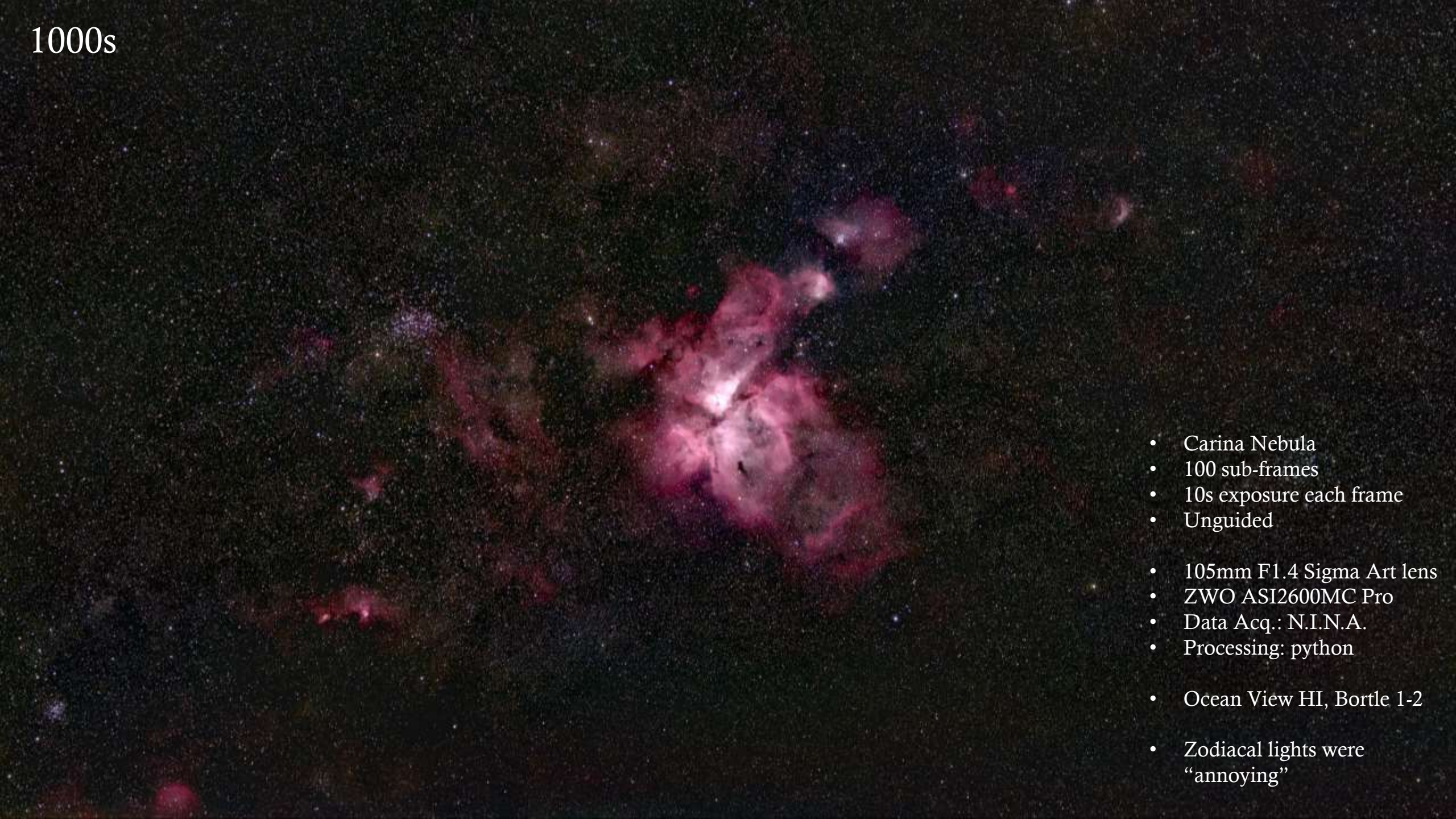
100 sub-frames
10s exposure each frame
Unguided

105mm F1.4 Sigma Art lens
ZWO ASI2600MC Pro
Data Acq.: N.I.N.A.
Processing: python

Ocean View HI, Bortle 1-2

Zodiacal lights were
“annoying”

1000s



- Carina Nebula
- 100 sub-frames
- 10s exposure each frame
- Unguided

- 105mm F1.4 Sigma Art lens
- ZWO ASI2600MC Pro
- Data Acq.: N.I.N.A.
- Processing: python

- Ocean View HI, Bortle 1-2

- Zodiacal lights were “annoying”

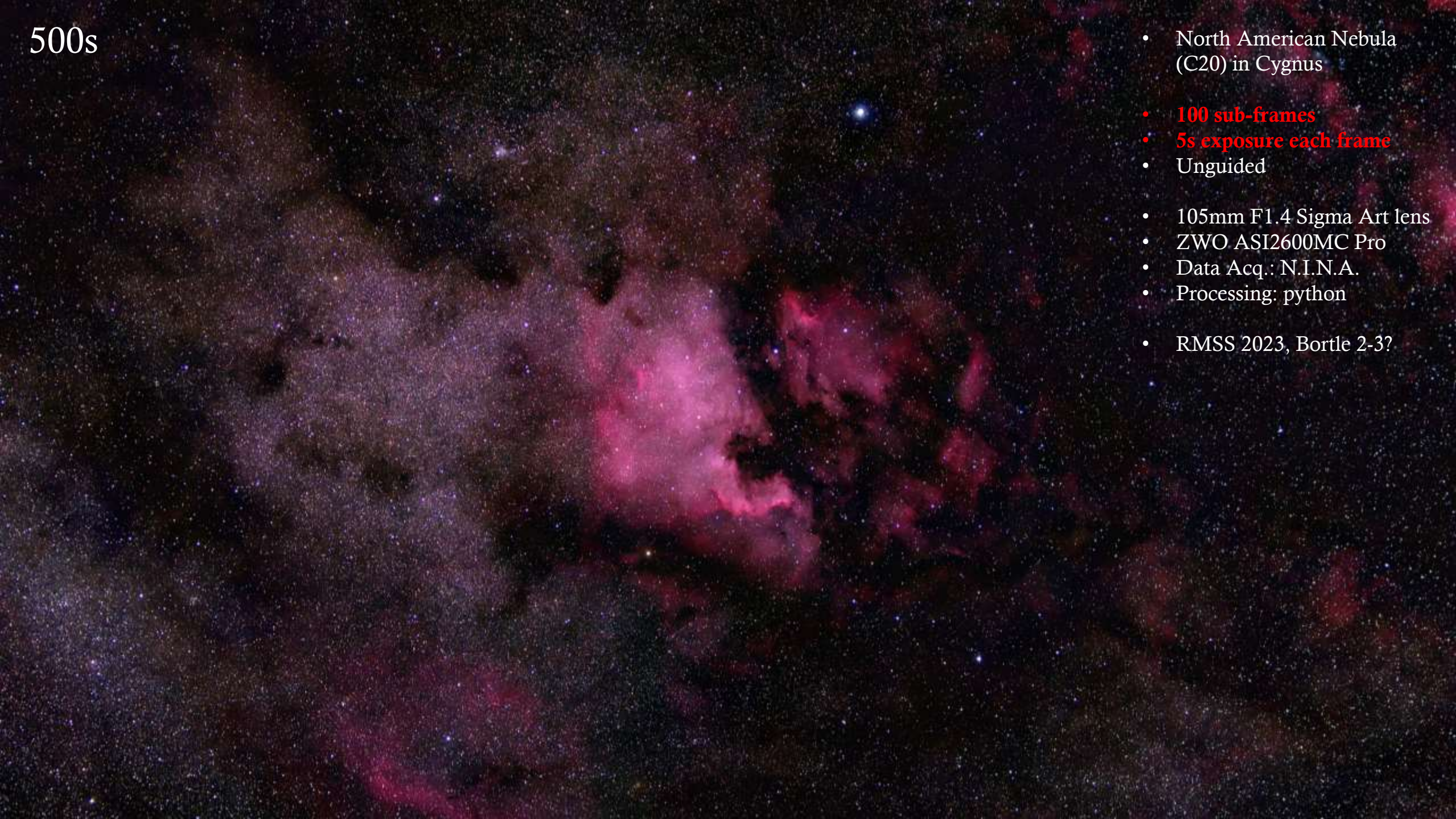
1000s

- Heart and Soul Nebula, with double cluster (Perseus)

- 200 sub-frames
- 5s exposure each frame
- Unguided

- 105mm F1.4 Sigma Art lens
- ZWO ASI2600MC Pro
- Data Acq.: N.I.N.A.
- Processing: python

- Lake Mead, NV, Bortle 4?



- North American Nebula (C20) in Cygnus

- **100 sub-frames**
- **5s exposure each frame**
- Unguided

- 105mm F1.4 Sigma Art lens
- ZWO ASI2600MC Pro
- Data Acq.: N.I.N.A.
- Processing: python

- RMSS 2023, Bortle 2-3?

- Dark Shark
- (LDN 1235, Cepheus)

- **100 sub-frames**
- **5s exposure each frame**
- Unguided

- 105mm F1.4 Sigma Art lens
- ZWO ASI2600MC Pro
- Data Acq.: N.I.N.A.
- Processing: python

- Lake Mead, NV, Bortle 4?

- Rho Ophiuchi cloud complex

- **50 sub-frames**
- **5s exposure each frame**
- Unguided

- 105mm F1.4 Sigma Art lens
- ZWO ASI2600MC Pro
- Data Acq.: N.I.N.A.
- Processing: python

- RMSS 2023, Bortle 2-3?

- Eagle+Swan+Sag. S.C.+Trifid+Lagoon Ensemble

- 26 sub-frames
- 5s exposure each frame
- Unguided

- 105mm F1.4 Sigma Art lens
- ZWO ASI2600MC Pro
- Data Acq.: N.I.N.A.
- Processing: python

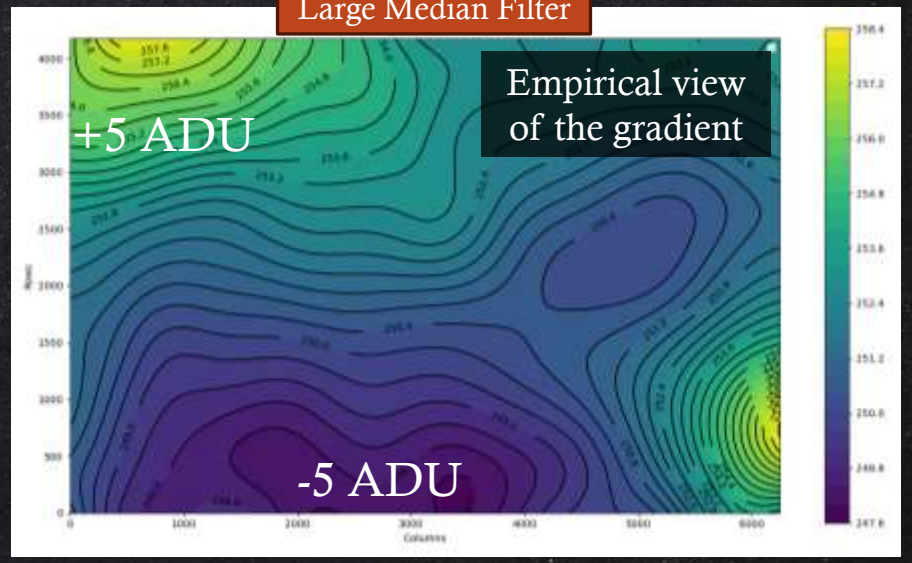
- RMSS 2023, Bortle 2-3?

Deep dive: Gradient and Background Estimation/Removal

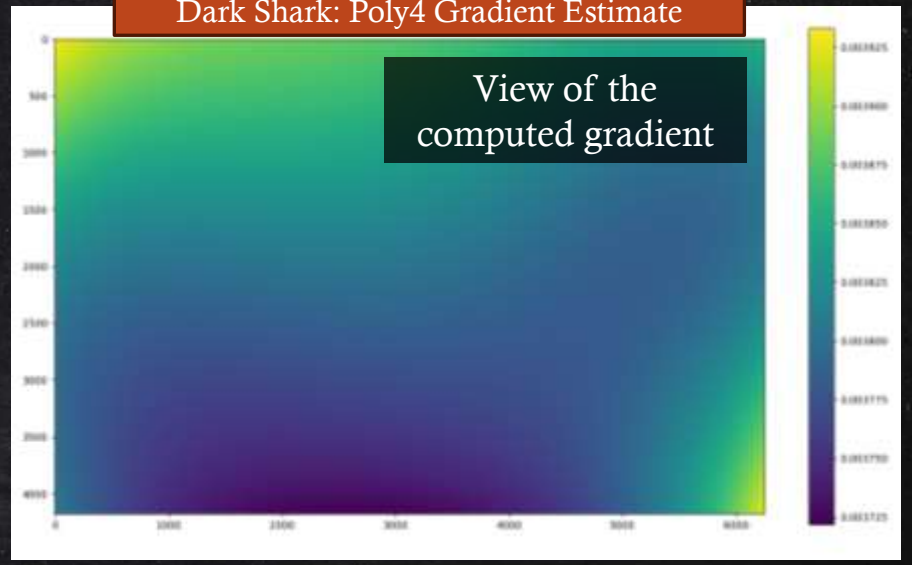
Deep dive: Gradient and Background Estimation/Removal

- Gradients often refer to spatially varying nuisance signals and trends in our images.
  - Causes: Light pollution, sky glow, moon, light leakage, uneven sensor conditions…
- Background often refers to the "zero point" in the image.
- In a computational workflow, the Gradient+Background can be determined at the same time.
- **Math problem = "solve for $Signal(x, y)$":**
  - $FFDF_Correct_Image = Signal(x, y) + Noise(ADU) + Gradient(x, y) + Background(\cdot)$
- Background estimation is one of the most challenging step (to do well) in an autonomous processing workflow.
  - **Gold standard/Status Quo:** User manually selects or guides the selection of sampling points for use in gradient estimation (and possibly background as well).
- Figures at right show the complexity of the gradients in a +55° Alt FOV at a Bortle 4 site. The magnitudes of the gradient are LARGER than the signal magnitude scale of dark nebula features and light nebulosity.

Dark Shark: Very Large Median Filter



Dark Shark: Poly4 Gradient Estimate



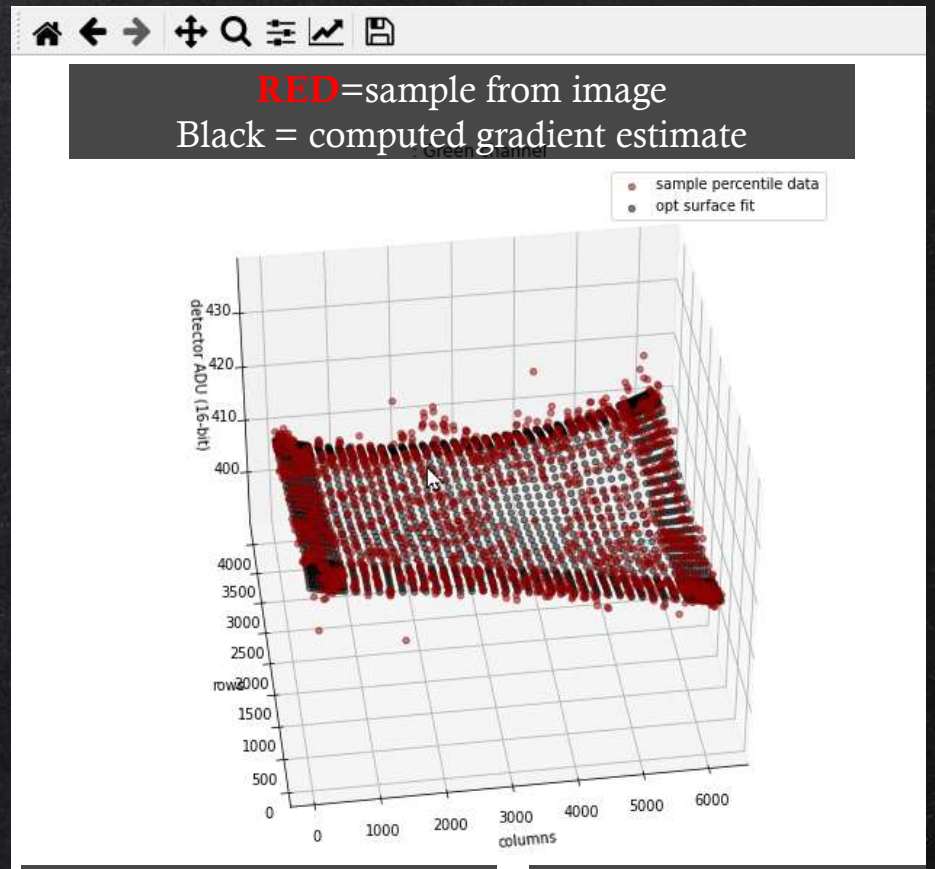
Gradient and Background Estimation/Removal: how to do it well and FAST

- **Math problem = "solve for $Signal(x, y)$":**
  - $FFDF_Correct_Image = Signal(x, y) + Noise(ADU) + Gradient(x, y) + Background(\cdot)$

Gradient Estimation Hazards:

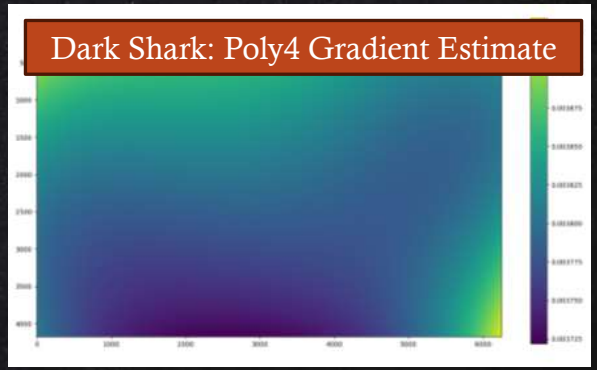
- Stars, especially bright stars
- Deep Sky Objects (extended nebulosity, real signal)
- High star density / star clouds (Milky Way)
- Shot Noise
- Computational complexity?

- **Fitting a surface to data, (even large amounts) is not necessarily a time consuming process**
  - **Good and Fast:** Least Squares polynomial regression (unique solution, robust)
  - **Great and O.K. Fast**: LOWESS local regression, Smooth splines (less robust)
  - **Great and Slow:** optimization based regression and solvers (non-unique, least robust)
- **My experience is that informed sampling (of the gradients) and robust statistical sampling methods work well. Takeaways:**
  - **4th order polynomials are general enough to cover most gradients with good sampling.**
  - **Extra edge sampling is effective at reducing errors where they are most common/visible.**
  - **No unique solution for all astrophotography objectives. An overfit or aggressive background can be used to emphasize a specific DSO by diminishing surrounding phenomena**
- **To do autonomous gradient estimation (for reproducible or near-real-time applications) it would be *just great* if we knew where not to sample!**



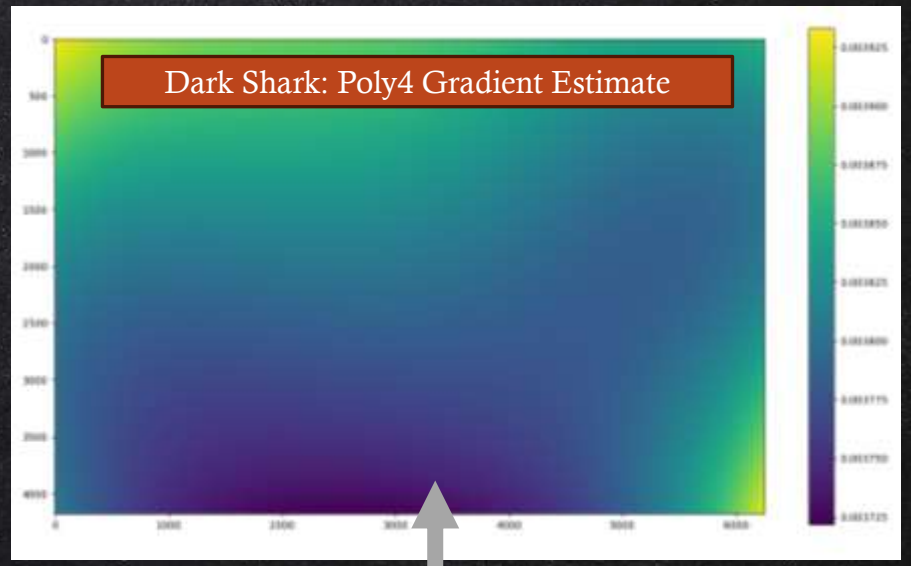
Z-axis (height) is the intensity level (ADU) of the black background

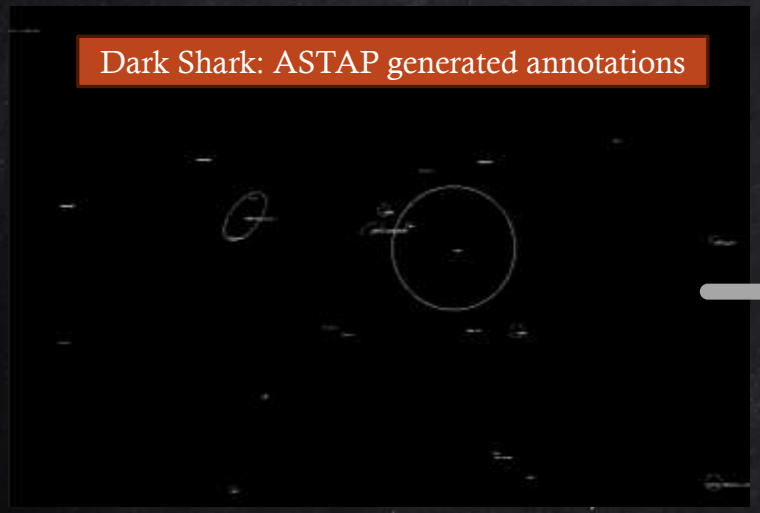
X and Y axes are the image columns/rows

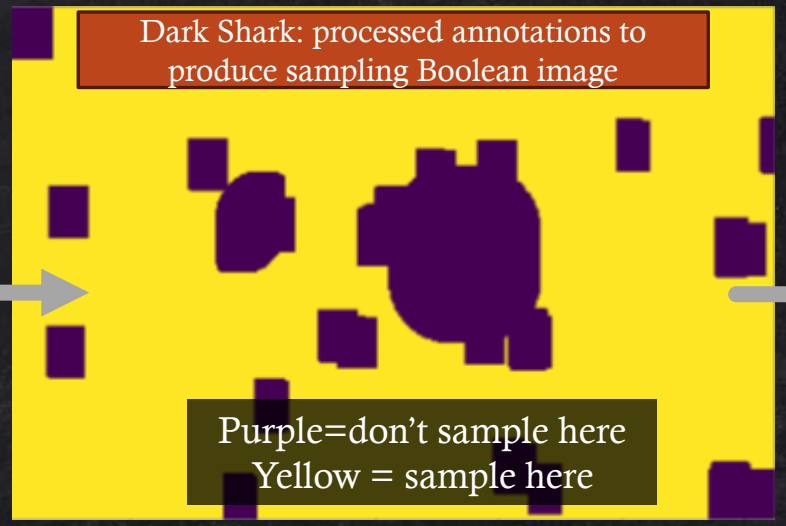


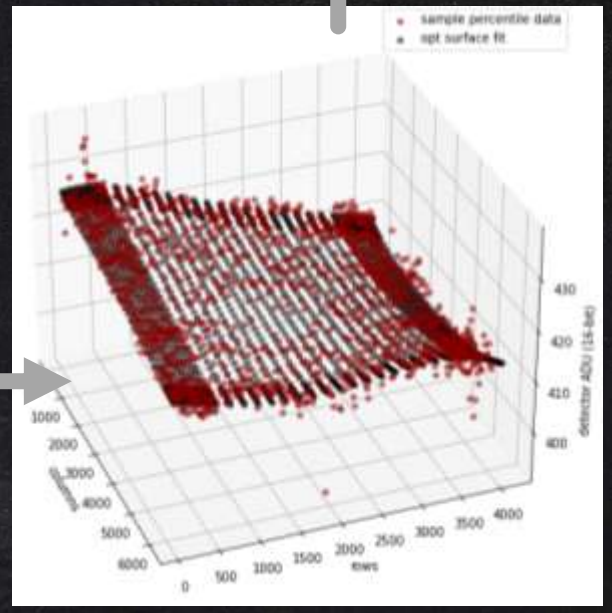
Gradient and Background Estimation/Removal

- Plate solving tells us where the camera is, so lets just ask a database if there is anything in our FOV to avoid?
- ASTAP can produce annotated images from the command line interface. AstroComPYute has a python wrapper so that it can be called from a processing workflow and for extracting the positions of the annotations.
- Note the holes in the sampling points of the 3D plot.
- Neat… hmmm… but we need to avoid DSOs, extended nebulosity, star clouds, …
- **I like this usage of external/extra info, but we are only scratching the surface here. LETS USE A SLEDGE HAMMER!**







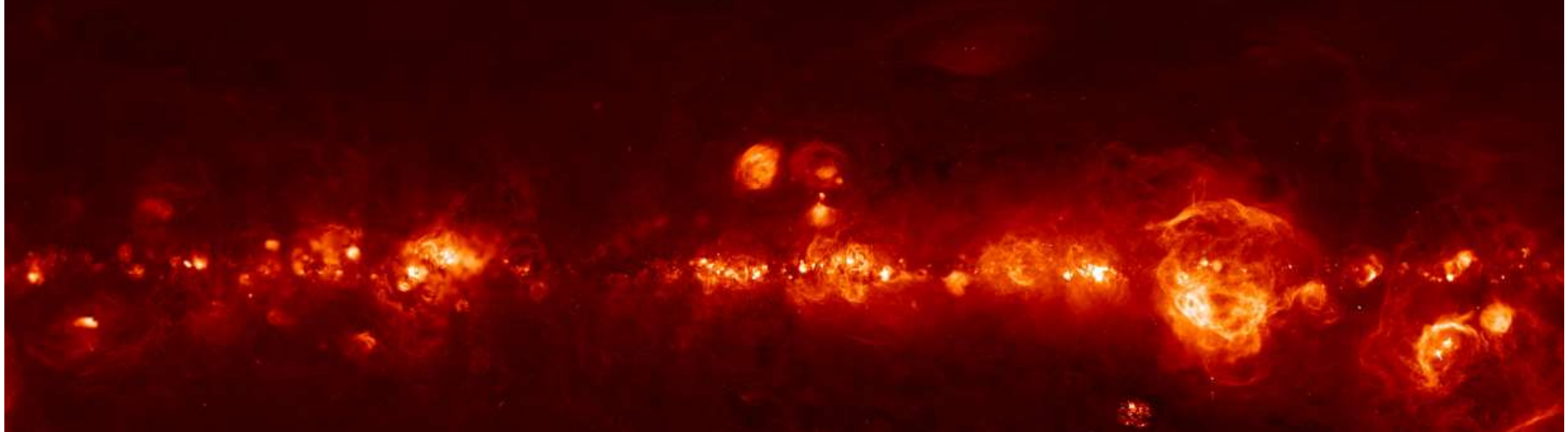


NASA Deep Star Maps

- All-sky star maps constructed from a sample of 1.7 billion stars from the Hipparcos-2, Tycho-2, and Gaia Data Release 2 star catalogs.
- Available in high resolutions (still lower res than our cameras take):
  - 16384 x 8192
  - 32768 x 16384
  - 65536 x 32768
- Celestial J2000 coordinates
- AstroComPYute pre-computes an interpolator that can be rapidly evaluated at plate-solve solutions.

NASA/Goddard Space Flight Center Scientific Visualization Studio. Gaia DR2: ESA/Gaia/DPAC. Constellation figures based on those developed for the IAU by Alan MacRobert of Sky and Telescope magazine (Roger Sinnott and Rick Fienberg).

H-alpha Full Sky Map: WHAM+VTSS+SHASSA



- All-sky survey from (predominantly) WHAM (northern hemisphere) and SHASSA (southern hemisphere) using the same apparatus (they moved it to both locations!)
- Resolution: 8640 x 4320 (2.5 arc-min pixels)
- Only in Galactic Lat/Long Coordinates
- This map/survey is processed with a scipy/astropy process to convert to celestial coordinates
- AstroComPYute pre-computes an interpolator that can be rapidly evaluated at plate-solve solutions.

H-alpha Full Sky Map: WHAM+VTSS+SHASSA

Here in celestial coordinates …that was driving me crazy!

- All-sky survey from (predominantly) WHAM (northern hemisphere) and SHASSA (southern hemisphere) using the same apparatus (they moved it to both locations!)
- Resolution: 8640 x 4320 (2.5 arc-min pixels)
- Only in Galactic Lat/Long Coordinates
- This map/survey is processed with a scipy/astropy process to convert to celestial coordinates
- AstroComPYute pre-computes an interpolator that can be rapidly evaluated at plate-solve solutions.

AstroComPYute in action: Informed Processing Example

AstroComPYute in action: Informed Processing Example of Sagittarius Nebulas

Importing and loading AstroComPYute

Light frame, master dark/flat

Read files

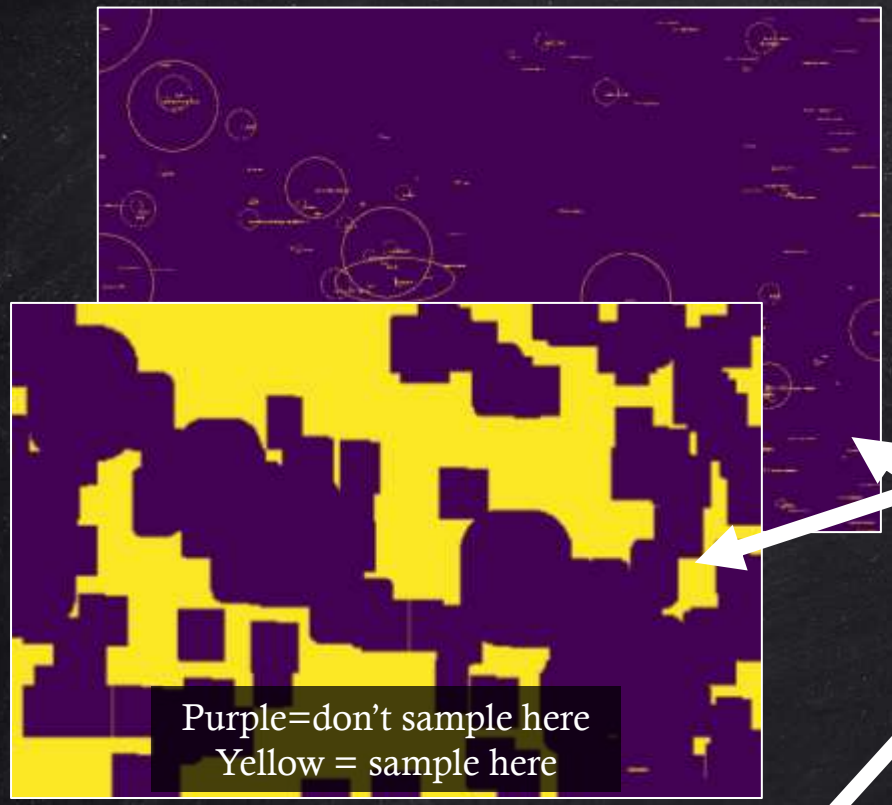
FF & DF correction

Visualization

```
#%%# Import AstroComPYute modules
import astrocompyute.apriori as apriori
import astrocompyute.fileio as fileio
import astrocompyute.imagemath as imagemath
import astrocompyute.registration as registration
import astrocompyute.correction as correction
import astrocompyute.visualization as astroviz
import astrocompyute.filters as myfilters
import astrocompyute.enhance as enhance
import astrocompyute.color as mycolor

from astrocompyute.visualization import ShowImage, ShowImageRGB,
QuickInspectRGB, PlotHistRGB, ContourMultiPlot

#%%##########################################################################
#### Single file processing and inspection example
fp = '2023-06-16_22-45-39__-
9.80_5.00s_0012_2.61_$$ECCENTRICITY$$_lagoon_swan_eagle_trifid.tif'


MasterFlatFP = 'FF_Sigma105_fullSpec.npy'
MasterDarkFP = 'DF_5s_n10c_g200.npy'

 # Load Dark Frame and Flat Field
DF_rgb = np.load(MasterDarkFP)
FF_rgb = np.load(MasterFlatFP)

#%%##########################################################################
# Read and debayer the image
FrameImage, frame_imBayer0 = fileio.ReadASI_TIF_FITS(fp, salt = False)
FrameImageInt = FrameImage.mean(axis = 2)

# FF and DF correction
FrameImage = correction.FFDF_correct(FrameImage,
                                     FF_rgb,
                                     DF_rgb,
                                     FFnorm2unity = True,
                                     norm2unity = False)

QuickInspectRGB(FrameImage, perc_black=5, perc_white = 99.)
```

User provides/changes for their workflow

AstroComPYute in action: Informed Processing Example



Generate annotations and DSO sampling Boolean

Plot Images

Plate Solve (ASTAP CLI)

Calculate RA/DEC coordinates of each pixel

```
ASTAP Plate Solve in: 2.1002936363220215 seconds
Using star database H18
ASTAP solver version CLI-2022.11.11
Solution["] x:=5.273216*x+ 5.515041*y+ -27954.499078,  y:=-
5.514563*x+ 5.279941*y+ 6142.376088
Solution found: 18: 11  44.6 -19d 41  19
Solved in 1.1 sec offset was 87.2d.  Mount offsetRA=-91.6d,
offsetDEC=19.7d.  Used stars down to magnitude: 9.8
Using star database H18
```

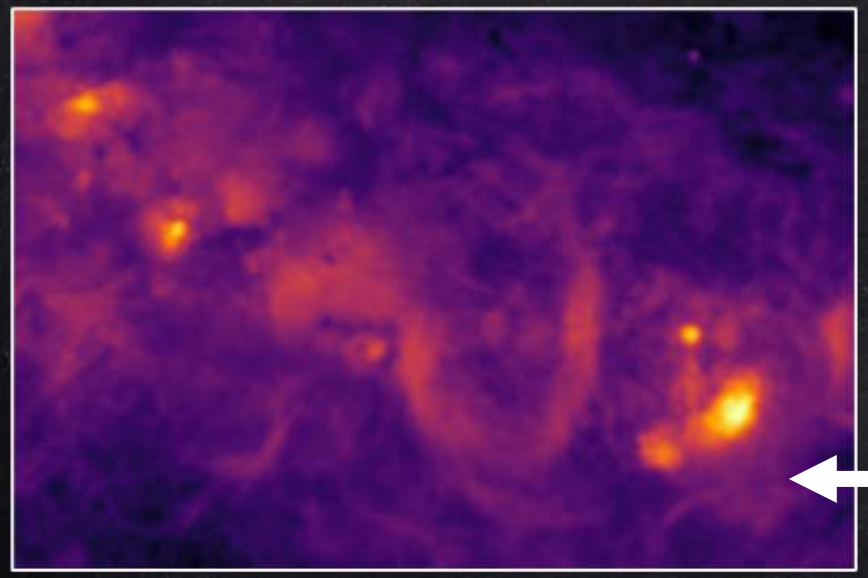
```python
#%%# Import AstroComPYute modules
#%%##########################################################################
# Use python wrapper for ASTAP to compute a DSO sampling boolean and
annotations
not_DSO_mask, annotation = apriori.PlateSolveMask(fp,
                    astap_executable = '"astap.exe"' ,
                    dilate_n = 200,
                    return_annotation_bool = True,
                    kill_the_file = True,
                    )

ShowImage(not_DSO_mask)
ShowImage(annotation)

#%%##########################################################################
# Python wrapper for ASTAP CLI to perform fast plate-solving (~1-2 seconds)
platesolve_solution = apriori.PlateSolve(fp,
                                         ra_guess = None, dec_guess = None,
                                         fov = 8.85 ,
                                         search = 170,
                                         astap_executable ='"Castap_cli.exe"')

# row/column coordinates of each pixel
ImageRows, ImageCols = imagemath.GetImageCoords(AnalysisImage=FrameImage)

# compute the RA and DEC coords of each pixel
deg_RA, deg_DEC = apriori.Pixels2RADEC_Transform3D_cuda(ImageRows.ravel(),
                                                        ImageCols.ravel(),
                                                        platesolve_solution)
```

AstroComPYute in action: Informed Processing Example

Prepare Deep Star Map Interp.

Evaluate Star Map at Image Coordinates

Plot result



Prepare Hα Interpolator

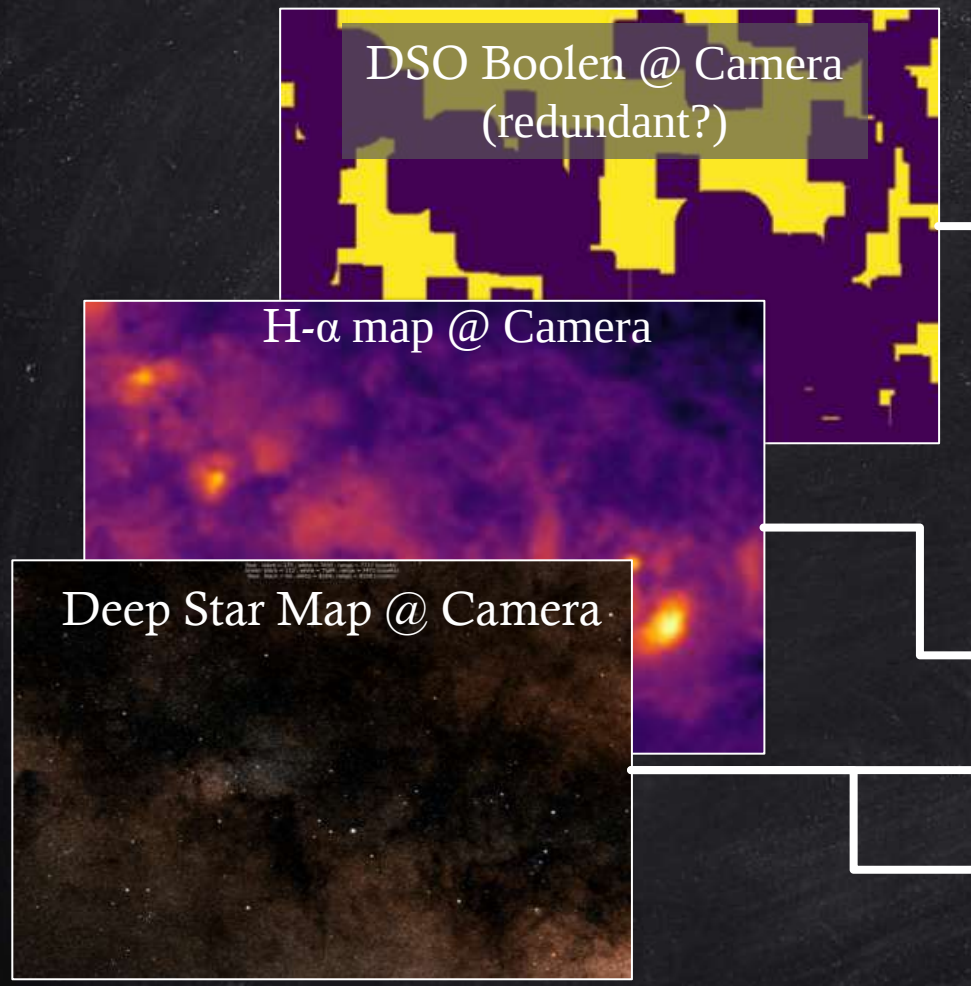
Evaluate Hα at Image Coordinates

Plot result

```
#%%##########################################################################
# retrieve the Gaia starmap interpolator (obtain from
https://svs.gsfc.nasa.gov/4851)
GaiaDR2_all_sky = 'starmap_2020_16k.exr'
EvalStarmap = apriori.GenStarmapFast(mode = 'rgb',
path_to_EXR_GaiaDR2_NASA_starmap=GaiaDR2_all_sky)

# evaluate the starmap interp at the image coordinates
starmap_at_image = EvalStarmap(deg_RA, deg_DEC) #.reshape(StackedImageRGB.shape)
starmap_grey = starmap_at_image.mean(axis = 2)

# in case the map is flipped, repair it here with a secondary test
starmap_grey, secondary_test = apriori.SecondaryTestStarmap(starmap_grey,
                                                            FrameImageInt,
                                                            threshold_perc = 95)


# visualize a linear stretch of the image
QuickInspectRGB(starmap_at_image, perc_black=5, perc_white = 99.95)

#%%##########################################################################
# H-alpha map file, obtain from:
#    https://faun.rc.fas.harvard.edu/dfink/skymaps/halpha/data/v1_1/index.html
Halpha_filepath =  'Halpha_map.fits'

# retrieve the H-alpha map interpolator
EvalHAmap = apriori.GenHAmapFast(path_to_Finkbeiner_allSky_Halpha = Halpha_filepath,
                                 dec_res=8192,
                                 ra_res=16384)

# evaluate the h-alpha map at the image coordinates
HAmap_at_image = EvalHAmap(deg_RA, deg_DEC).reshape(FrameImageInt.shape)

# repair the H-alpha map if it is flipped as well
HAmap_at_image = HAmap_at_image[::secondary_test[0],::secondary_test[1]]

Ha_p1, Ha_p995 = np.percentile(HAmap_at_image, [1,99.5])
ShowImage(HAmap_at_image, min_color_val=Ha_p1, max_color_val=Ha_p995)
```

Gradient and Background Estimation/Removal

- **The retrieved Deep Star Map and H-α map at the camera FOV can then combine to produce a sampling strategy for gradient estimation.**

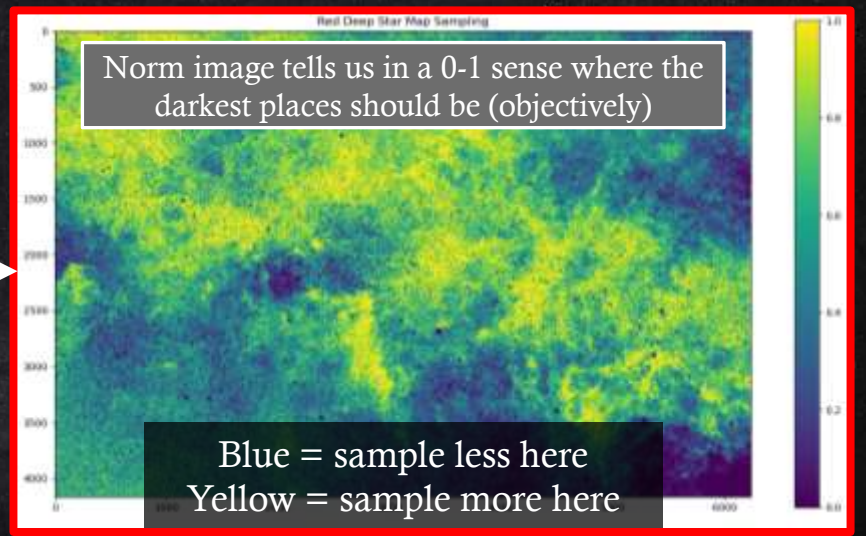


Star Identification & masking

Combine: AND

Threshold

Normalize



Gradient and Background Estimation/Removal

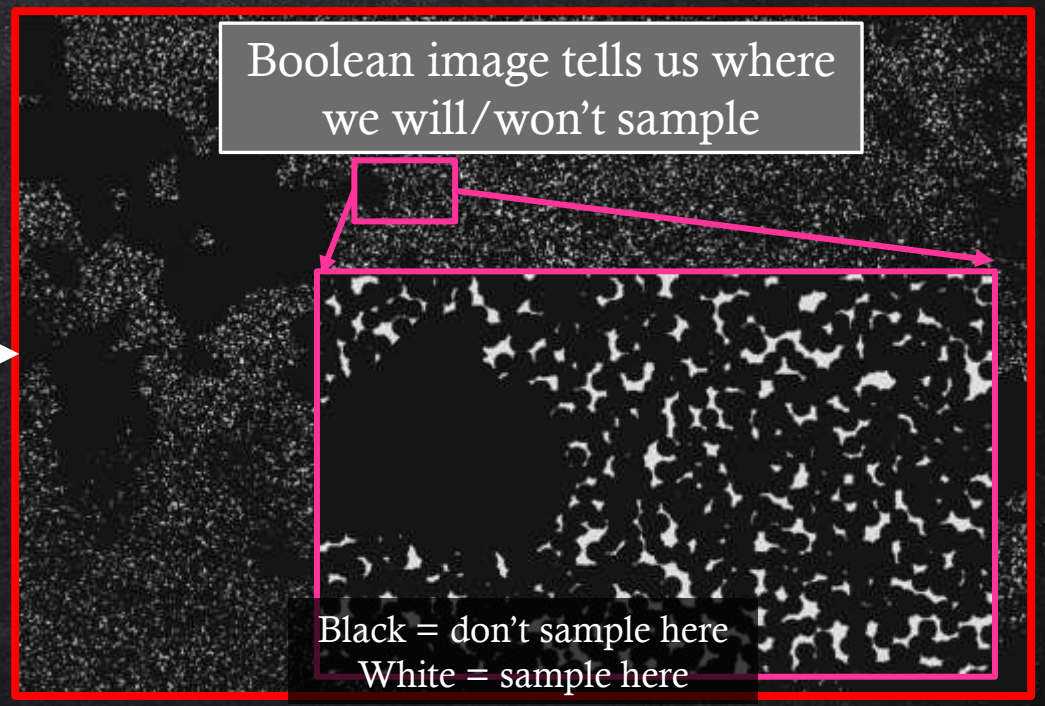
- **The retrieved Deep Star Map and H-α map at the camera FOV can then combine to produce a sampling strategy for gradient estimation.**

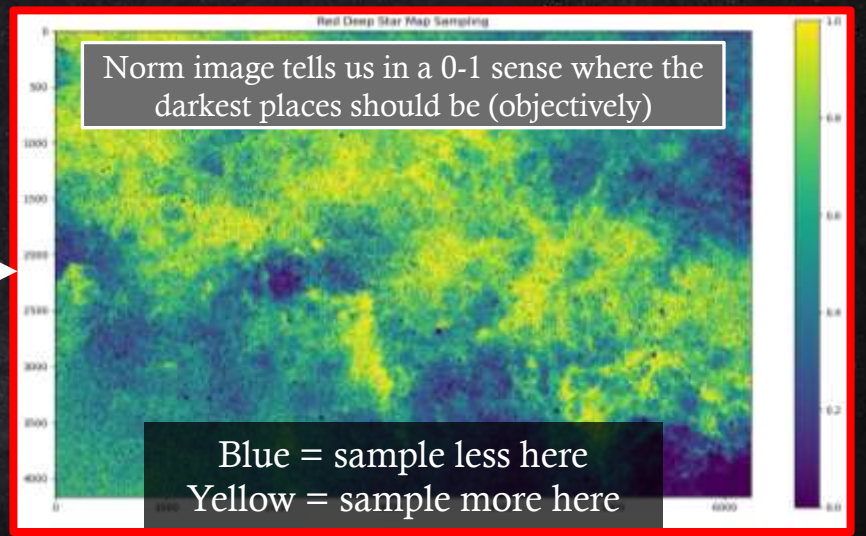
Star Identification & masking

Combine: AND

Threshold

Normalize





AstroComPYute in action: Full Processing Example

(demo)

Conclusions, Looking Ahead, References, Further Reading

Final Thoughts, Looking Ahead

- **Where are we now:**
  - Base end-to-end processing: <15 seconds @ 26MP resolution
  - Add star separation+reduction+starless processing ~45 seconds
- **What would it take to get to a 5-second latency astrophotography image**
  - Open the door to: <5s subs, shoot untracked (<100mm FL), unguided,
  - By the end of 2024 GPU's and CPU's have 3x processing speed →
  - Less than 5 seconds for base processing, 15 seconds for full star reduction processing?
  - …that's pretty close!
  - …how do we reduce the noise?

- **Compute hardware aside what else can be done:**
  - Parallel imaging trains can increase sample rates and accumulation (dragonfly)
  - Imagine teaming up with your friends across the country, sending data to AWS and collab processing in true massive parallel form.
  - Imagine a future gen of smart telescope with multiple objectives and cloud processing…
- **Other computational astrophotography possibilities:**
  - Basic and spatially-varying deconvolution/deblurring? (scikit-image restoration, AstroSharp)
  - Multi-parameter optimization for automatic collimation + focus (Data Acq direction)
  - All-sky dust survey, other emission wavelengths???
  - Automatic sweep/auto-stich mosaics!? (might already work)
  - Dynamic stretching + binning + noise-reduction aggressiveness so as to not exceed noise targets.

Further reading and resources

- ASTAP:
  - **https://www.hnsky.org/astap**
- StarNet++:
  - **https://www.starnetastro.com/**
- MARS-pi Survey project:
  - https://pixinsight.com/doc/docs/MARS/MARS.html
- All sky mosaic RGB survey (2009)
  - https://www.milkywaysky.com/
- NASA Deep Sky Maps
  - https://svs.gsfc.nasa.gov/4851
  - NASA/Goddard Space Flight Center Scientific Visualization Studio. Gaia DR2: ESA/Gaia/DPAC. Constellation figures based on those developed for the IAU by Alan MacRobert of Sky and Telescope magazine (Roger Sinnott and Rick Fienberg).
- All-sky H-alpha survey (in progress, DR0 released)
  - https://mdw.astro.columbia.edu/
  - Finkbeiner, Douglas P. "A full-sky Hα template for microwave foreground prediction." The Astrophysical Journal Supplement Series 146.2 (2003): 407.
- All-Sky Dust Mapping:
  - http://argonaut.skymaps.info/
- Richard Wright's: "Why Guiding Needs to Die (And How It's Going to Happen)":
  - https://www.youtube.com/live/p4bpFJSdHDY
- Astro processing in Python:
  - https://yager.io/Astro.html
  - https://github.com/DaveStrickland/AstroPhotography

THANK YOU!

Download and try AstroComPYute…

GitHub

github.com/Daniel-Champion/AstroComPYute

…anyone interested in contributing? Crazy ideas?

Compact/portable desktop computer

Camera and fast lens (F1.2) on a Vixen rail

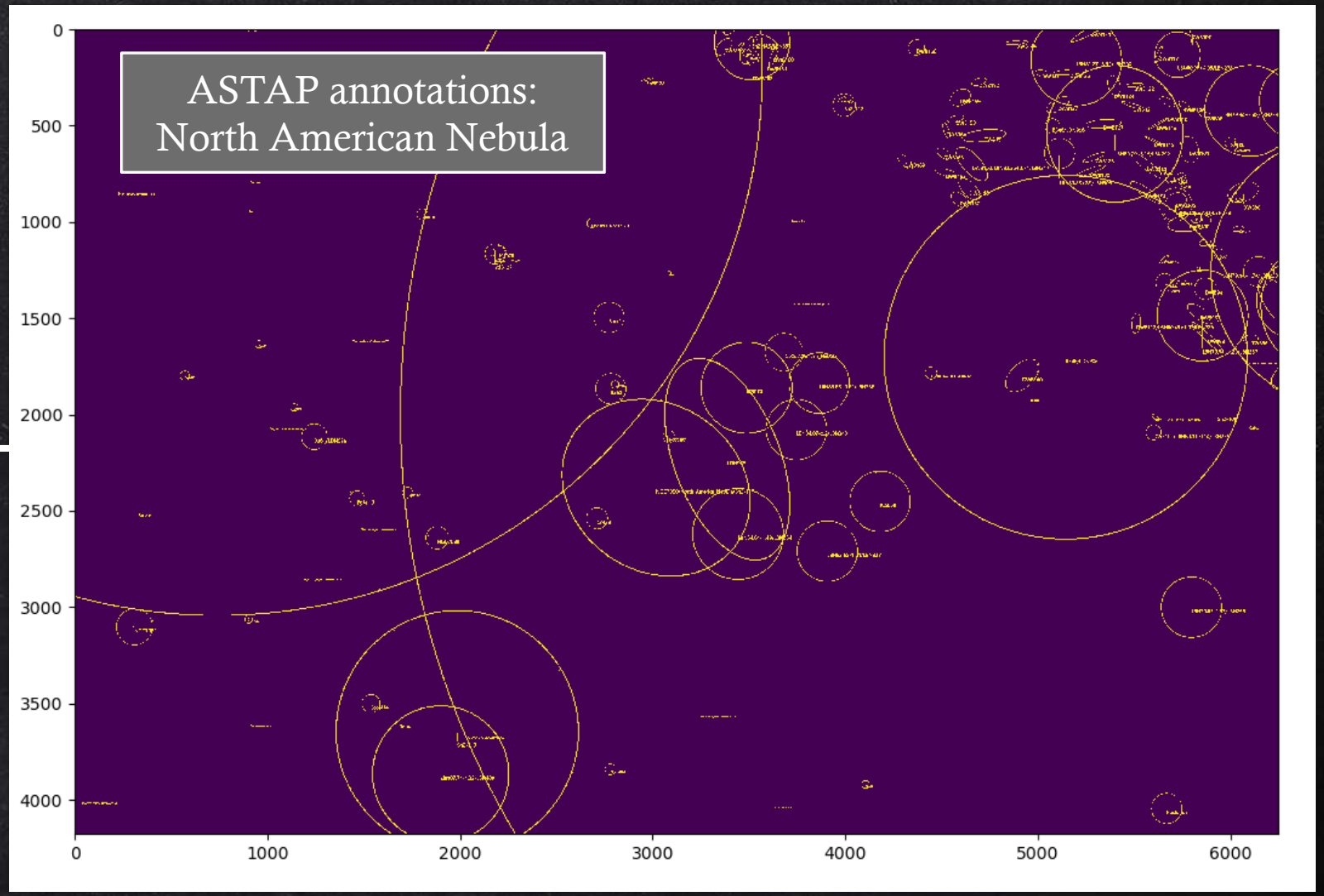
Monitor

Annotations can work great, but not always

- There are some regions in the sky that are absolutely filled with objects and annotations.
- These can swamp the annotation-based Boolean image and render it unusable.

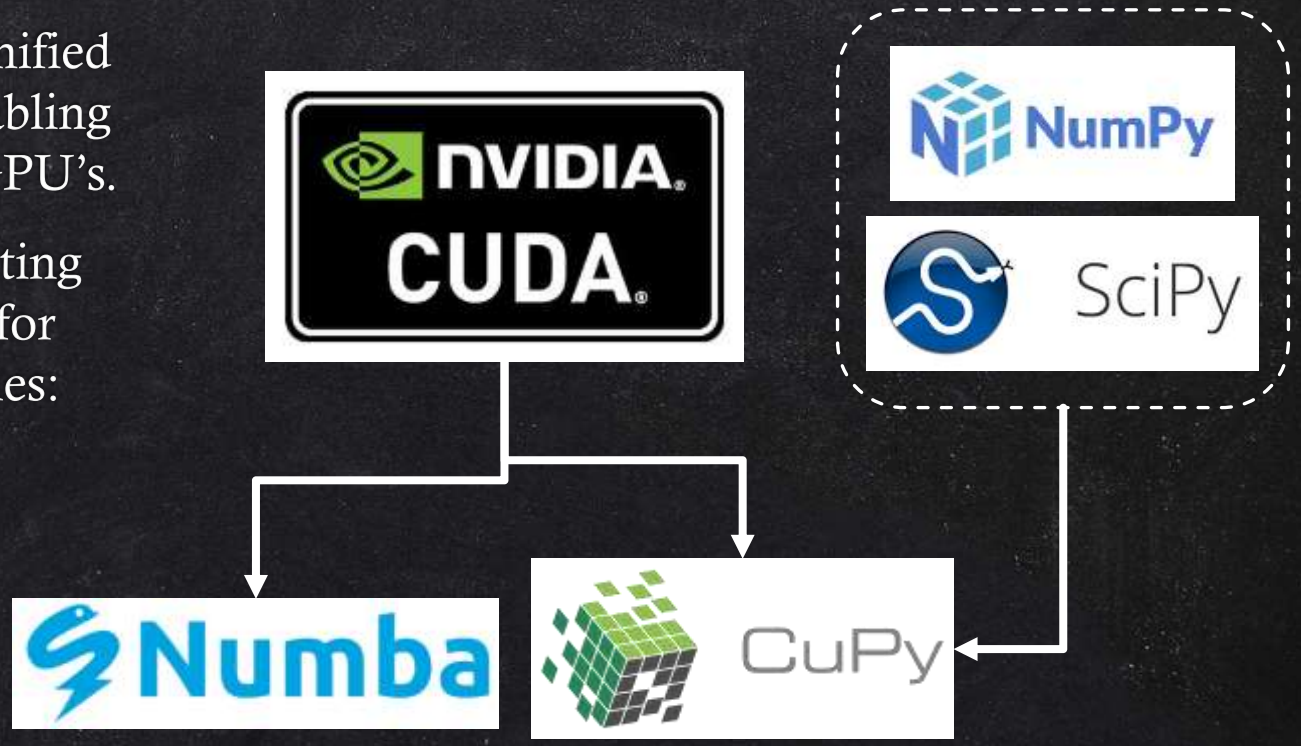
Resulting sampling Boolean image!!

Purple=don't sample here
Yellow = sample here



GPU General Purpose Compute

- In 2007 Nvidia released the Compute Unified Device Architecture (CUDA) toolkit enabling general-purpose computing on Nvidia GPU's.
- Enables the massive parallelism and floating point arithmetic on the GPU to be used for more than just graphics, some key libraries:
  - cuBLAS – basic linear algebra
  - cuFFT – FFT library
  - cuSOLVER – dense and sparse solvers
  - cuSPARSE – sparse linear algebra
  - …



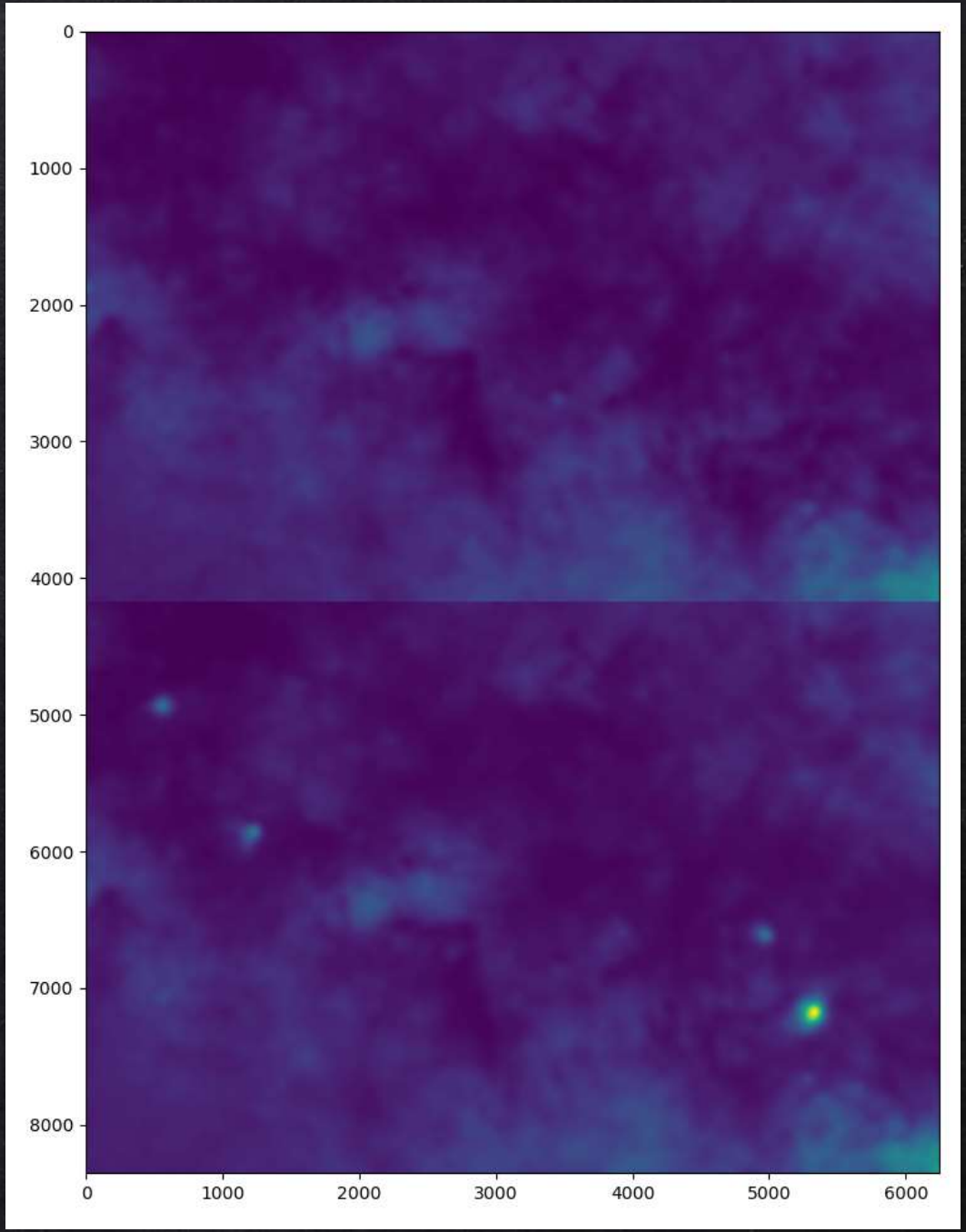
Starmap 16k example evaluated and zoomed @ 7”/pixel

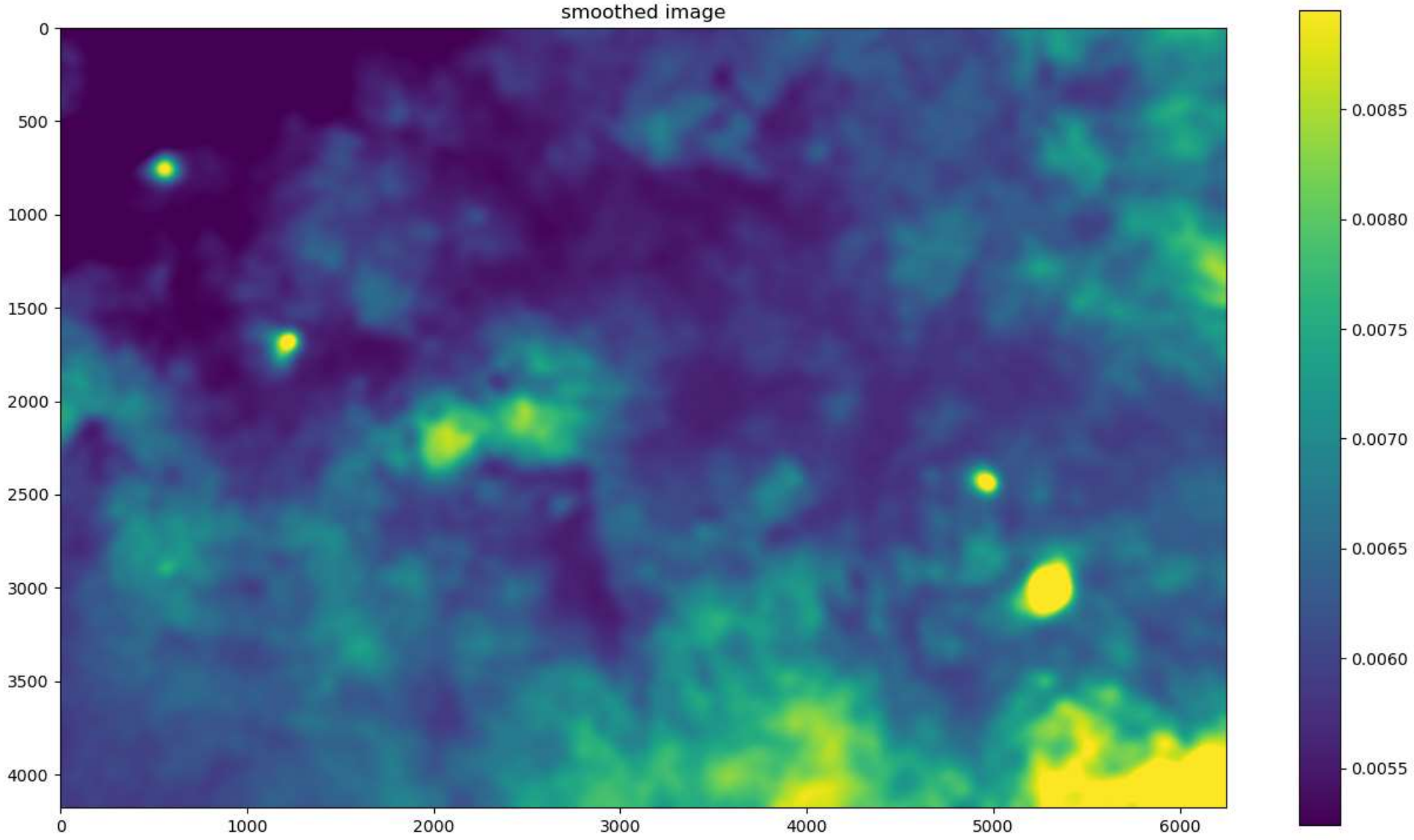
Alternative ways of using the Deep Star Map

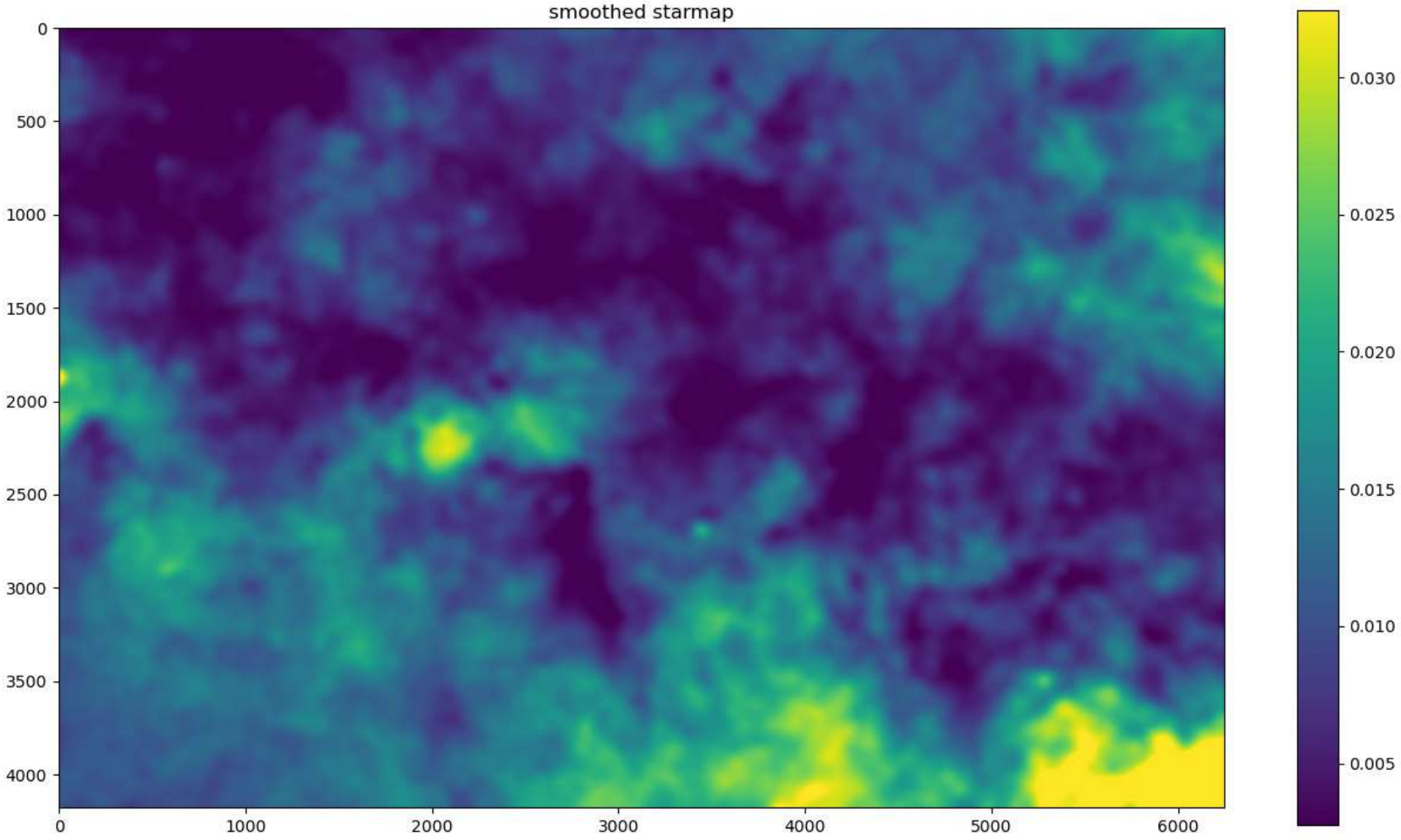
Figure Top: starmap (smoothed)

Figure Bottom: smoothed camera image, where a transformation function was determined that maps camera to starmap in an error minimizing way.

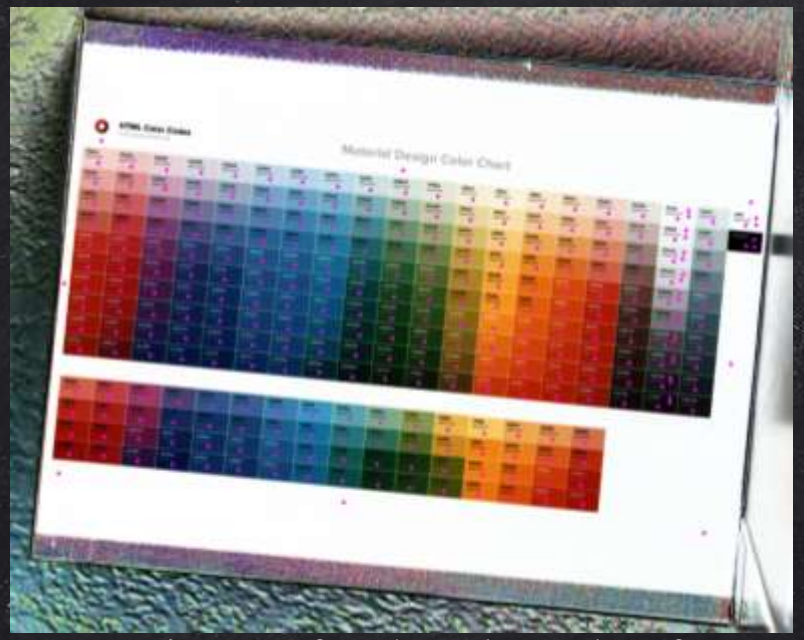
This solves for the latent gradient/image transform that matches astrometry measurments.




smoothed image
0
500
1000
1500
2000
2500
3000
3500
4000
0
1000
2000
3000
4000
5000
6000
0.0085
0.0080
0.0075
0.0070
0.0065
0.0060
0.0055


smoothed starmap
0
500
1000
1500
2000
2500
3000
3500
4000
0
1000
2000
3000
4000
5000
6000
0.030
0.025
0.020
0.015
0.010
0.005

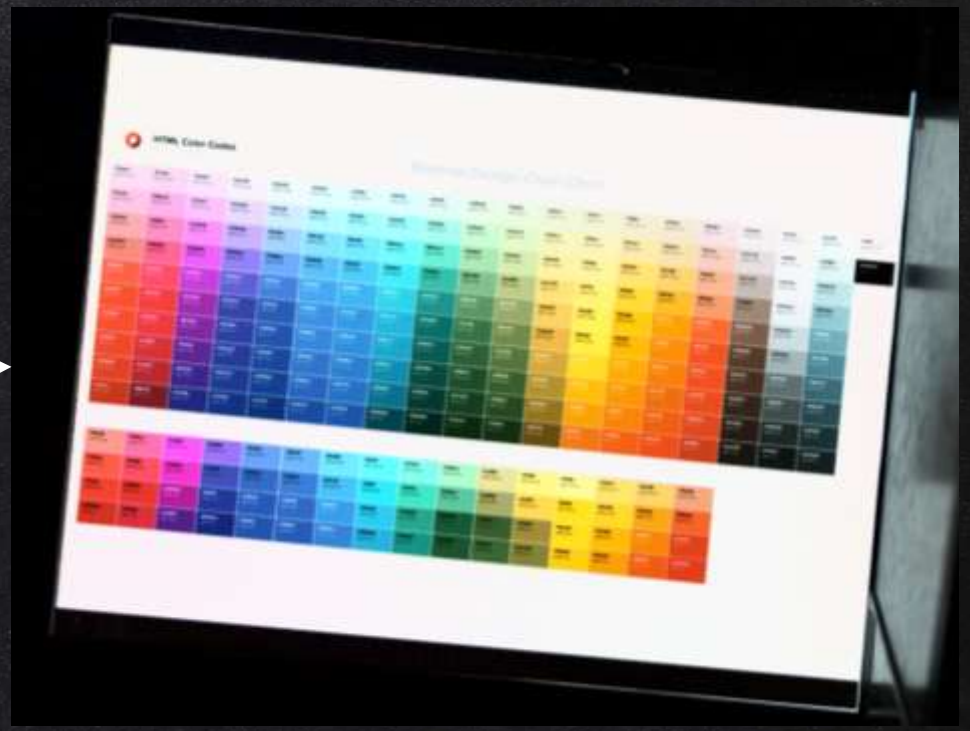
Color correction



Camera image of Color Chart, the “features”

Color Chart, “known answer”

Training data for ML Regression



Low complexity regressor for mapping observed color data to corrected color data. Produce this for each camera and each filter one uses.