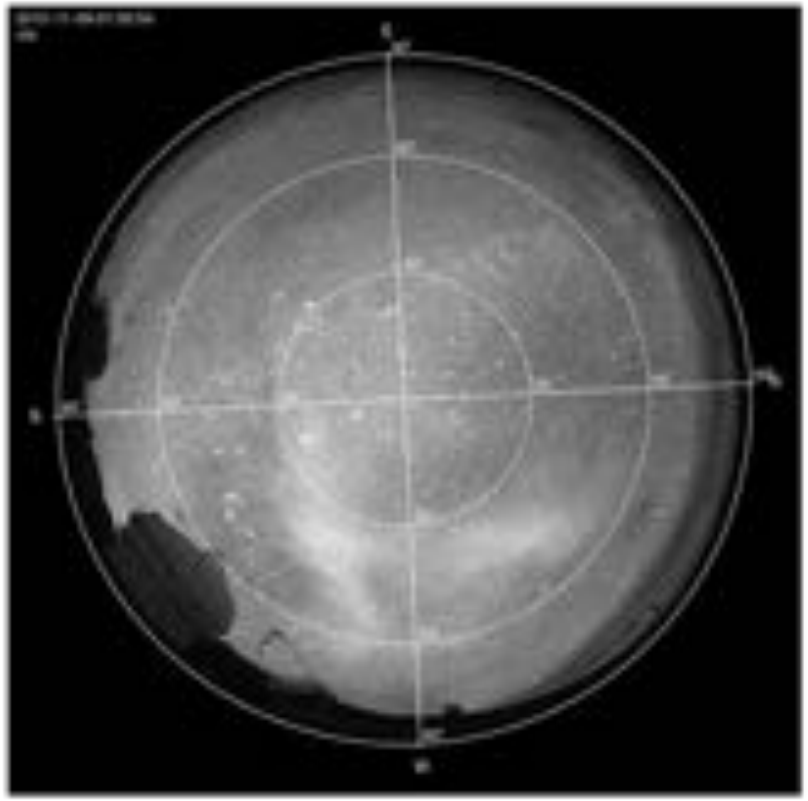
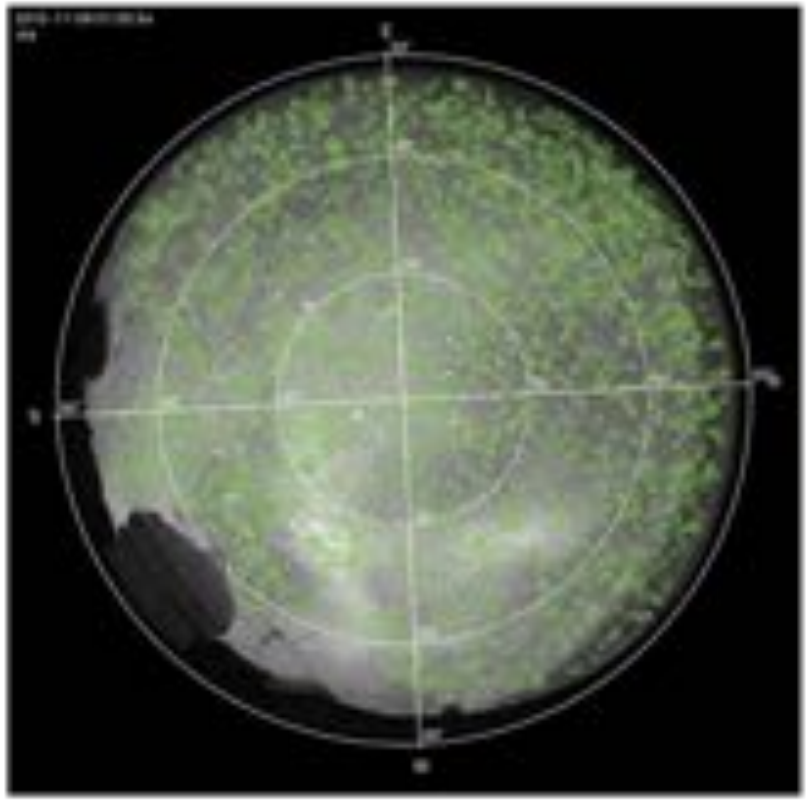
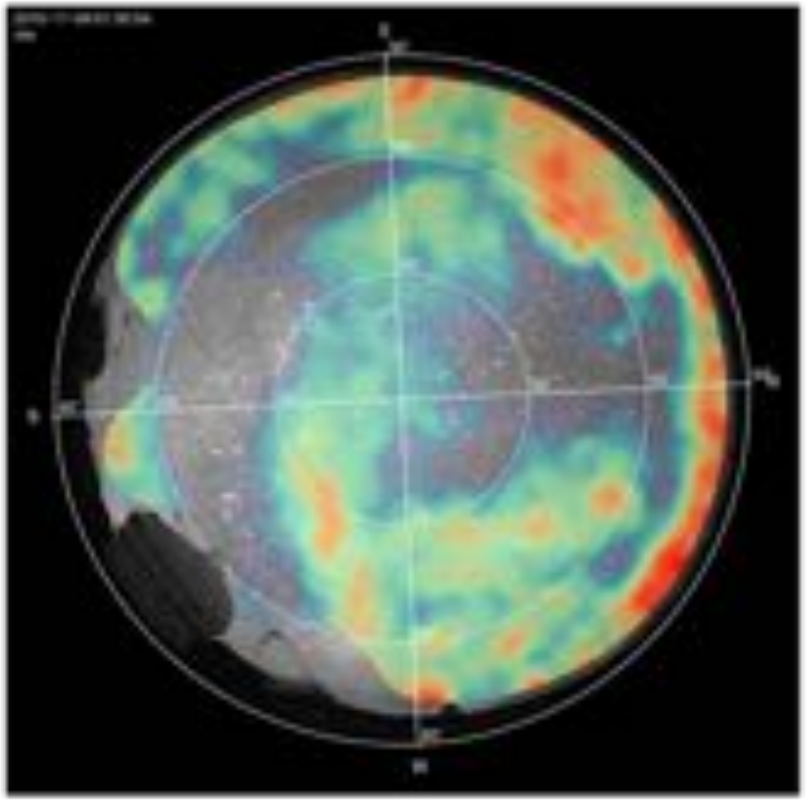




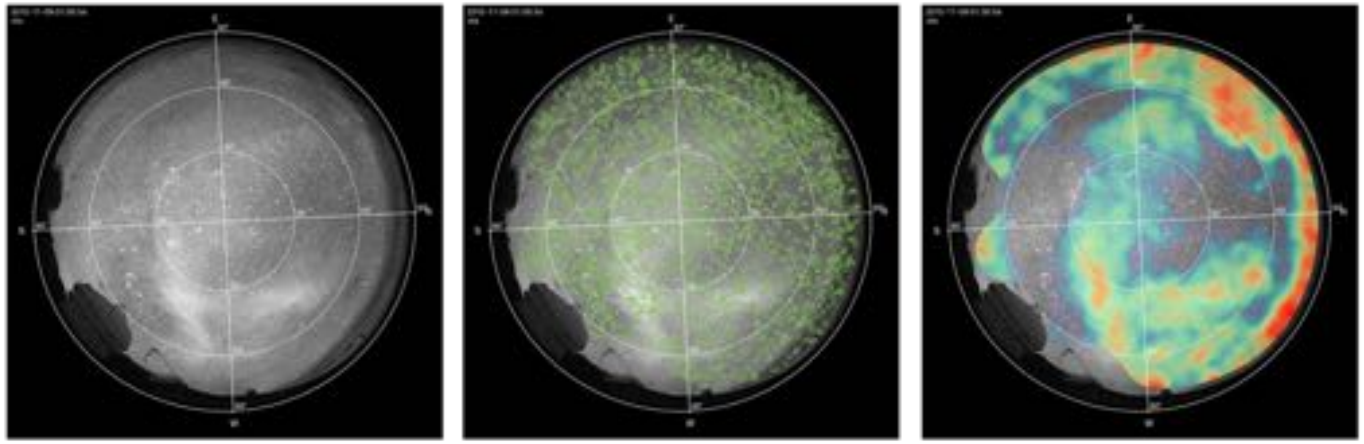
A New Tool to Analyse All Sky Images

Sabrina Einecke
Tobias Hoinka
Helena Nawrath

THE TOOL

quocca: quick **o**bservation of **c**loud **c**overage using **a**ll sky images

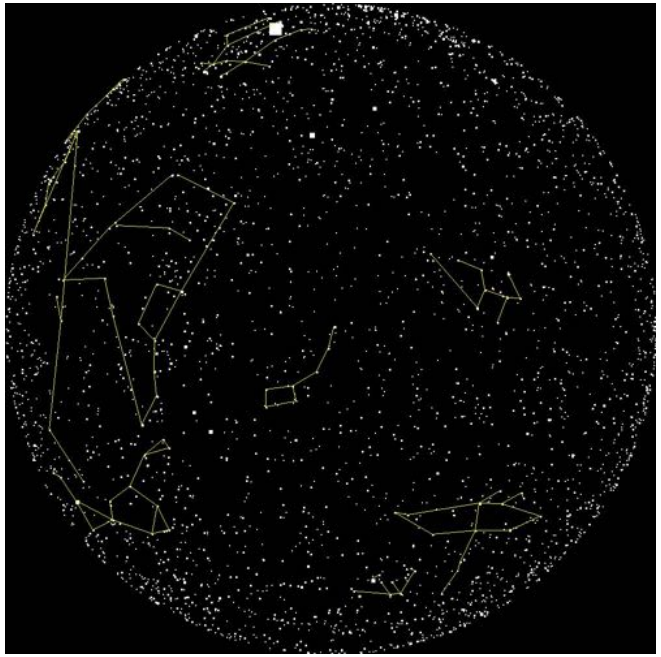
- Python module available via GitHub
- Suitable for multiple different All Sky Cameras (e.g. CTA, IceAct)
- Detection of stars
- Generation of cloud maps
- Extendable to cloud prediction
- Extendable to cloud height determination



THE INGREDIENTS

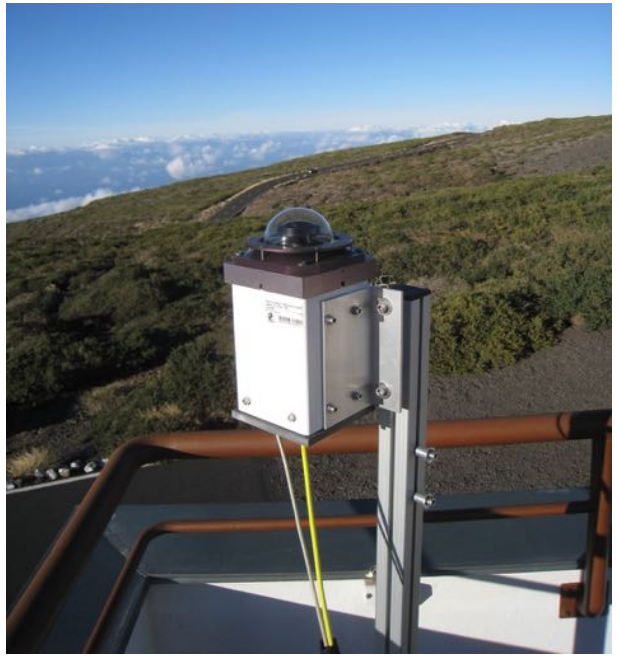
Catalog of Stars

- RA/Dec
- Magnitude
- Variability



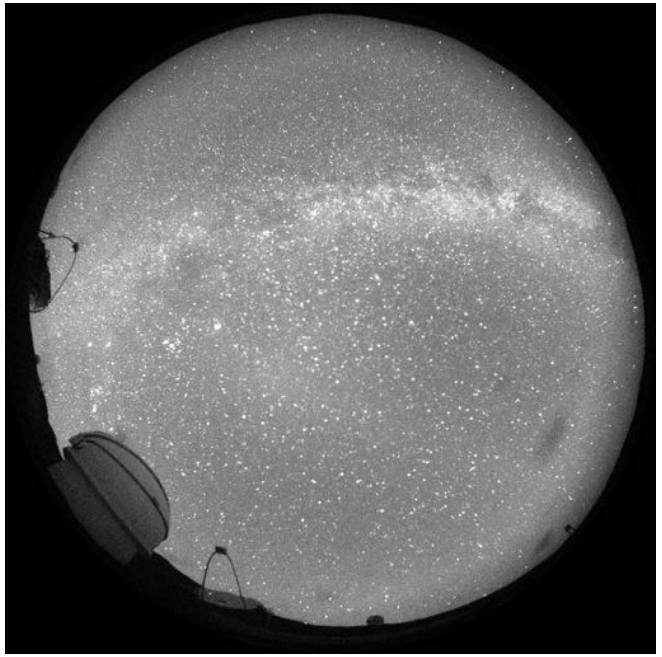
All Sky Camera

- Geographic location
- CCD chip information
- Lens information
- Orientation



All Sky Image

- Image
- Time
- Camera



THE BASIC APPROACH

1. Definition of camera
2. Import of image to be analysed
3. Add catalog stars to be considered for this image
4. Estimation of magnitude and visibility of chosen stars

```
from quocca.camera import Camera

cam = Camera('cta')
img = cam.read('2015_11_04-00_01_31.mat')
img.add_catalog()
res = img.detect(method='llh')
```

CAMERA DEFINITION

Definition File (cameras.yaml)

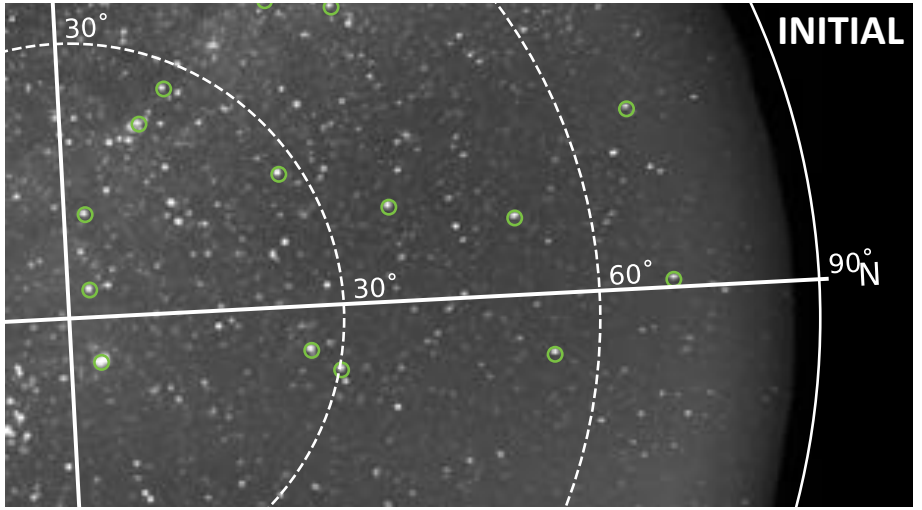
- Name of the camera
- Azimuth offset
- Location: Height, latitude, longitude
- Lens specifying the mapping
- Mask for excluding the environment
- Format of the timestamp
- Size of the CCD chip
- Resolution of the CCD chip
- Calibration values for the detection methods

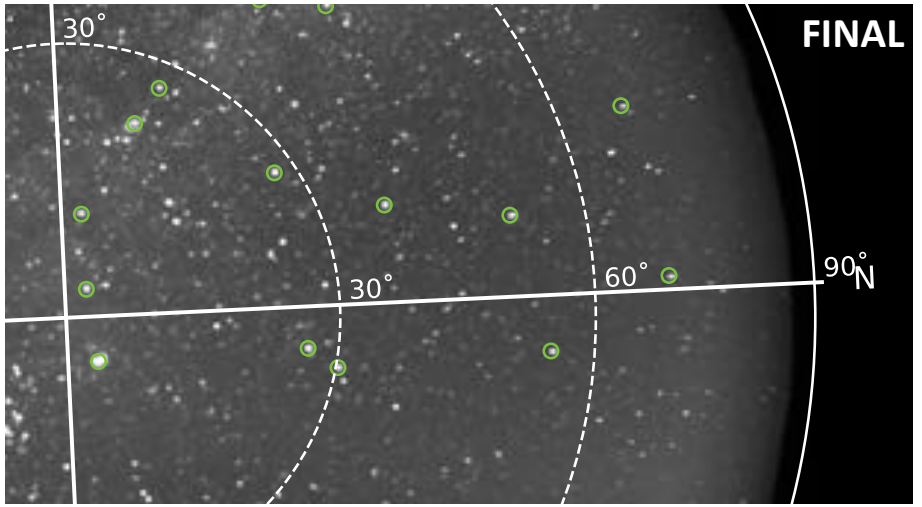
```
cta:
  az_offset: 87.29914181597786
  filter_star_detection:
    '1970-01-01T00:00:01.000': 0.2617892814129304
    '2015-12-19T00:00:01.000': 0.8903290237727741
  llh_star_detection:
    '1970-01-01T00:00:01.000': 1.48672653012947
    '2015-12-19T00:00:01.000': 0.4729189469377924
  location:
    height: 2200.0
    lat: 28.7618172
    lon: -17.8913442
  mapping: nonlin
  mask: resources/cta_mask.png
  max_val: 65536
  radius: 847.5182294305233
  resolution:
    x: 1699
    y: 1699
  size:
    x: 1000
    y: 1000
  timestamps:
  - UTC1
  - TIMEUTC
  zenith:
    x: 848.2927316556994
    y: 848.6473285385366
iceact:
  az_offset: 2.954425486041723
  filter_star_detection:
    '1970-01-01T00:00:01.000': 0.23302372984015565
  llh_star_detection:
    '1970-01-01T00:00:01.000': 0.5113120911770741
  location:
    height: 2801.0
    lat: -89.99
    lon: -63.45
  mapping: lin
  mask: resources/iceact_mask.png
  max_val: 65536
  radius: 303.1264527743301
  resolution:
    x: 640
    y: 480
  size:
    x: 1000
    y: 1000
  timestamps:
  - DATE-OBS
  zenith:
    x: 326.5390254890759
    y: 249.69522492175076
```

DETERMINATION OF CAMERA PARAMETERS

Exact measurement of camera parameters difficult
→ Initialisation of camera parameters as measured
→ Optimisation of camera parameters by fit

```
from quocca.camera import Camera
from quocca.utilities import fit_camera_params

cam = Camera('cta')
fit_camera_params('2015_11_04-00_01_31.mat',
                  cam, update=True)
```

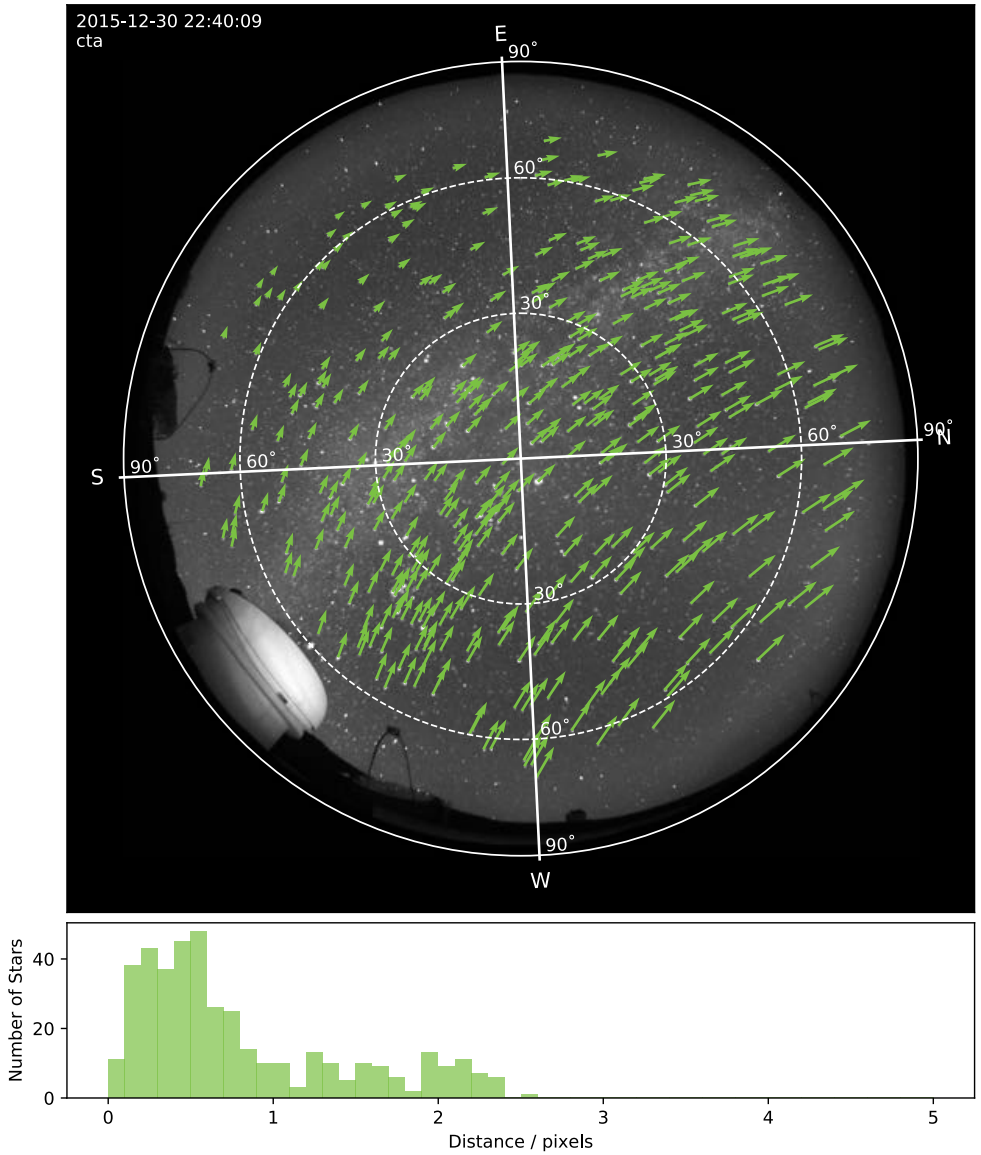




DETERMINATION OF CAMERA PARAMETERS

Residual Offsets

Comparison between projected and fitted positions



IMPORT OF IMAGES

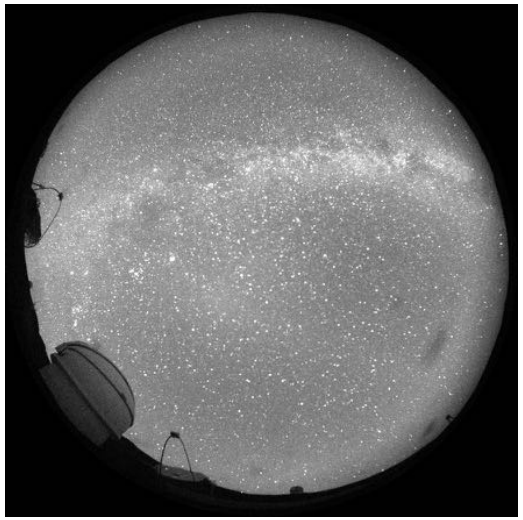
Supported file types:

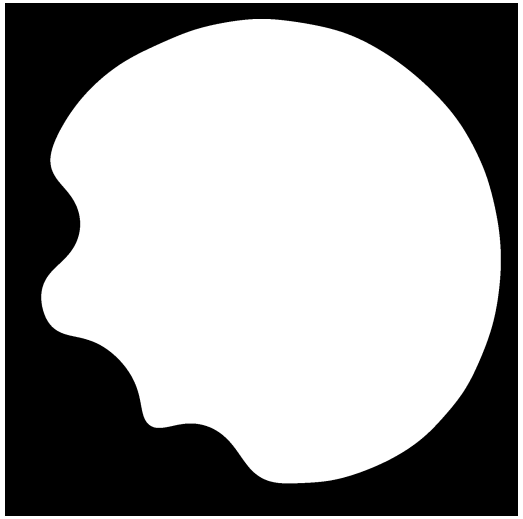
- *.fits
- *.mat

CATALOG STARS TO BE CONSIDERED

- Choice of star catalog
- Choice of catalog subset
  - Within image
  - Within environment mask
  - Specific magnitude
  - Specific variability
  - Specific altitude
  - Minimum distance between stars
    (if too close, only the brighter one is chosen)
  - Minimum distance to celestial bodies

```
img.add_catalog(catalog='hipparcos',
                max_mag=5.5, max_var=2
                min_alt=0, min_dist=10.0)
```





ESTIMATION OF STAR'S VISIBILITY

Method: Likelihood Fit

- Select only a subset of the image around the star
- Remove already fitted stars from image
- Apply a smoothing before the fit
- Fit two-dimensional Gaussian to each star
  with magnitude, background, position, size as parameters

$$f_{x_0,y_0,M,b,\sigma}(x,y) = M \cdot \exp(-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}) + b$$

$$LLH(x_0,y_0,M,b,\sigma|x_i,y_i) = \sum_i \left(f\left(x_i,y_i\right) - I\left(x_i,y_i\right)\right)^2$$

```
res = img.detect(method='llh',
                 sigma=1.6, fit_size=4, tol=1e-15,
                 presmoothing=0.5, remove_detected_stars=True)
```

ESTIMATION OF STAR'S VISIBILITY

Method: Laplacian of Gaussian Filter

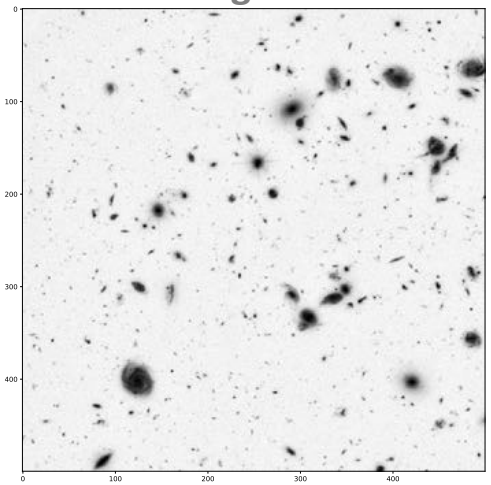
- Select only a subset of the image around the star
- Convolution of image with kernel
  (Kernel: Laplace operator applied to two-dimensional Gaussian)
- "Magnitude" = Specific quantile of filter response

$$f(x,y) = \frac{1}{2\pi\sigma^2}\exp(-\frac{x^2+y^2}{2\sigma^2})$$

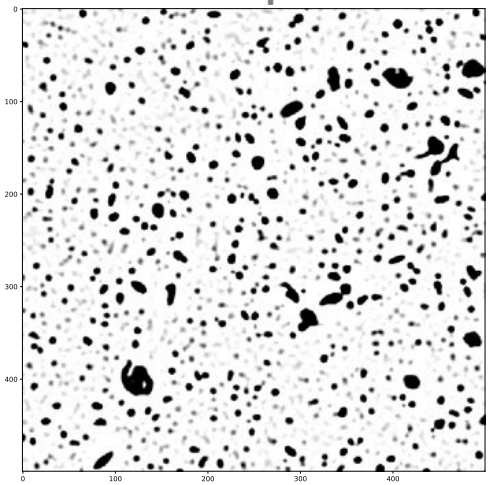
$$\Delta f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

```
res = img.detect(method='filter',
                 sigma=1.0, fit_size=4, quantile=100)
```

Original



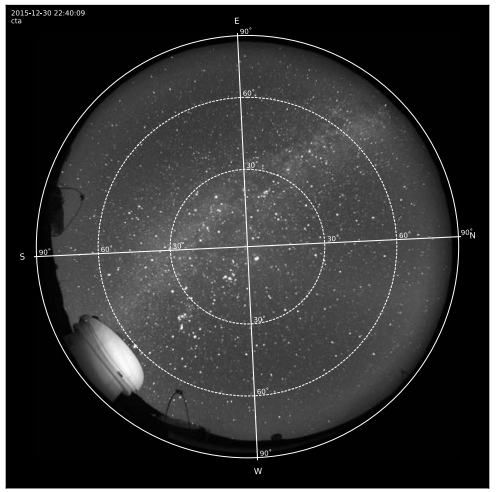
Filter Response

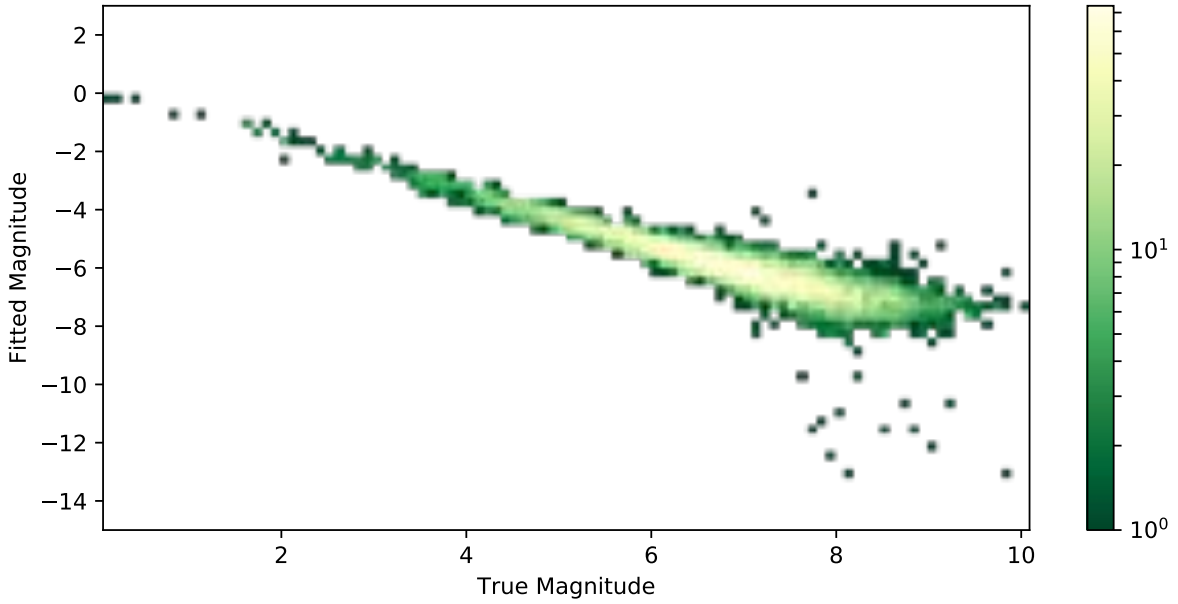


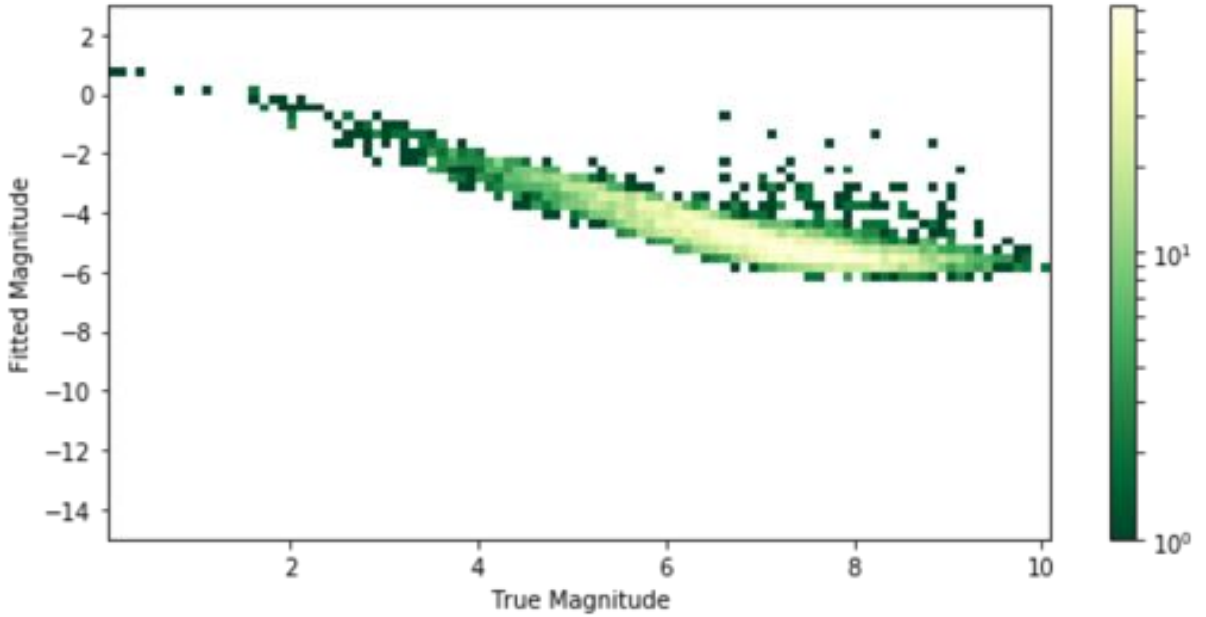
ESTIMATION OF STAR'S VISIBILITY

Comparison between True and Estimated Magnitude

Clear sky







CALIBRATION OF METHODS

Approach

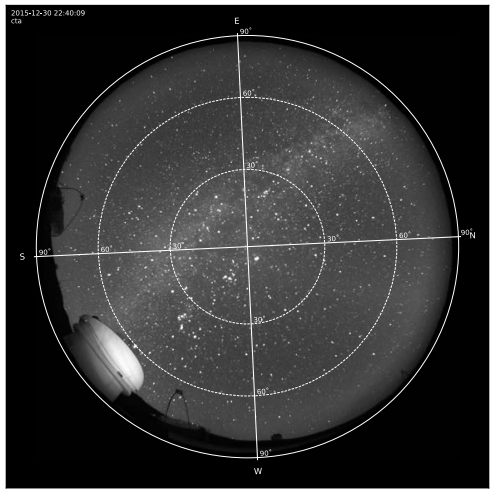
- Choose image with very clear sky
- Choose time span the calibration is valid for
- Determine calibration constants for each method
- Store constants in data base (cameras.yaml)

```
cta:
  az_offset: 87.2993699936737
  filter_star_detection:
    '1970-01-01T00:00:01.000': 0.2617892814129304
    '2015-12-19T00:00:01.000': 0.09033273267322699
  llh_star_detection:
    '1970-01-01T00:00:01.000': 1.4867742767618106
    '2015-12-19T00:00:01.000': 0.4729175929680222
```

```
from quocca.camera import Camera
from quocca.utilities import calibrate_method

cam = Camera('cta')
calibrate_method('2015_12_30-22_39_24.mat', cam,
                 method='filter',
                 time='2015-12-19T00:00:01.000',
                 update=True)
```

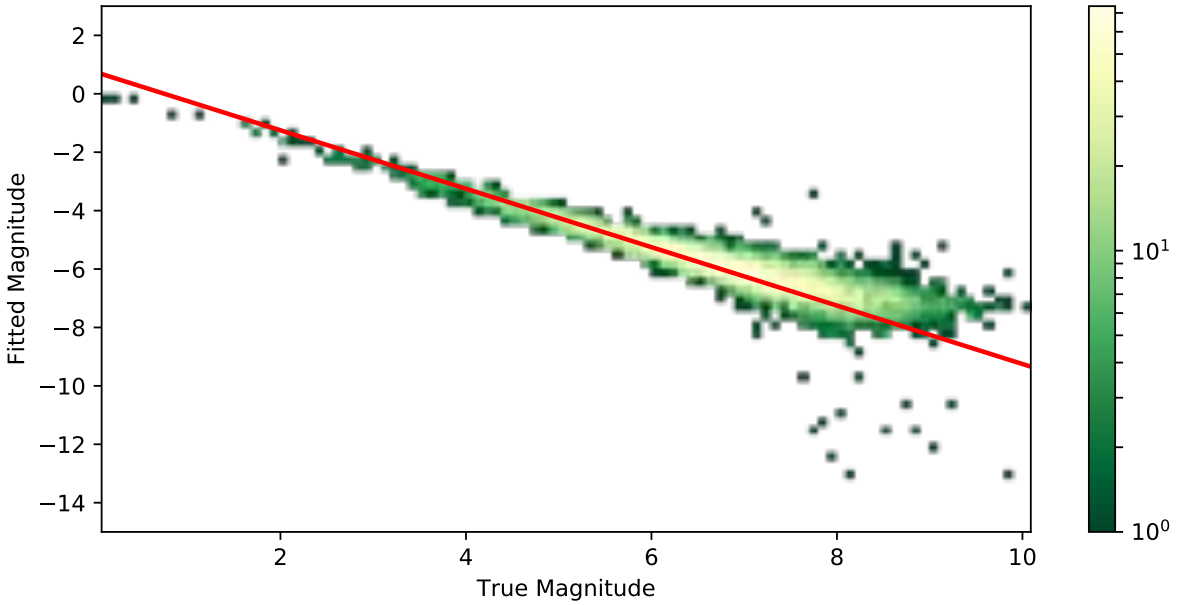
ESTIMATION OF STAR'S VISIBILITY

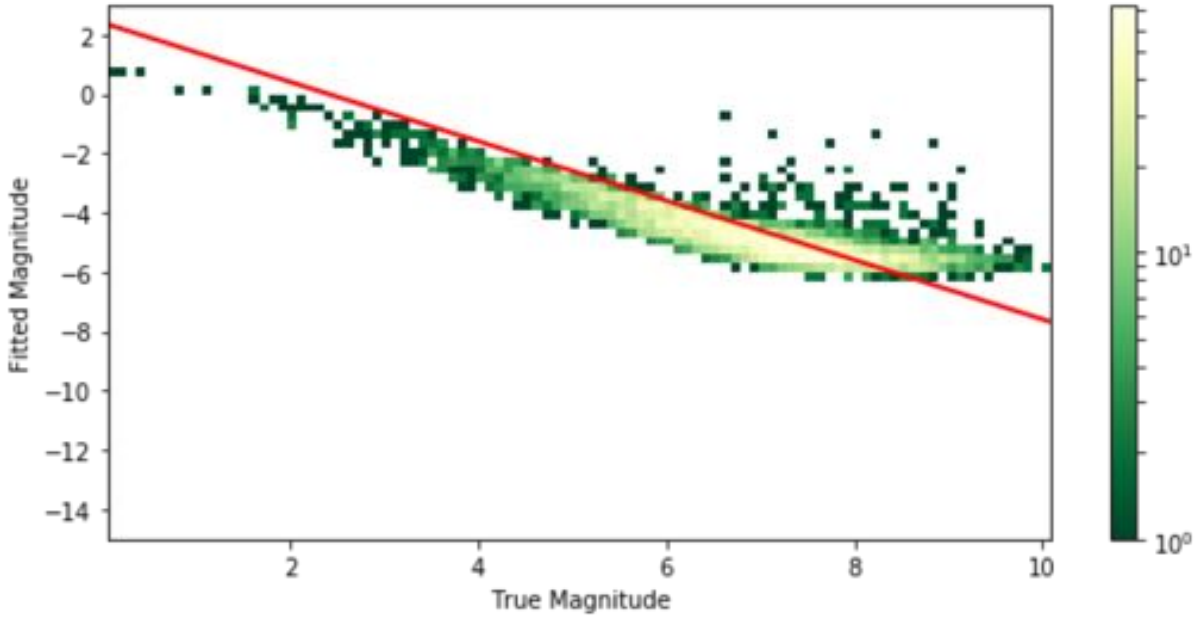
To Do: Polynomial calibration



Comparison between True and Estimated Magnitude

Clear sky

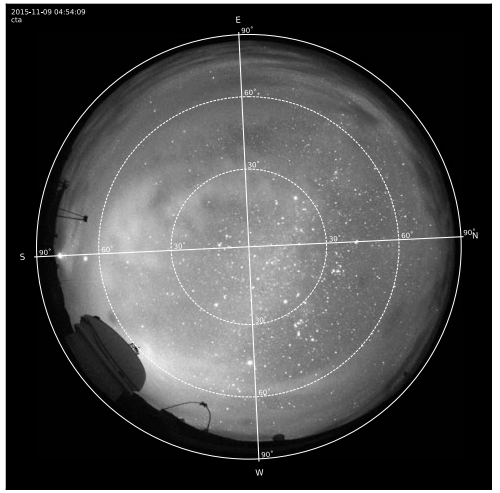


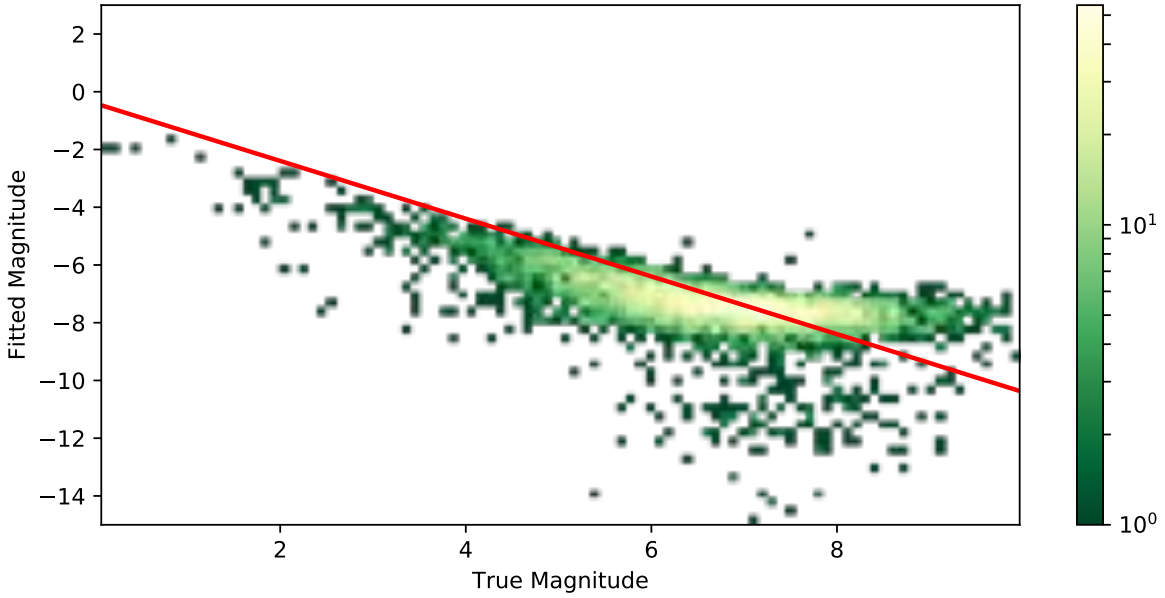


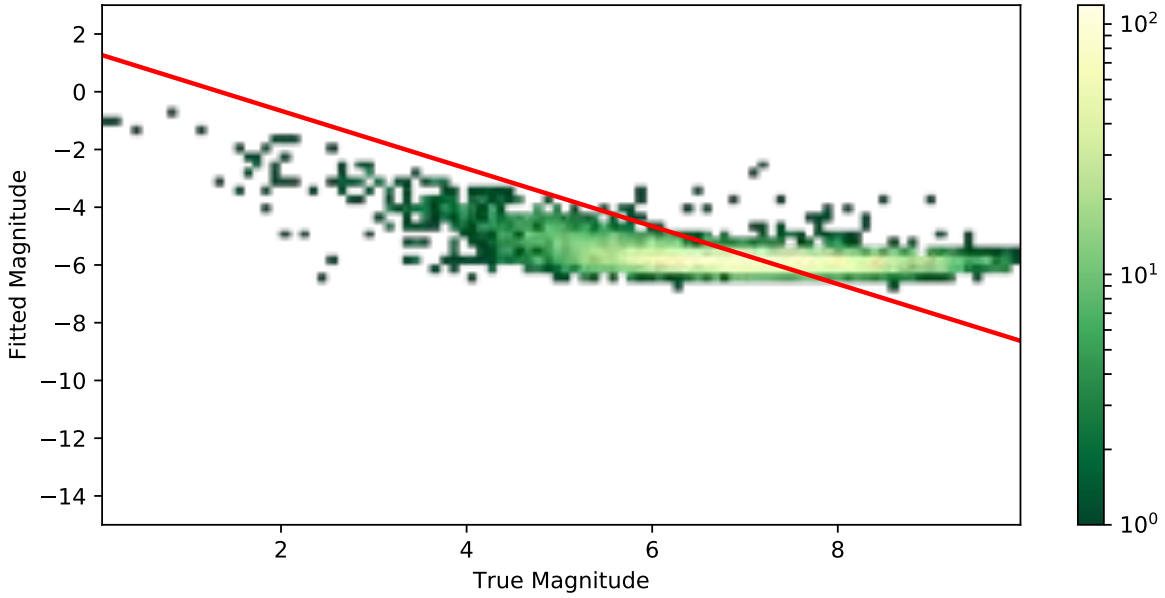
ESTIMATION OF STAR'S VISIBILITY

Comparison between True and Estimated Magnitude

Cloudy sky

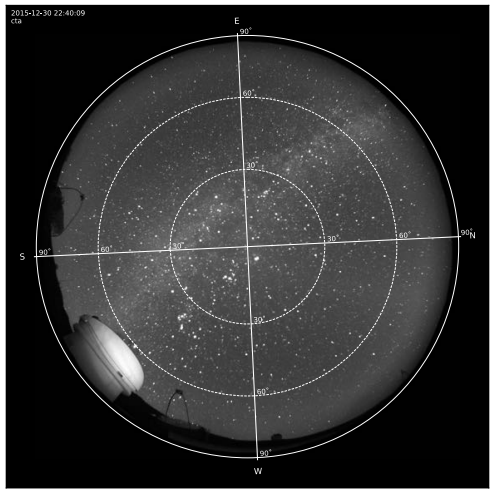






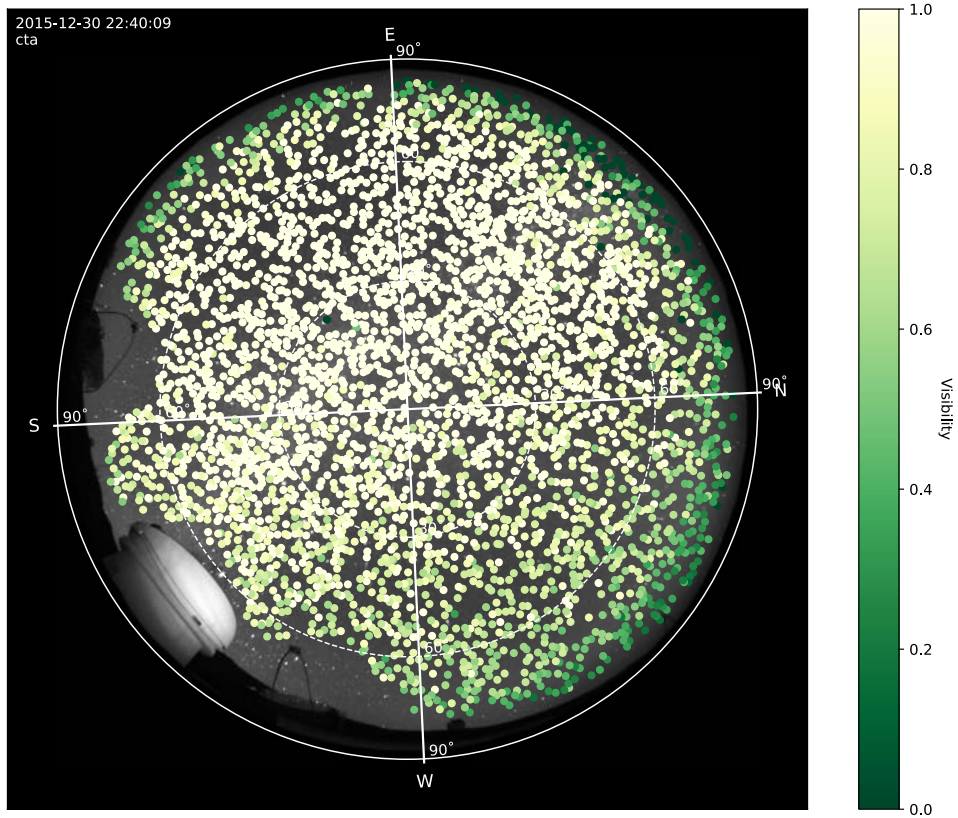
ESTIMATION OF STAR'S VISIBILITY

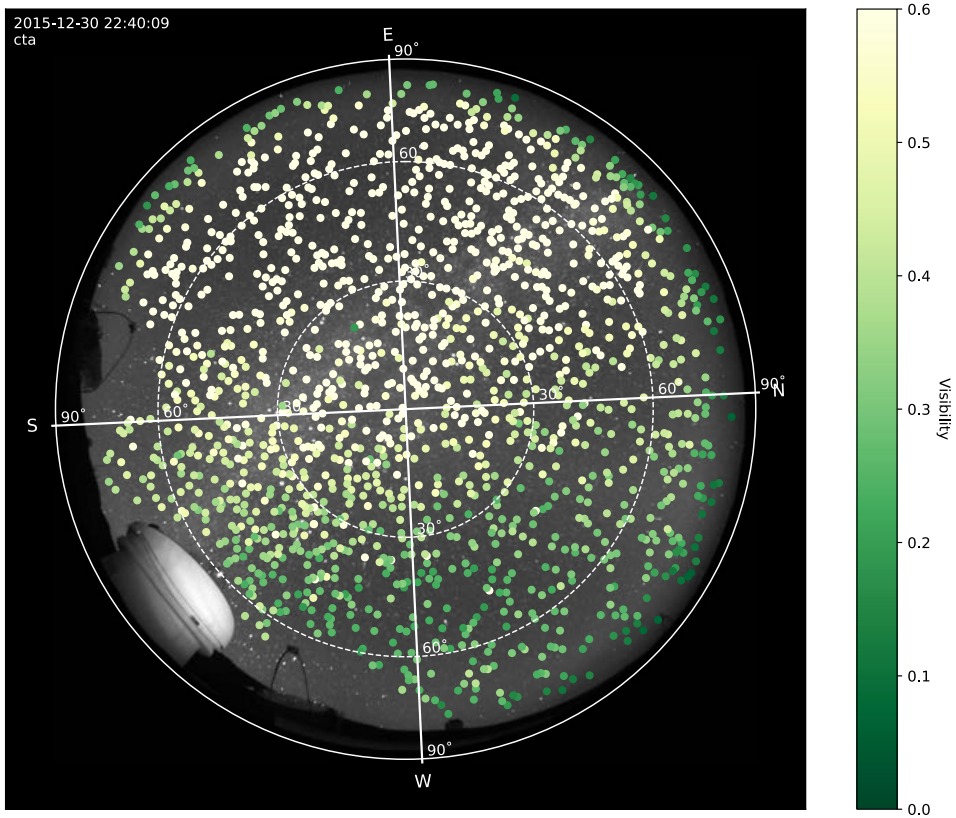
To Do: Correct zenith dependency



Distribution of Visibility on Sky Map

Clear sky

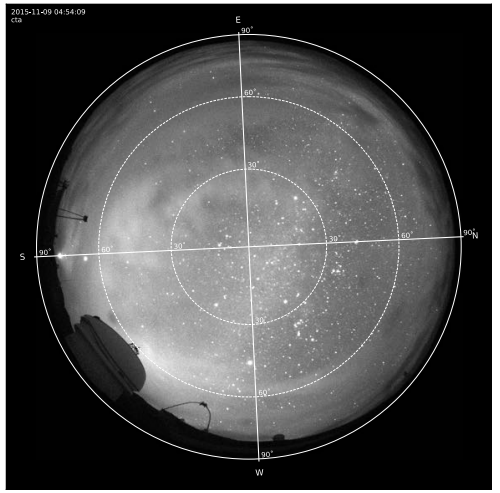


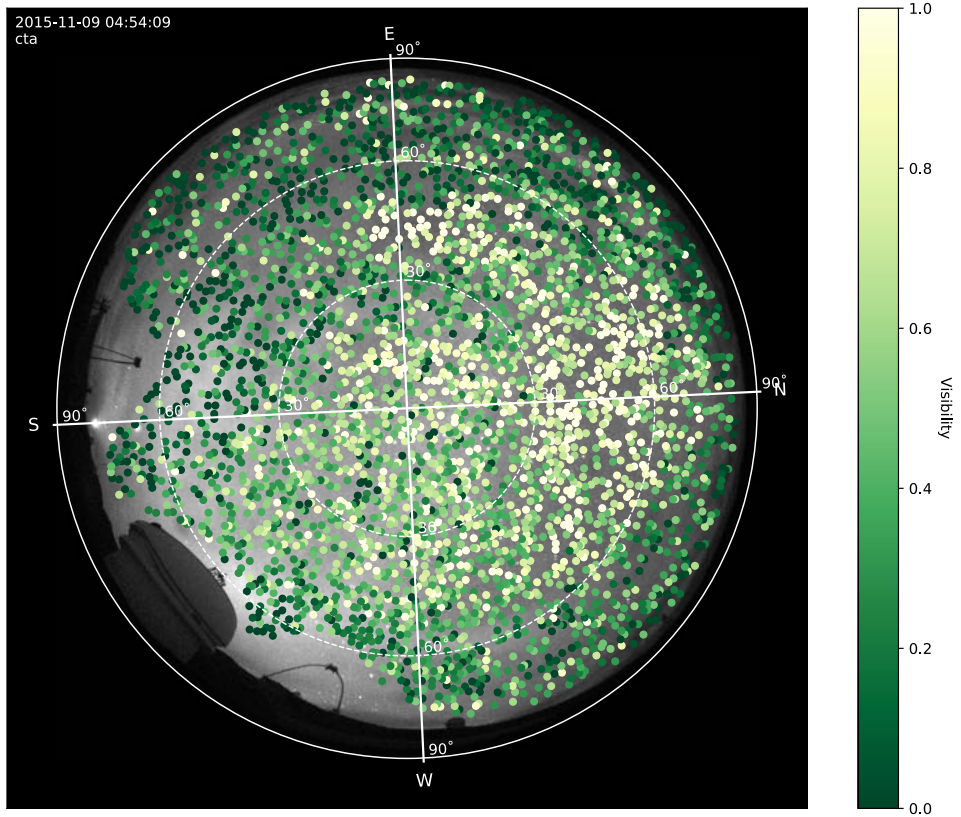


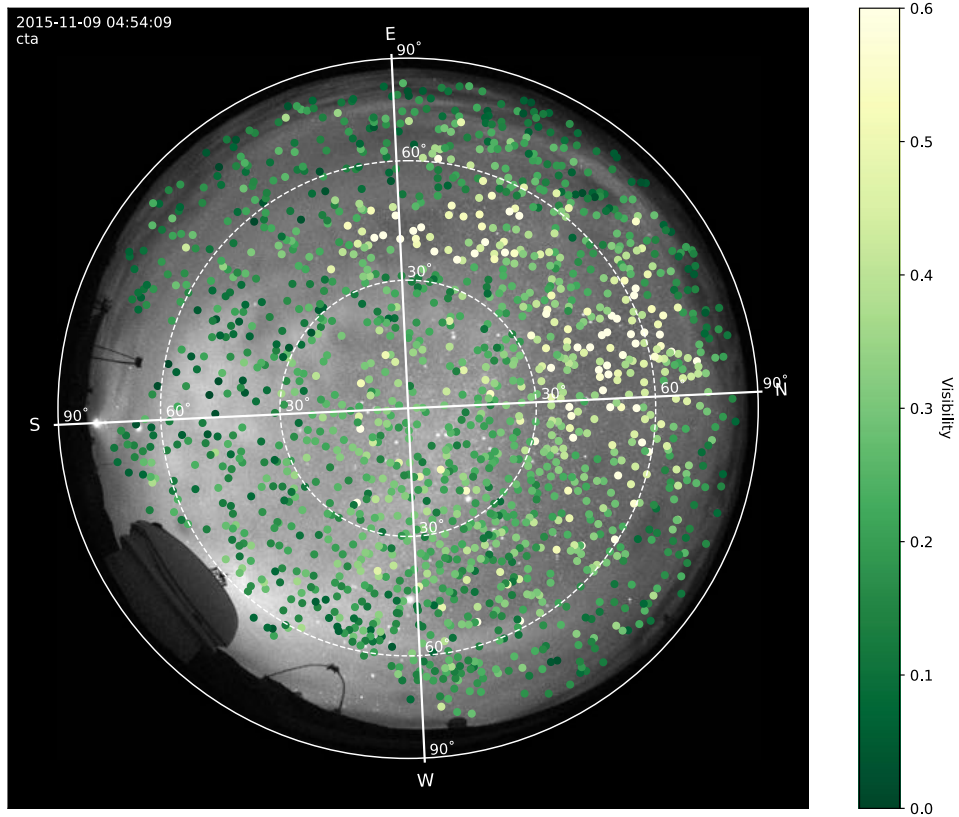
ESTIMATION OF STAR'S VISIBILITY

Distribution of Visibility on Sky Map

Cloudy sky







GENERATION OF CLOUD MAPS

- Input: Visibilities
- Different methods:
  - RunningAvg
  - GaussianRunningAvg
  - kNNMedian

```
from quocca.cloudiness import GaussianRunningAvg, cloud_map

cmap = cloud_map(res, cam, results.x_fit,
                 cloudiness_calc=GaussianRunningAvg,
                 smoothing=3, radius=25)
```

```
from quocca.cloudiness import KNNMedian, cloud_map

cmap = cloud_map(res, cam, results.x_fit,
                 cloudiness_calc=KNNMedian,
                 smoothing=3, k=5)
```

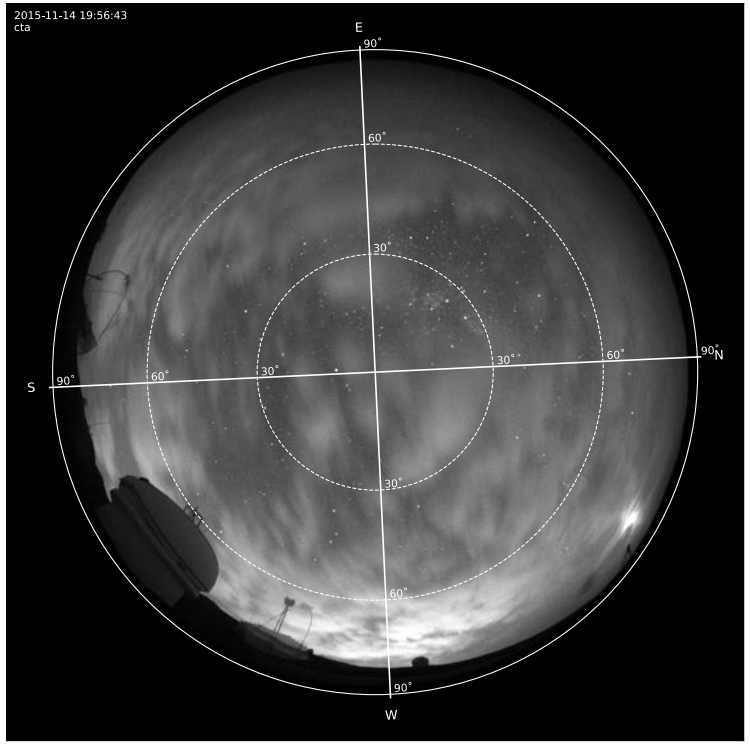
```
from quocca.plotting import show_img, show_clouds

fig, ax = plt.subplots()
ax = show_img(img, ax=ax)
ax = show_clouds(img, cmap, opaque=True, ax=ax)
```

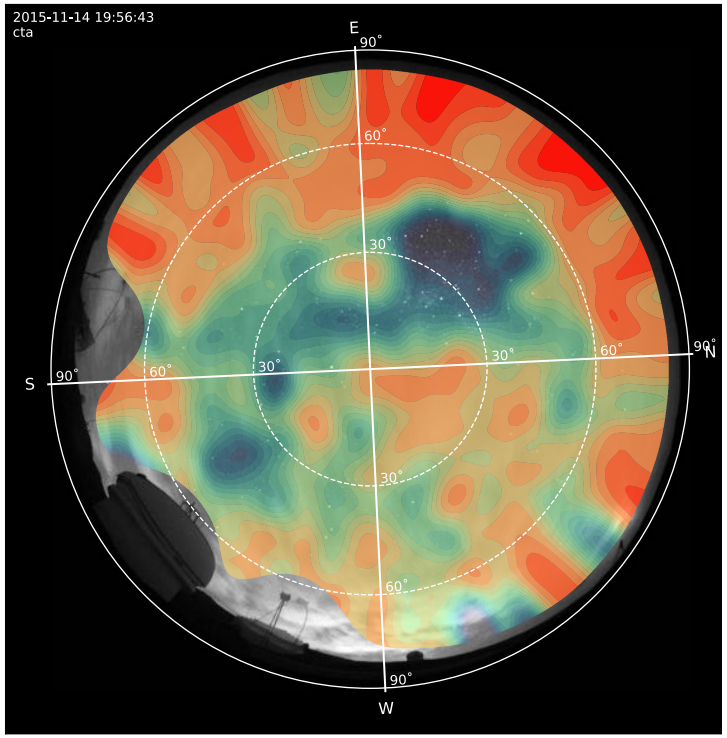
GENERATION OF CLOUD MAPS

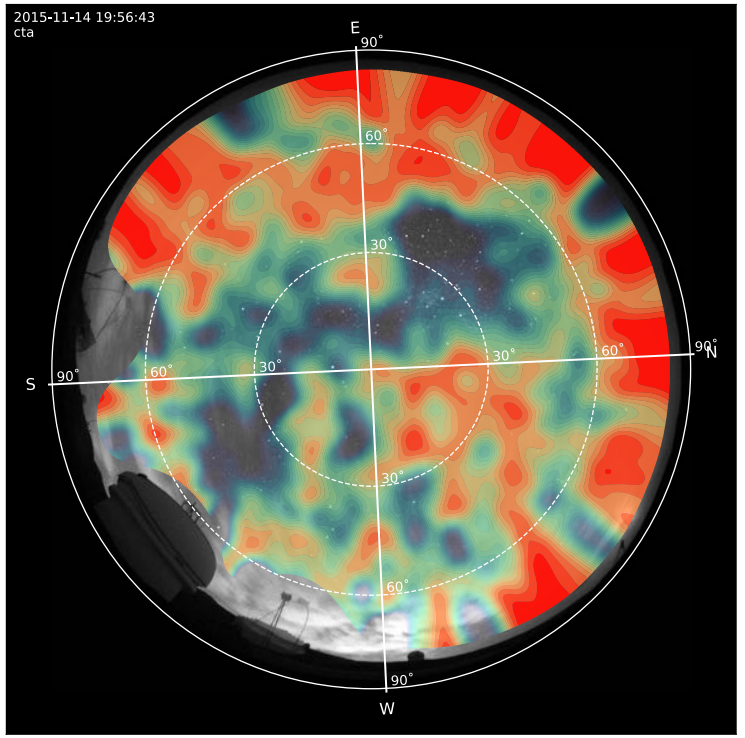
- Input: Visibilities
- Different methods:
  - RunningAvg
  - GaussianRunningAvg
  - kNNMedian

GaussianRunningAvg (r=25)

kNNMedian (k=5)



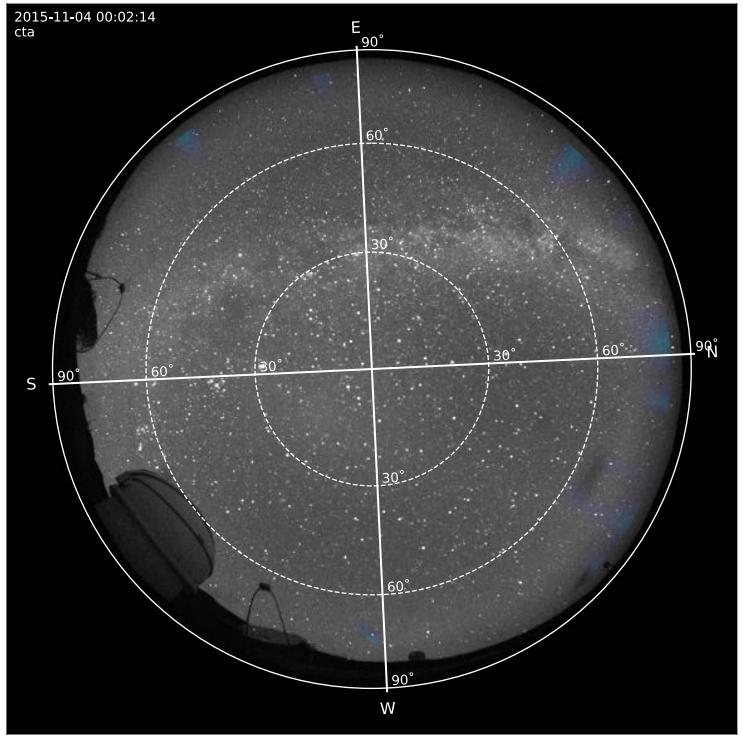




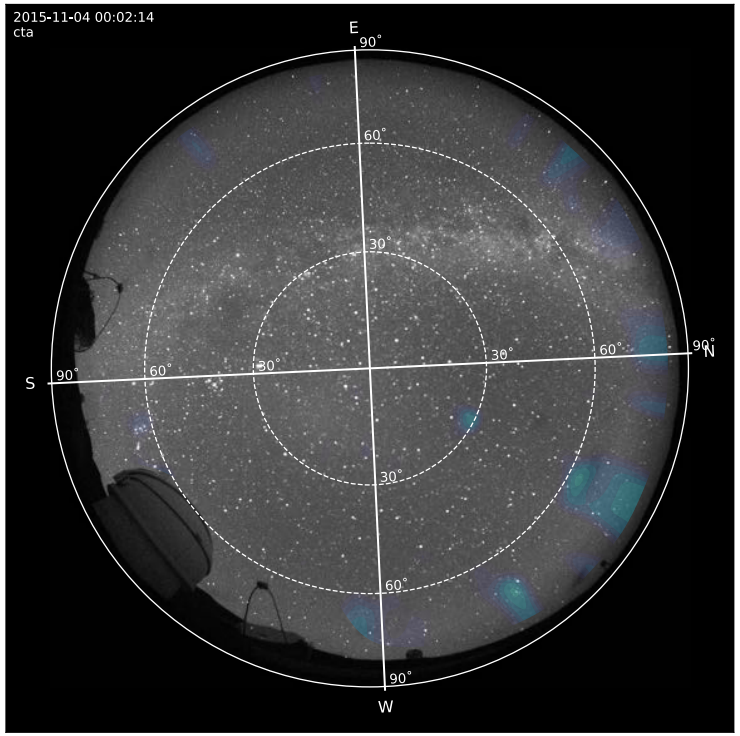
GENERATION OF CLOUD MAPS

- Input: Visibilities
- Different methods:
  - RunningAvg
  - GaussianRunningAvg
  - kNNMedian

GaussianRunningAvg (r=25)

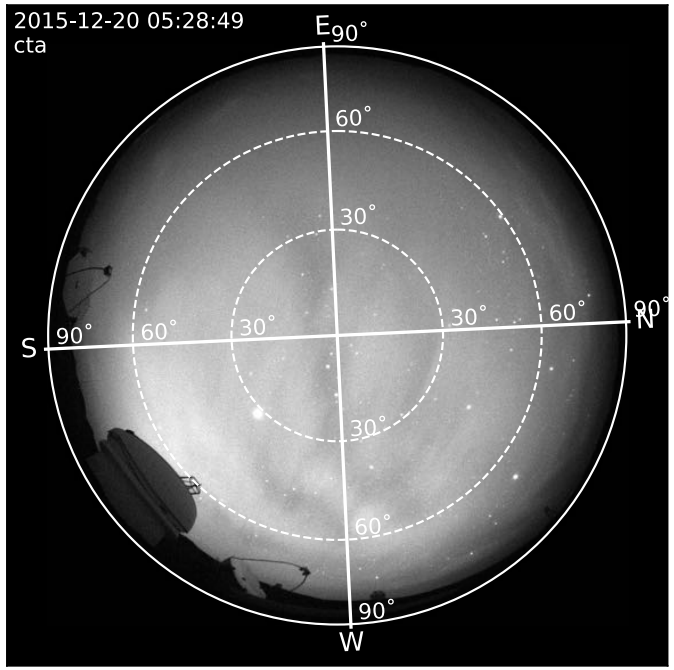


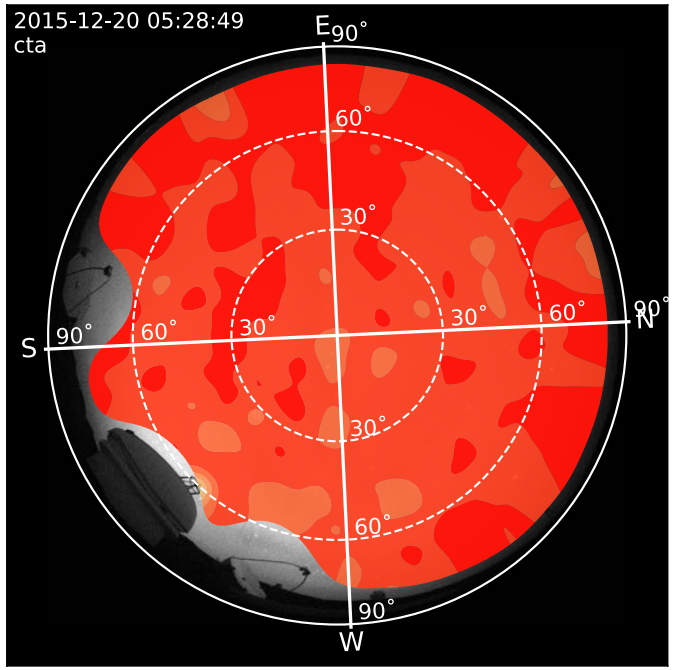
kNNMedian (k=5)

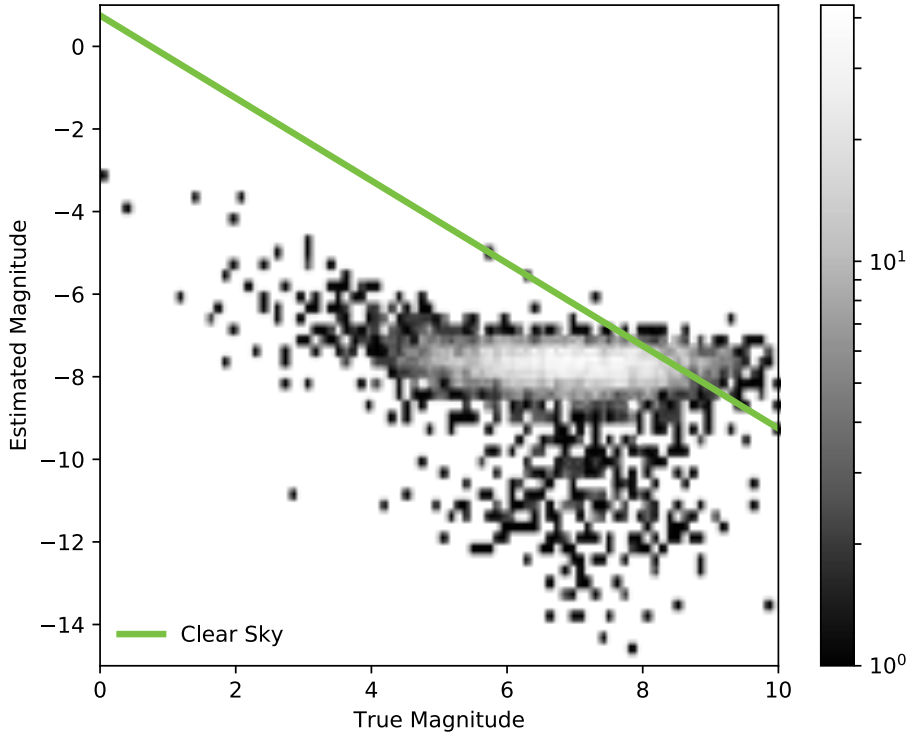


LIMITATIONS OF THE METHOD

Entire Cloudy Sky

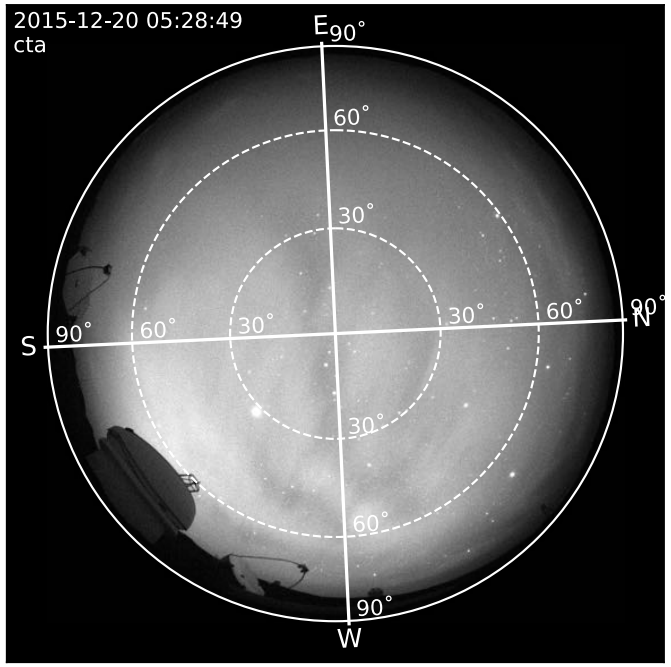


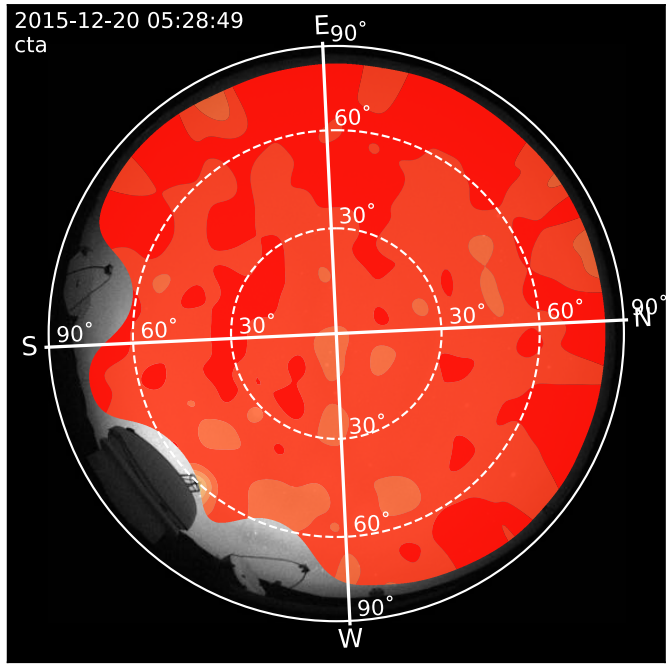


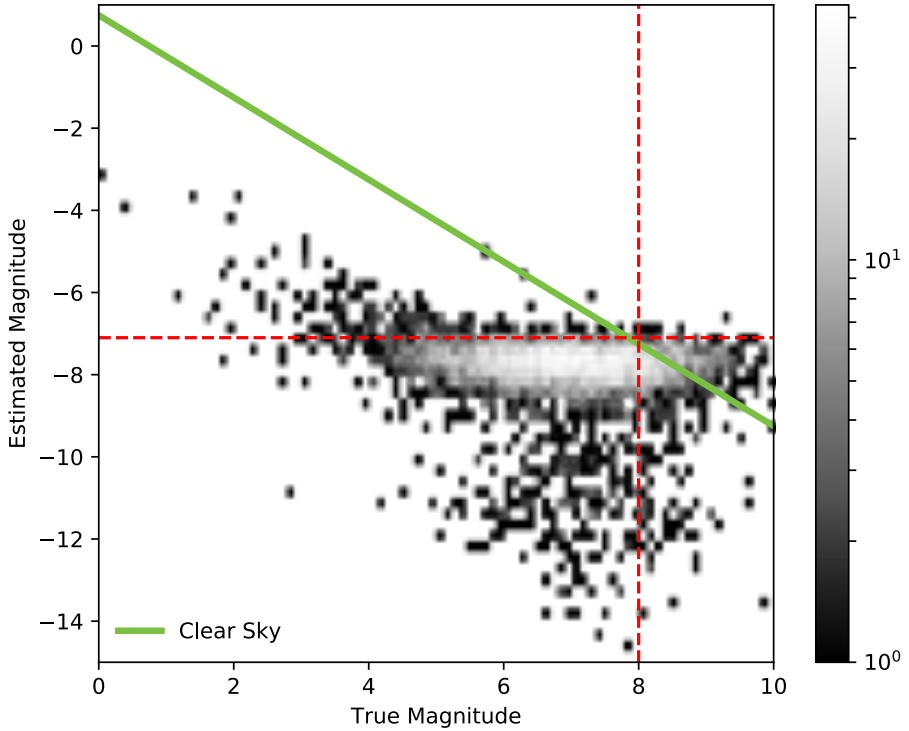


LIMITATIONS OF THE METHOD

Entire Cloudy Sky

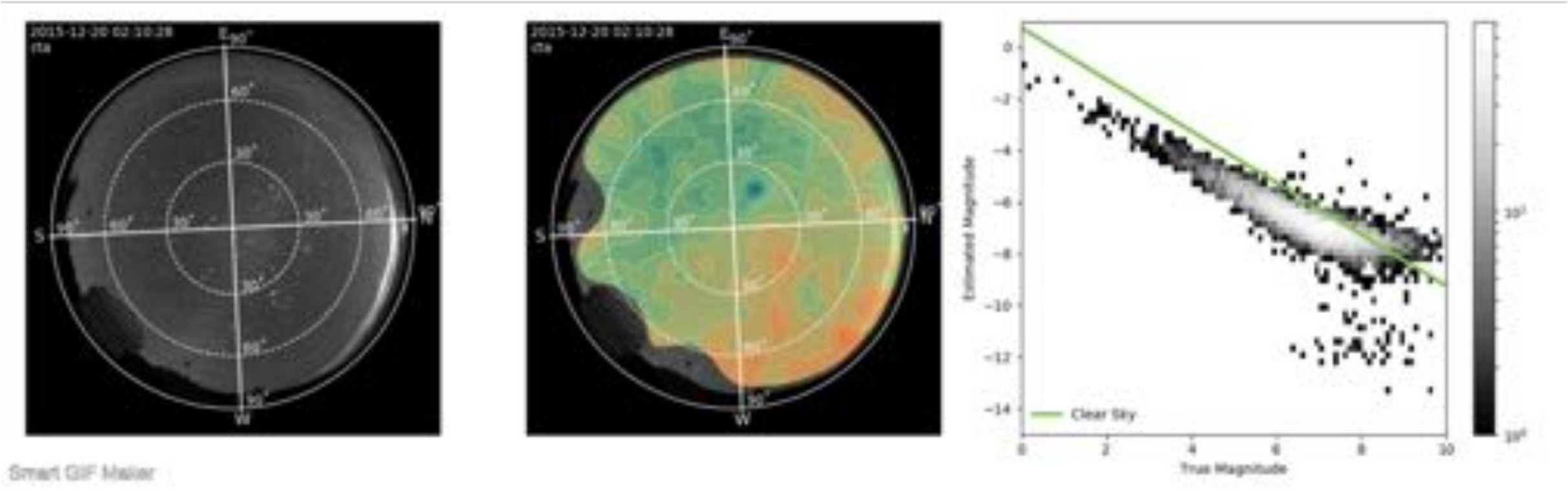






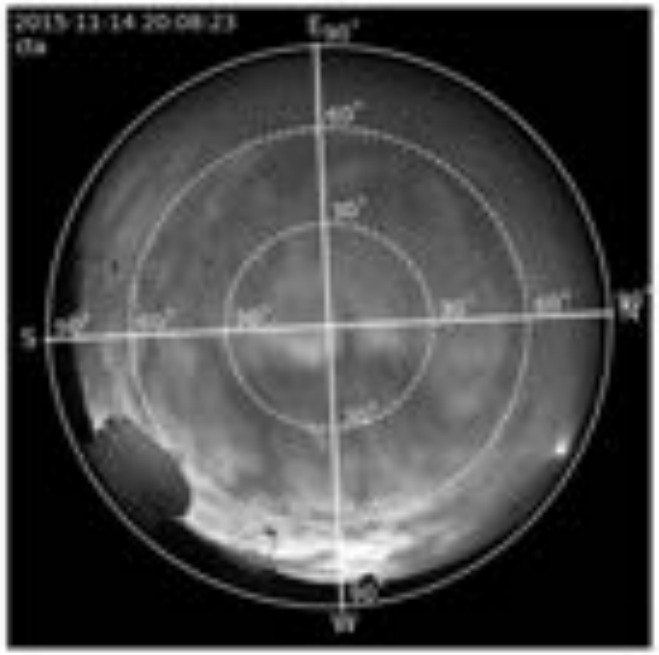
EXAMPLES OF CLOUD MAPS

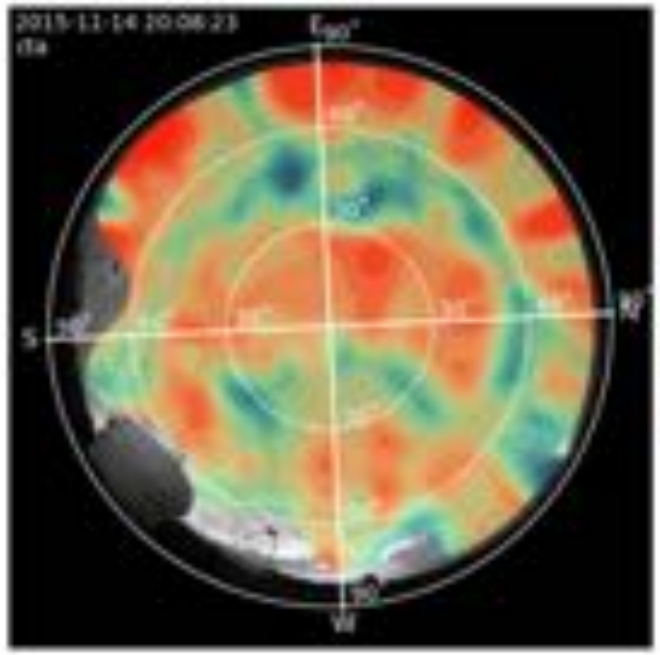
Dec 20, 2015

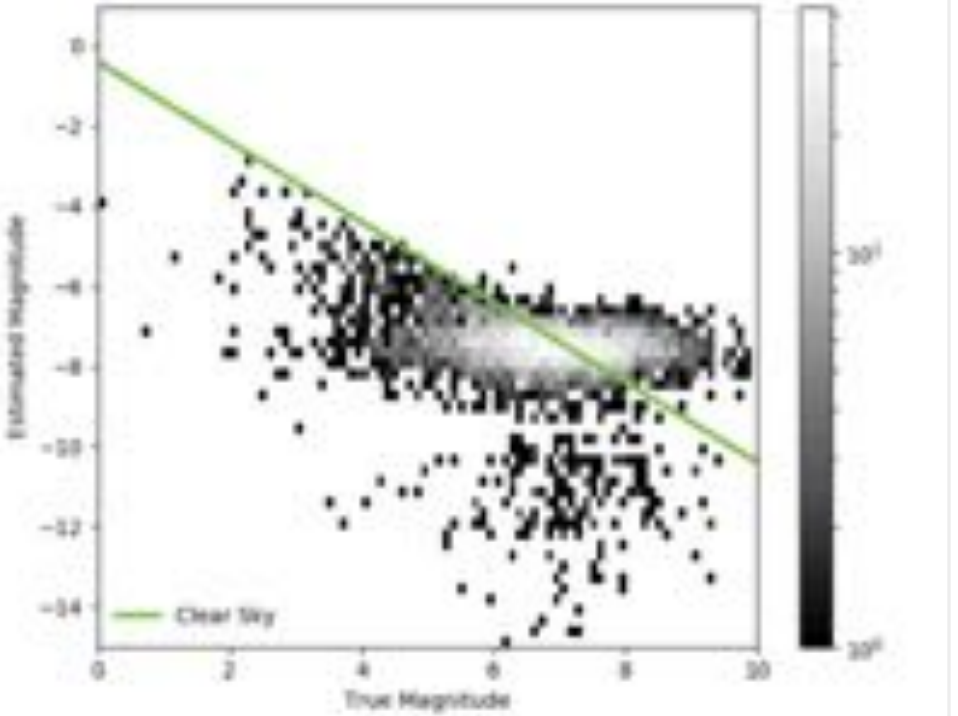


EXAMPLES OF CLOUD MAPS

Nov 14, 2015



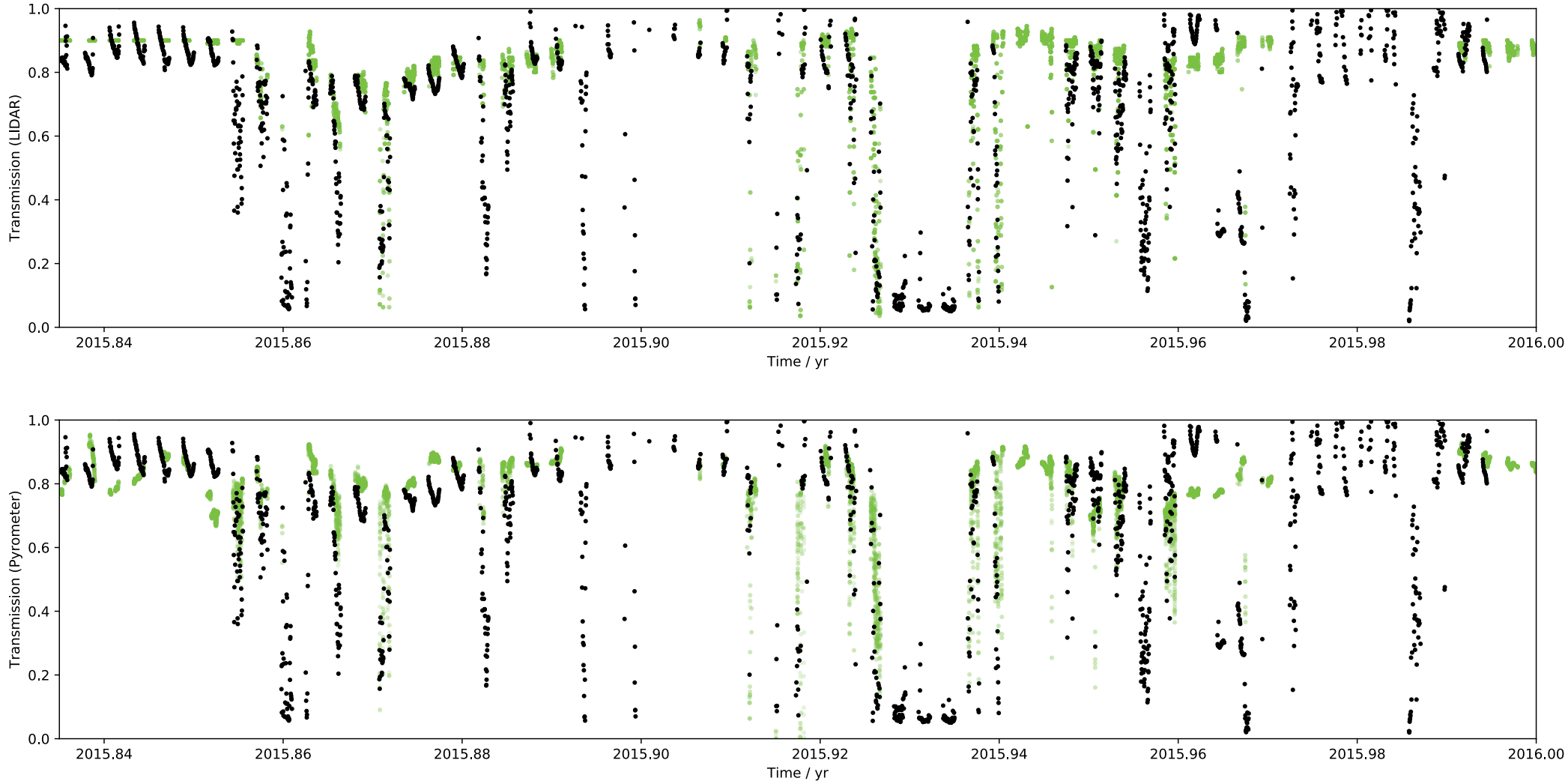




Smart GIF Maker

COMPARISON WITH OTHER INSTRUMENTS

Nov. & Dec. 2015



CONCLUSION AND FUTURE PROSPECTIVES

- Development of python module
- Development of new and more sensitive method to analyse all sky images
- Design ready for more complex implementations and studies
- Tested on images from CTA and IceAct

Next steps:

- Optimisation of settings and corrections
- Long-term processing
- Implementation of cloud motion prediction
- Implementation of cloud height determination
- Use for adaptive scheduling