



Finding optimal hyperparameters for cleaning algorithms for the Cherenkov Telescope Array

Bachelor thesis half-time talk

Anno Knierim

July 15, 2022

E5b Astroparticle Physics
Department of Physics – TU Dortmund



Table of contents

Introduction

- The Cherenkov Telescope Array
- CTAs low-level data processing pipeline software: `ctapipe`
- Cleaning Algorithms

Data Processing with `ctapipe`

Results

- ROC Curves
- Ratio of Surviving Pixels
- Metrics
- Angular Resolution
- Effective Area

Outlook and Summary

Introduction

The Cherenkov Telescope Array (CTA)

- 2 sites: CTA North and CTA South
- 3 types of telescopes:
 - Small-Sized Telescope (SST)
 - Medium-Sized Telescope (MST)
 - Large-Sized Telescope (LST)

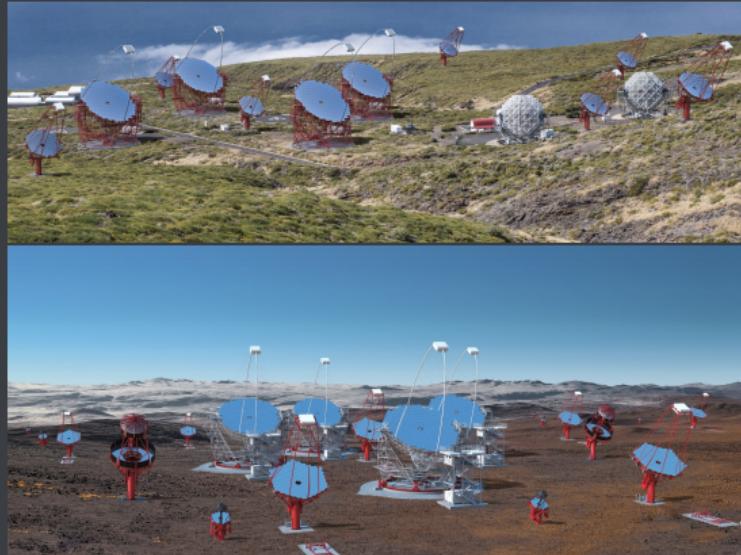
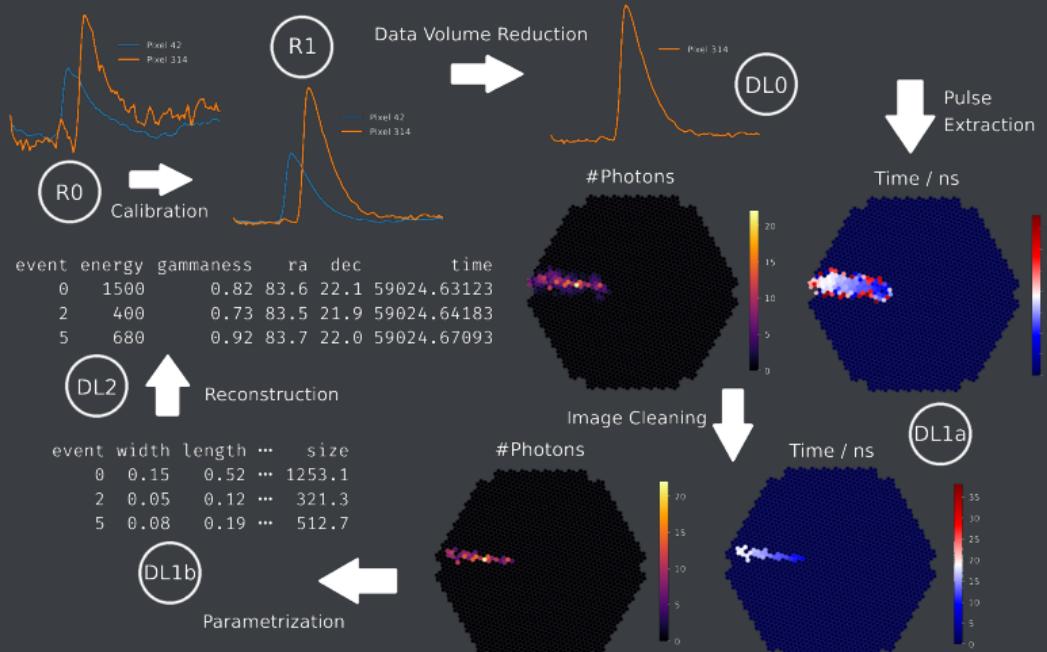


Image Credit: G. Pérez Diaz. CTA/ IAC. 2016. URL:
<https://www.cta-observatory.org/about/how-cta-works/>
(visited on 07/10/2022).

ctapipe



Adapted from J. Hackfeld and M. Nöthe. "Analyzing the Data Volume Reduction for the LST-1 Prototype of the Cherenkov Telescope Array." MA thesis. Bochum, 2021.

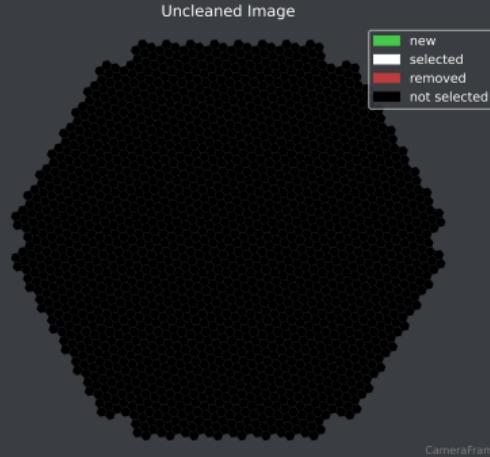
Cleaning Algorithms

- TailcutsImageCleaner
- MARSImageCleaner
- FACTImageCleaner
- TimeConstrainedImageCleaner

Cleaning Algorithms

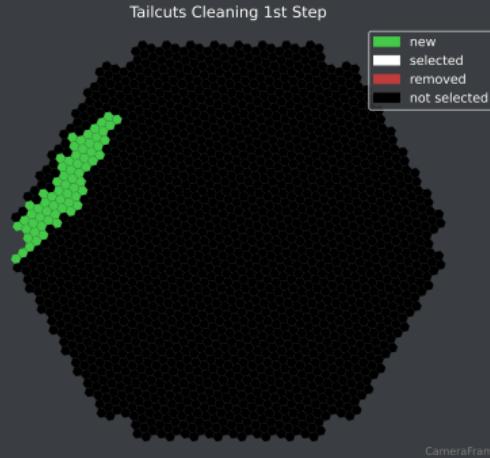
- TailcutsImageCleaner
 - 1. Selects pixels that pass the picture threshold
 - 2. Adds pixels that pass the boundary threshold
- MARSImageCleaner
- FACTImageCleaner
- TimeConstrainedImageCleaner

Cleaning Algorithms



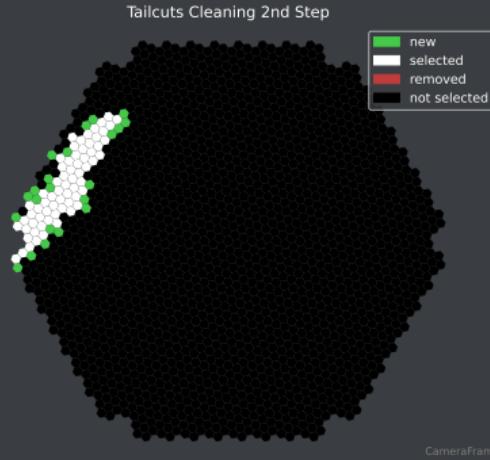
1. Selects pixels that pass the picture threshold
2. Adds pixels that pass the boundary threshold

Cleaning Algorithms



1. Selects pixels that pass the picture threshold
2. Adds pixels that pass the boundary threshold

Cleaning Algorithms

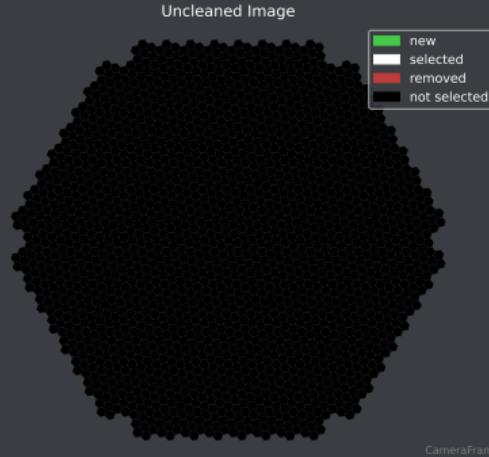


1. Selects pixels that pass the picture threshold
2. Adds pixels that pass the boundary threshold

Cleaning Algorithms

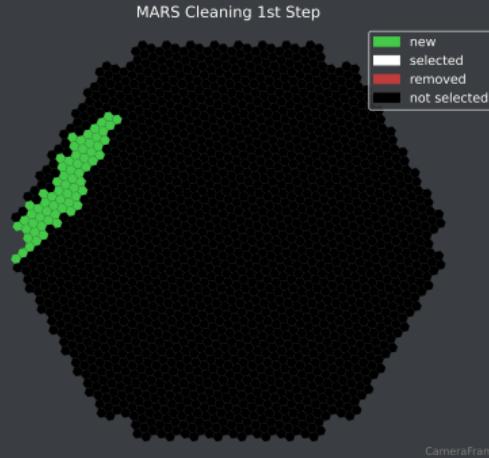
- TailcutsImageCleaner
 - MARSImageCleaner
 - FACTImageCleaner
 - TimeConstrainedImageCleaner
1. Selects pixels that pass a picture and boundary threshold, analogous to TailcutsImageCleaner
 2. Adds pixels that are a neighbor of a neighbor of a core pixel, if they are above the boundary threshold

Cleaning Algorithms



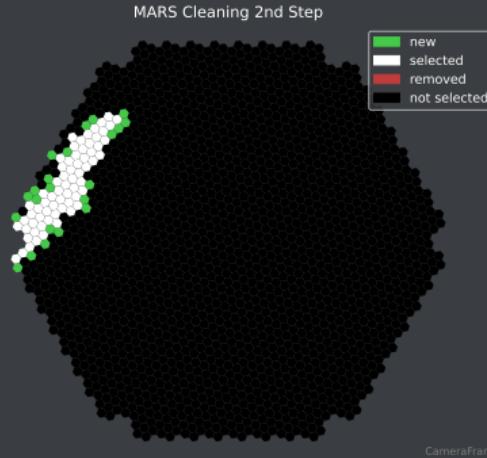
1. Selects pixels that pass a picture and boundary threshold, analogous to TailcutsImageCleaner
2. Adds pixels that are a neighbor of a neighbor of a core pixel, if they are above the boundary threshold

Cleaning Algorithms



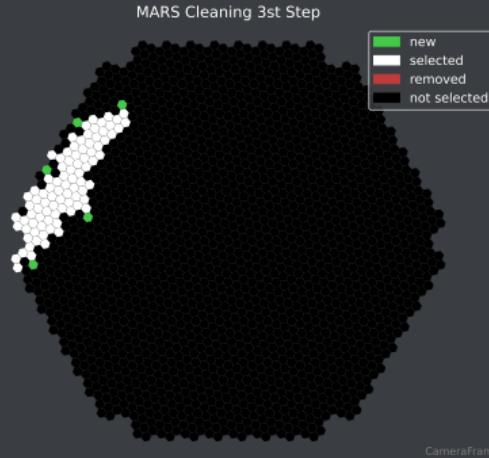
1. Selects pixels that pass a picture and boundary threshold, analogous to TailcutsImageCleaner
2. Adds pixels that are a neighbor of a neighbor of a core pixel, if they are above the boundary threshold

Cleaning Algorithms



1. Selects pixels that pass a picture and boundary threshold, analogous to TailcutsImageCleaner
2. Adds pixels that are a neighbor of a neighbor of a core pixel, if they are above the boundary threshold

Cleaning Algorithms



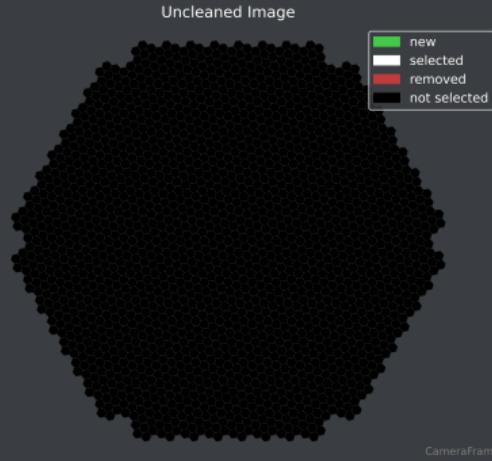
1. Selects pixels that pass a picture and boundary threshold, analogous to TailcutsImageCleaner
2. Adds pixels that are a neighbor of a neighbor of a core pixel, if they are above the boundary threshold

Cleaning Algorithms

- TailcutsImageCleaner
- MARSImageCleaner
- FACTImageCleaner
- TimeConstrainedImageCleaner

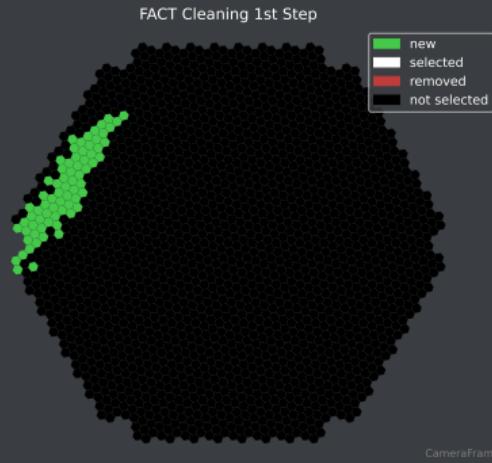
1. Finds all pixels that contain more photons than the picture threshold
2. Removes pixels with less than N neighbors
3. Adds remaining neighbors that are above the boundary threshold
4. Removes pixels that have less than N neighbors, that arrive within a given timeframe (here: 5 ns)
5. Removes pixels that have less than N neighbors
6. Removes pixels that have less than N neighbors, arriving within a given timeframe (same as in step 4)

Cleaning Algorithms



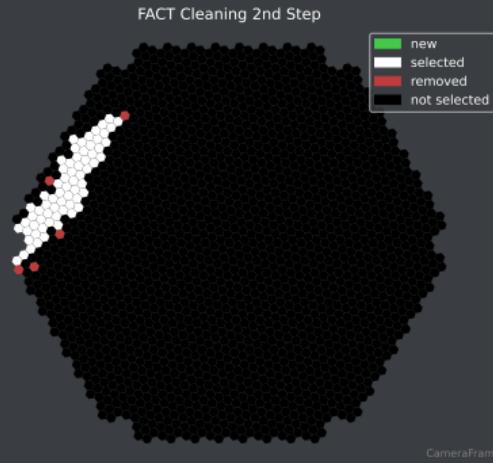
1. Finds all pixels that contain more photons than the picture threshold
2. Removes pixels with less than N neighbors
3. Adds remaining neighbors that are above the boundary threshold
4. Removes pixels that have less than N neighbors, that arrive within a given timeframe (here: 5 ns)
5. Removes pixels that have less than N neighbors
6. Removes pixels that have less than N neighbors, arriving within a given timeframe (same as in step 4)

Cleaning Algorithms



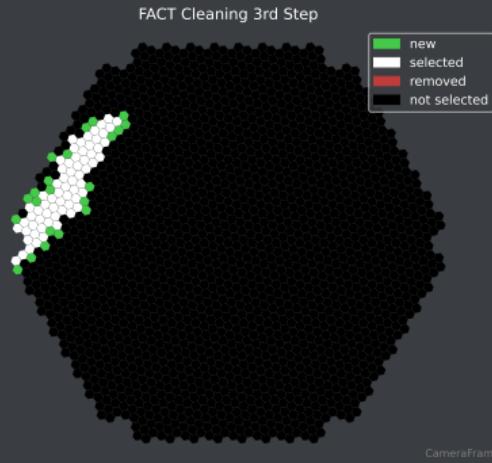
1. Finds all pixels that contain more photons than the picture threshold
2. Removes pixels with less than N neighbors
3. Adds remaining neighbors that are above the boundary threshold
4. Removes pixels that have less than N neighbors, that arrive within a given timeframe (here: 5 ns)
5. Removes pixels that have less than N neighbors
6. Removes pixels that have less than N neighbors, arriving within a given timeframe (same as in step 4)

Cleaning Algorithms



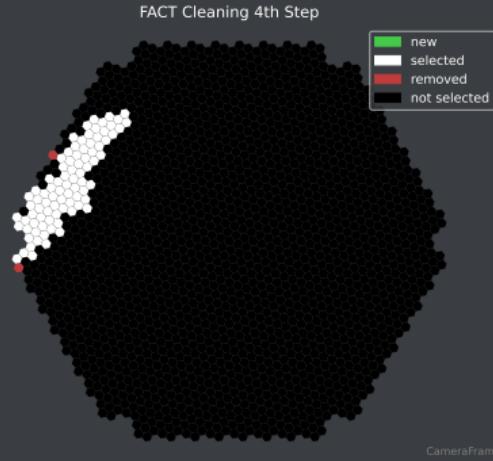
1. Finds all pixels that contain more photons than the picture threshold
2. Removes pixels with less than N neighbors
3. Adds remaining neighbors that are above the boundary threshold
4. Removes pixels that have less than N neighbors, that arrive within a given timeframe (here: 5 ns)
5. Removes pixels that have less than N neighbors
6. Removes pixels that have less than N neighbors, arriving within a given timeframe (same as in step 4)

Cleaning Algorithms



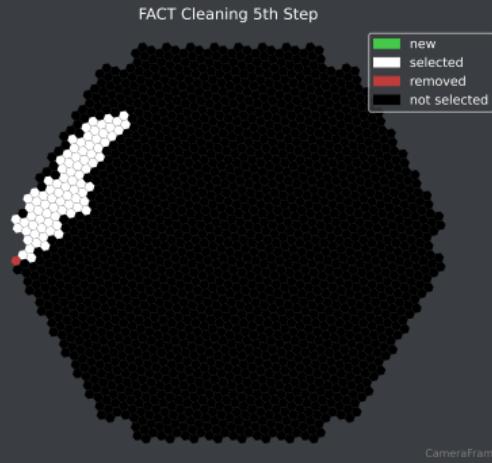
1. Finds all pixels that contain more photons than the picture threshold
2. Removes pixels with less than N neighbors
3. Adds remaining neighbors that are above the boundary threshold
4. Removes pixels that have less than N neighbors, that arrive within a given timeframe (here: 5 ns)
5. Removes pixels that have less than N neighbors
6. Removes pixels that have less than N neighbors, arriving within a given timeframe (same as in step 4)

Cleaning Algorithms



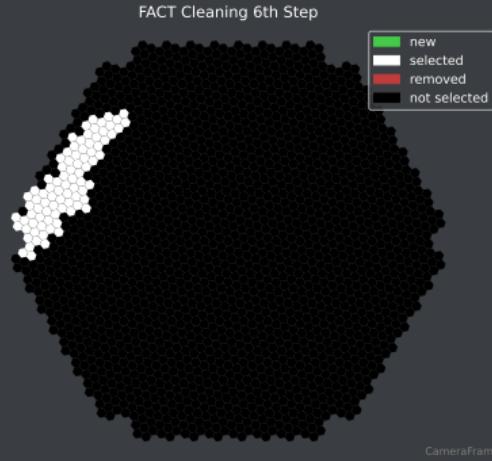
1. Finds all pixels that contain more photons than the picture threshold
2. Removes pixels with less than N neighbors
3. Adds remaining neighbors that are above the boundary threshold
4. Removes pixels that have less than N neighbors, that arrive within a given timeframe (here: 5 ns)
5. Removes pixels that have less than N neighbors
6. Removes pixels that have less than N neighbors, arriving within a given timeframe (same as in step 4)

Cleaning Algorithms



1. Finds all pixels that contain more photons than the picture threshold
2. Removes pixels with less than N neighbors
3. Adds remaining neighbors that are above the boundary threshold
4. Removes pixels that have less than N neighbors, that arrive within a given timeframe (here: 5 ns)
5. Removes pixels that have less than N neighbors
6. Removes pixels that have less than N neighbors, arriving within a given timeframe (same as in step 4)

Cleaning Algorithms



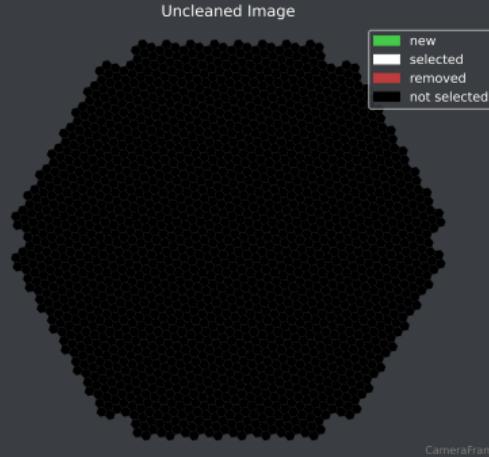
1. Finds all pixels that contain more photons than the picture threshold
2. Removes pixels with less than N neighbors
3. Adds remaining neighbors that are above the boundary threshold
4. Removes pixels that have less than N neighbors, that arrive within a given timeframe (here: 5 ns)
5. Removes pixels that have less than N neighbors
6. Removes pixels that have less than N neighbors, arriving within the given timeframe (same as in step 4)

Cleaning Algorithms

- TailcutsImageCleaner
- MARSImageCleaner
- FACTImageCleaner
- TimeConstrainedImageCleaner

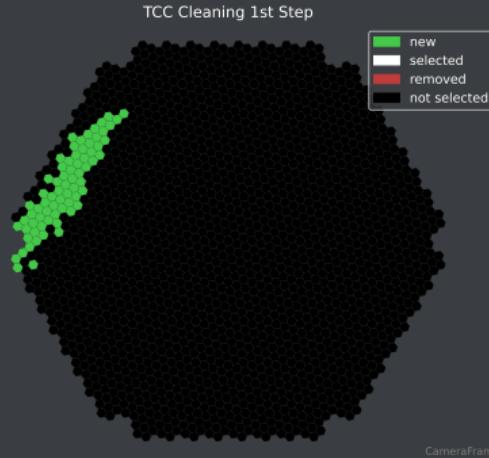
1. Finds all core pixels above the `picture threshold`
2. Removes pixels with less than N neighbors
3. Keeps all pixels that arrive within a time limit of the average arrival time (`time_limit_core`: 4.5 ns)
4. Finds all neighboring pixels above the `boundary threshold`
5. Removes all pixels with less than N neighbors arriving within a given timeframe (`time_limit_boundary`: 1.5 ns)

Cleaning Algorithms



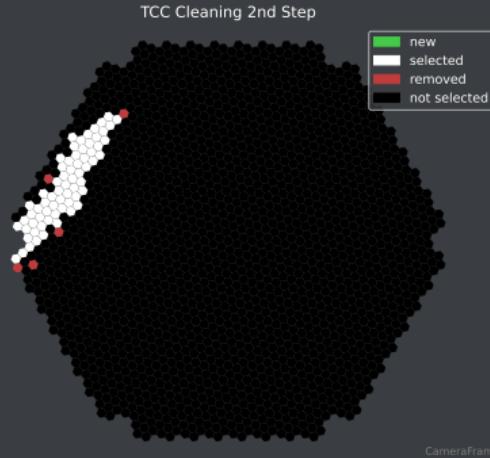
1. Finds all core pixels above the `picture threshold`
2. Removes pixels with less than N neighbors
3. Keeps all pixels that arrive within a time limit of the average arrival time (`time_limit_core`: 4.5 ns)
4. Finds all neighboring pixels above the `boundary threshold`
5. Removes all pixels with less than N neighbors arriving within a given timeframe (`time_limit_boundary`: 1.5 ns)

Cleaning Algorithms



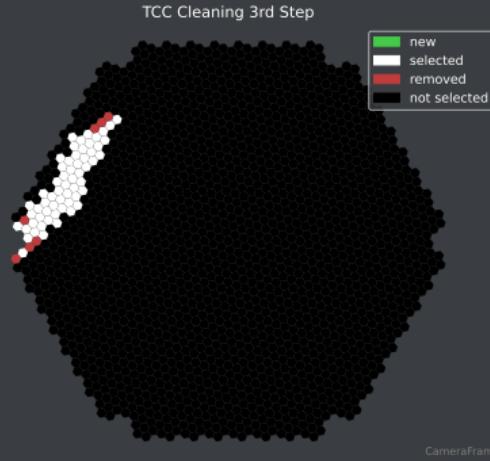
1. Finds all core pixels above the `picture threshold`
2. Removes pixels with less than N neighbors
3. Keeps all pixels that arrive within a time limit of the average arrival time (`time_limit_core`: 4.5 ns)
4. Finds all neighboring pixels above the `boundary threshold`
5. Removes all pixels with less than N neighbors arriving within a given timeframe (`time_limit_boundary`: 1.5 ns)

Cleaning Algorithms



1. Finds all core pixels above the `picture threshold`
2. Removes pixels with less than N neighbors
3. Keeps all pixels that arrive within a time limit of the average arrival time (`time_limit_core`: 4.5 ns)
4. Finds all neighboring pixels above the `boundary threshold`
5. Removes all pixels with less than N neighbors arriving within a given timeframe (`time_limit_boundary`: 1.5 ns)

Cleaning Algorithms



1. Finds all core pixels above the `picture threshold`
2. Removes pixels with less than N neighbors
3. Keeps all pixels that arrive within a time limit of the average arrival time (`time_limit_core`: 4.5 ns)
4. Finds all neighboring pixels above the `boundary threshold`
5. Removes all pixels with less than N neighbors arriving within a given timeframe (`time_limit_boundary`: 1.5 ns)

Cleaning Algorithms



1. Finds all core pixels above the `picture threshold`
2. Removes pixels with less than N neighbors
3. Keeps all pixels that arrive within a time limit of the average arrival time (`time_limit_core`: 4.5 ns)
4. Finds all neighboring pixels above the `boundary threshold`
5. Removes all pixels with less than N neighbors arriving within a given timeframe (`time_limit_boundary`: 1.5 ns)

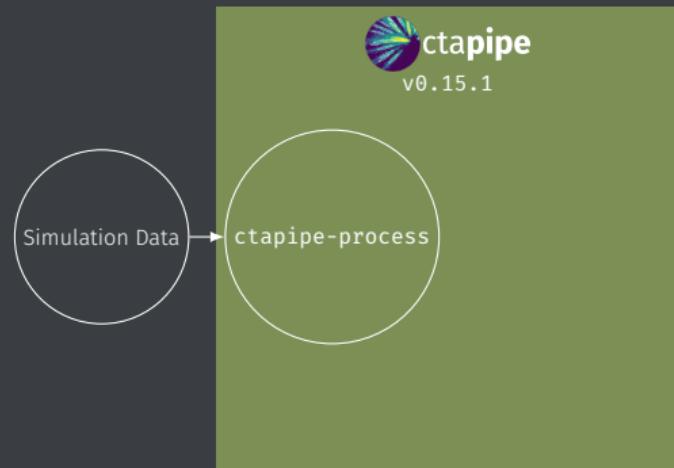
Cleaning Algorithms

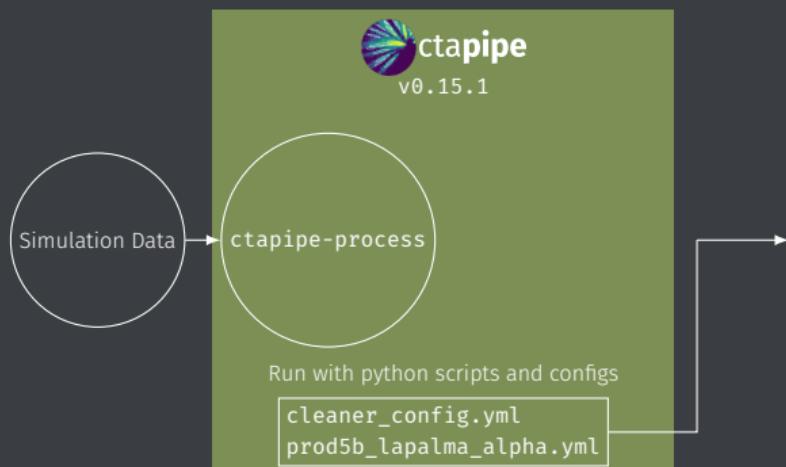


1. Finds all core pixels above the `picture threshold`
2. Removes pixels with less than N neighbors
3. Keeps all pixels that arrive within a time limit of the average arrival time (`time_limit_core`: 4.5 ns)
4. Finds all neighboring pixels above the `boundary threshold`
5. Removes all pixels with less than N neighbors arriving within a given timeframe (`time_limit_boundary`: 1.5 ns)

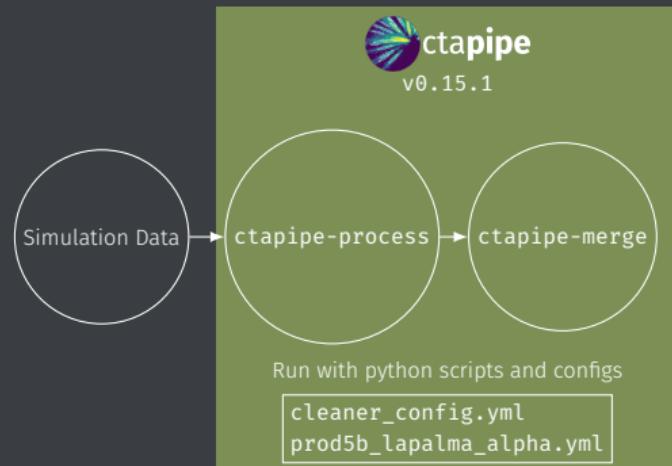
Data Processing with `ctapipe`

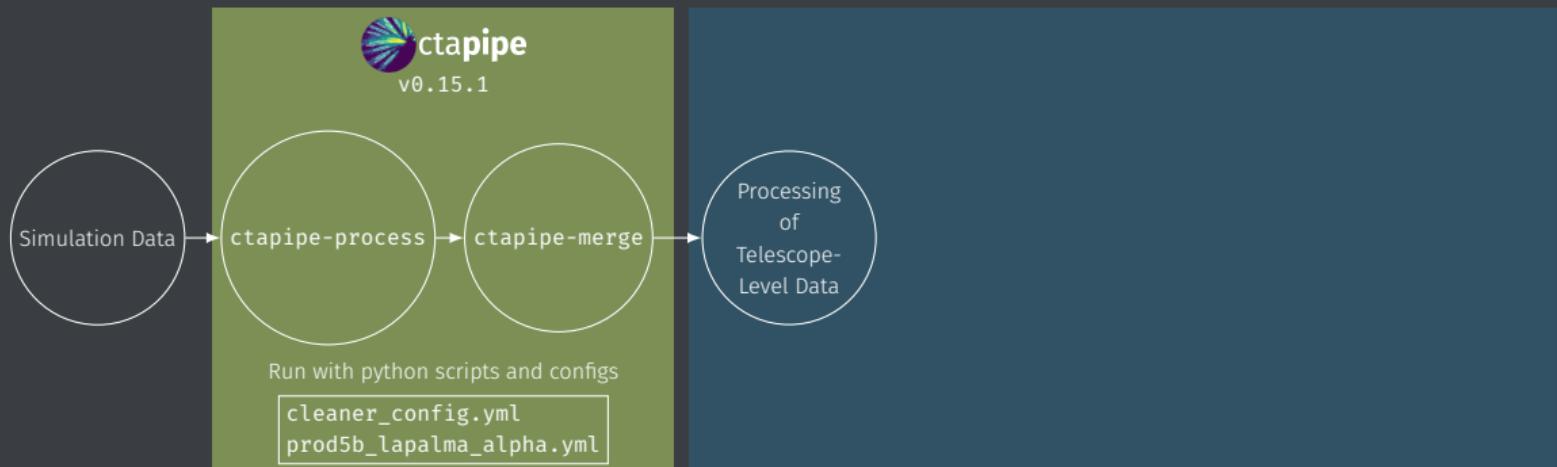


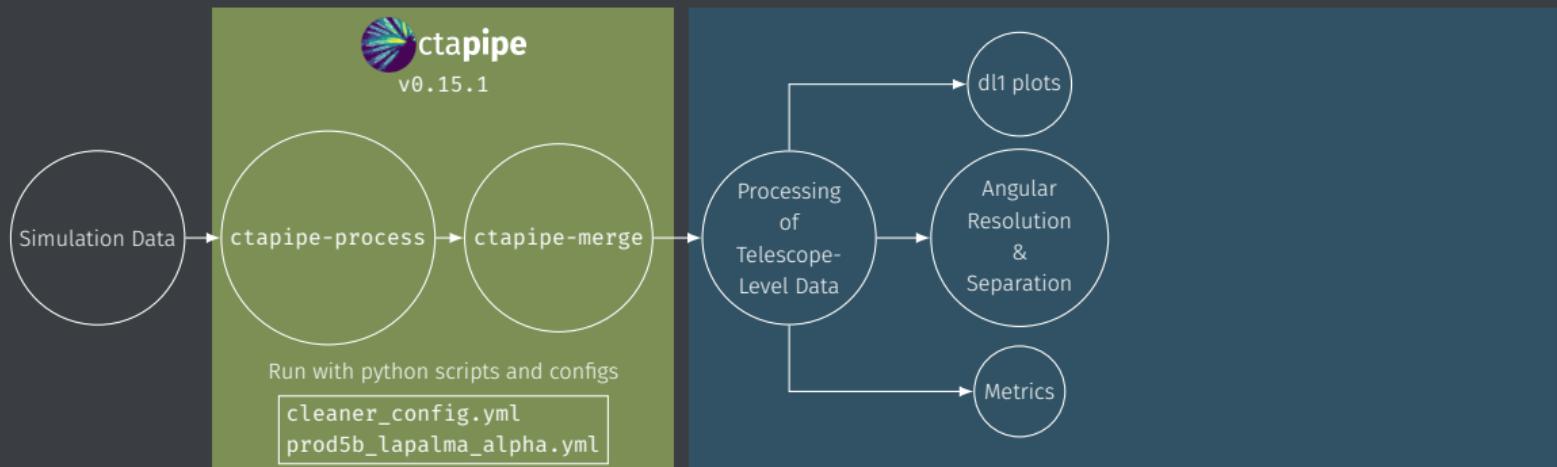


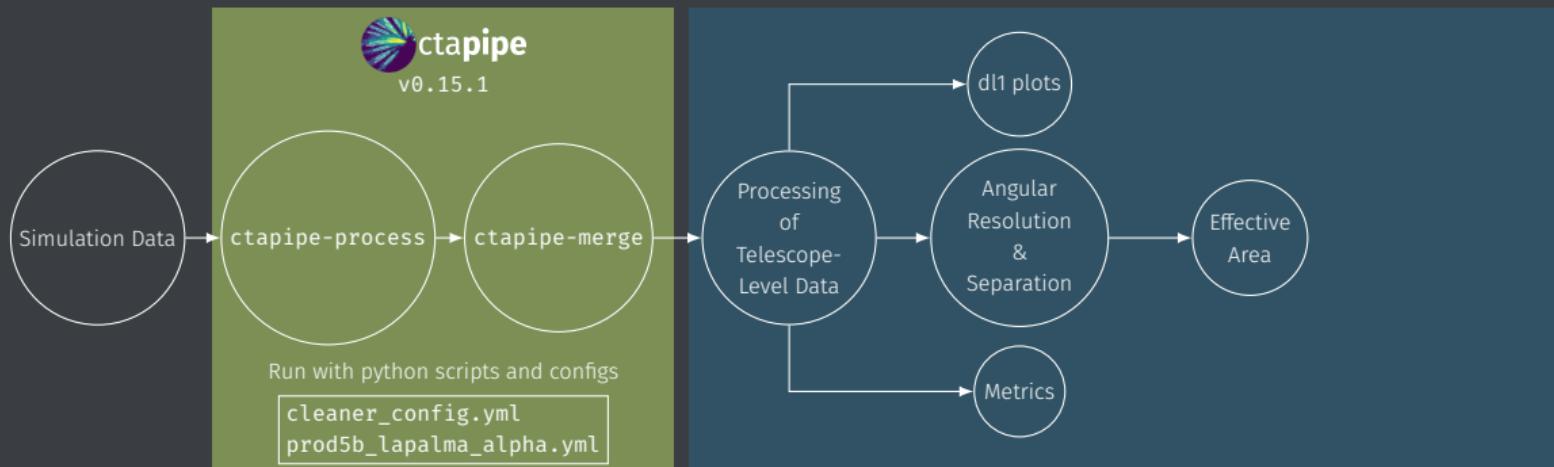


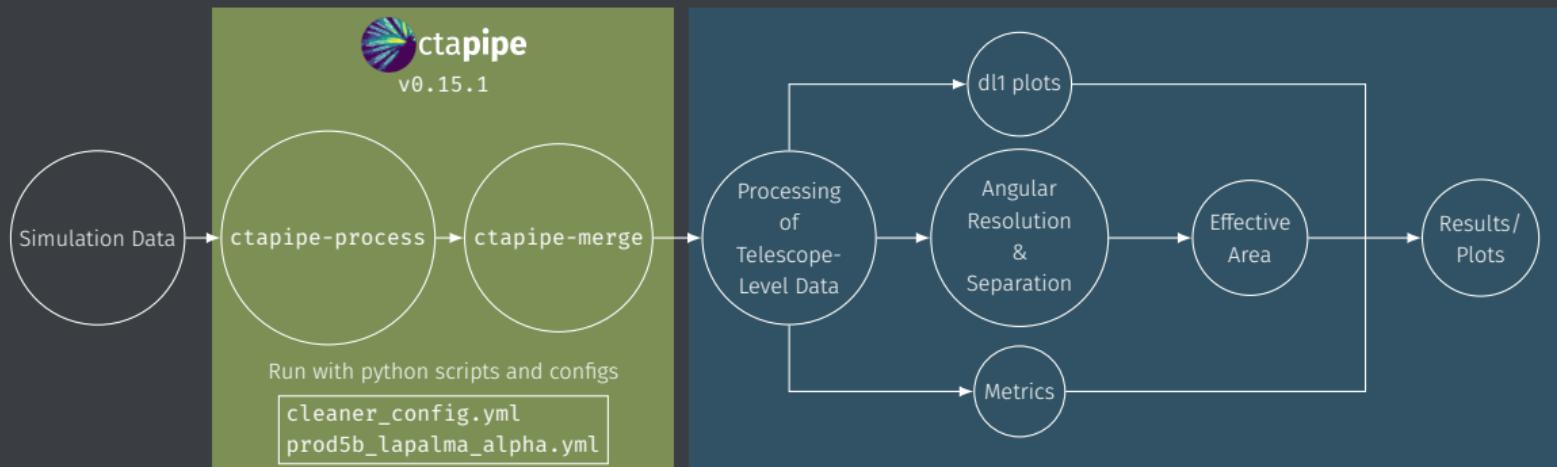
```
ImageProcessor:  
    image_cleaner_type: MARSImageCleaner  
  
MARSImageCleaner:  
    picture_threshold_pe:  
        - [type, "LST*", 8.5]  
        - [type, "MST*NectarCam", 9.0]  
    boundary_threshold_pe:  
        - [type, "LST*", 4.75]  
        - [type, "MST*NectarCam", 4.5]  
    keep_isolated_pixels: false  
    min_picture_neighbors: 2  
  
ImageQualityQuery:  
    quality_criteria:  
        - ["enough_pixels", "np.count_nonzero(image) > 2"]  
        - ["enough_charge", "image.sum() > 50"]
```





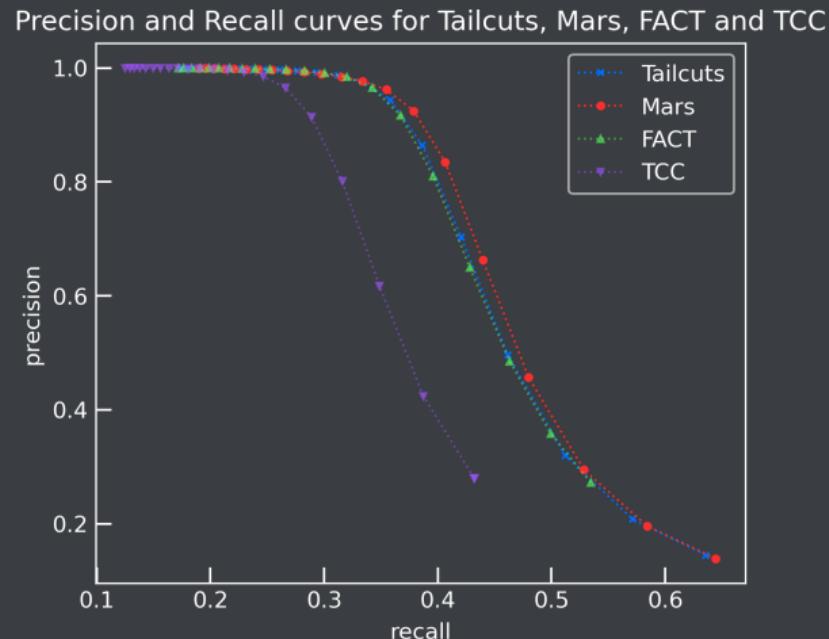
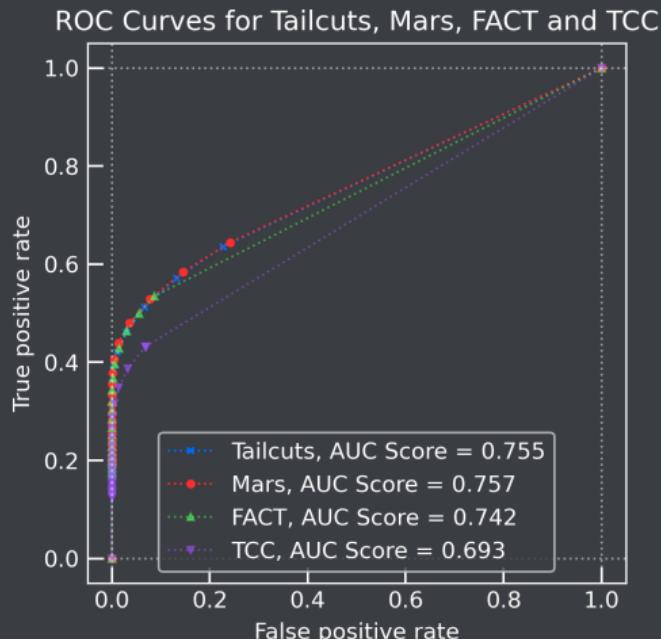




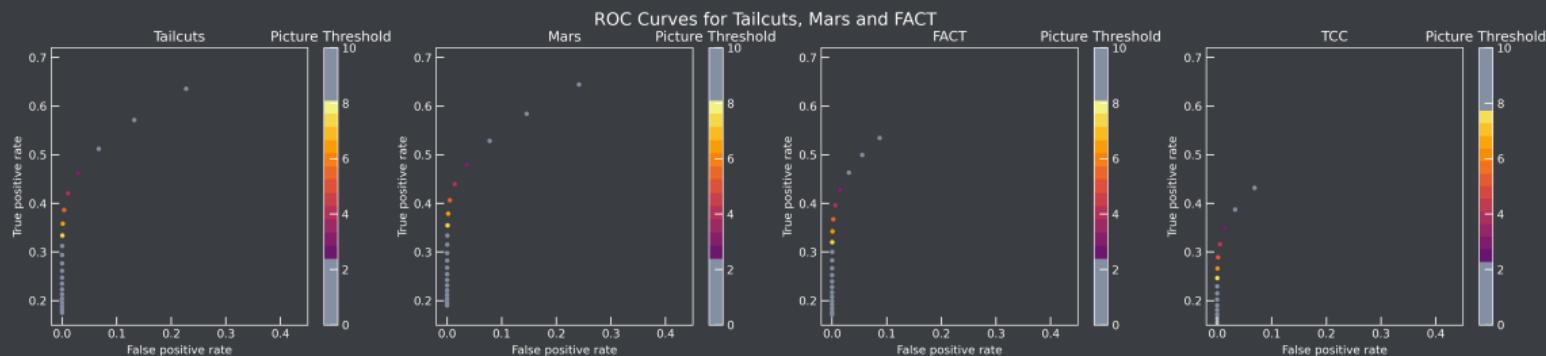


Results

ROC curves and precision and recall curves

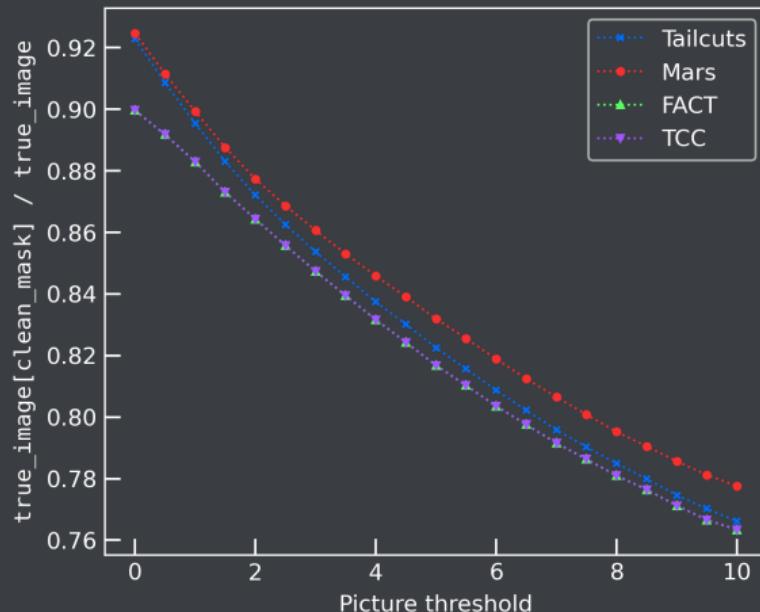
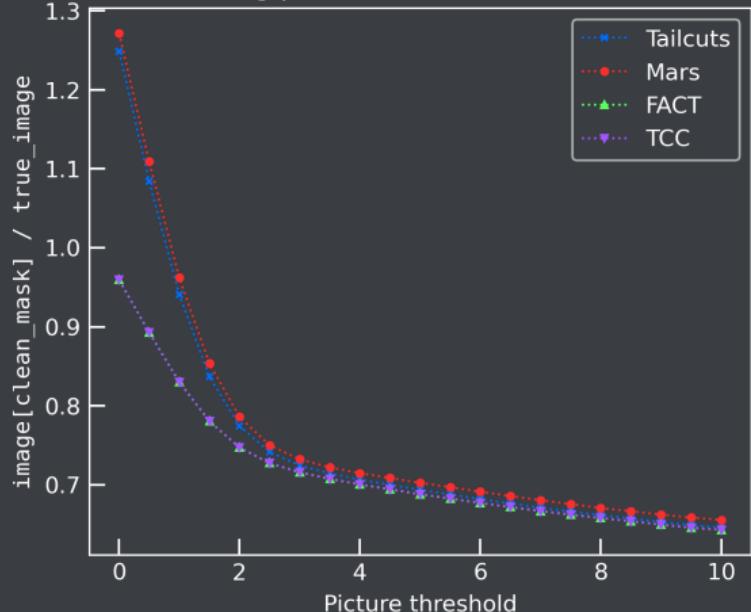


Picture Thresholds



Ratio of Surviving Pixels

Ratio of surviving pixels for Tailcuts, Mars, FACT and TCC



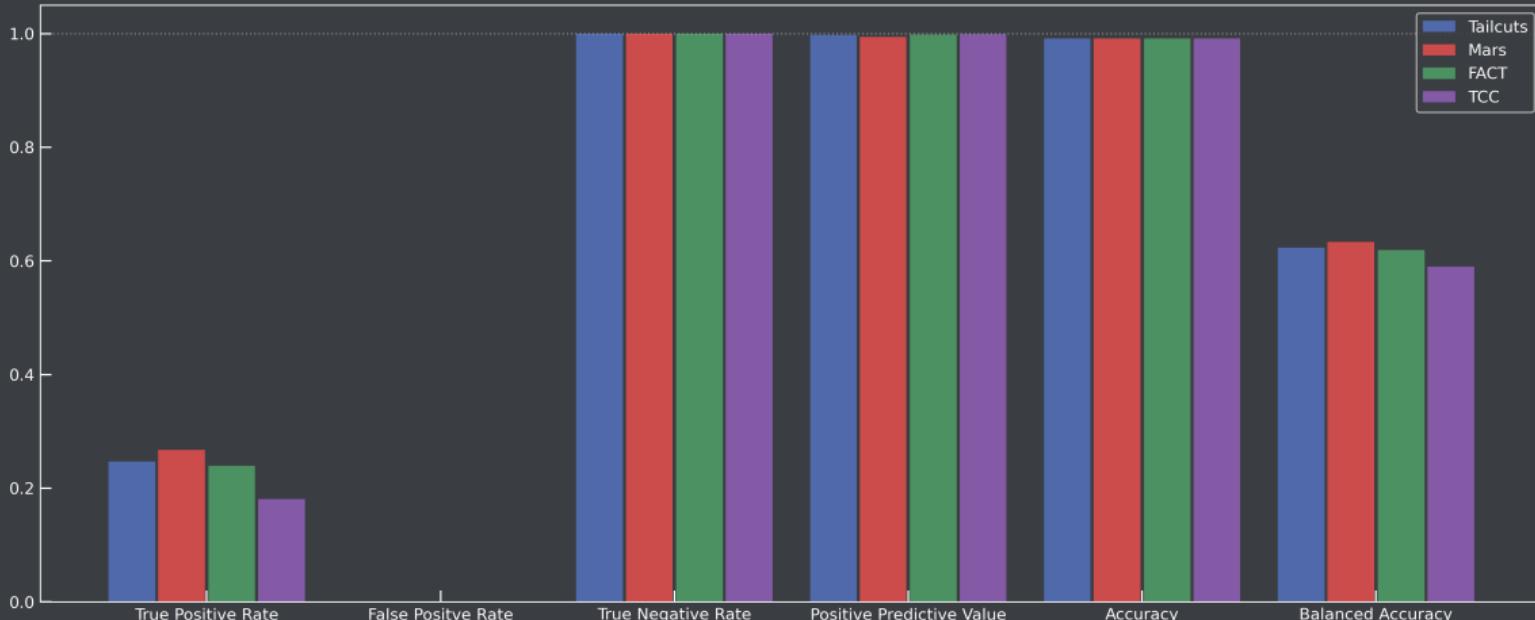
Metrics

Metrics for Tailcuts, Mars, FACT and TCC
(2.0 / 1.0)



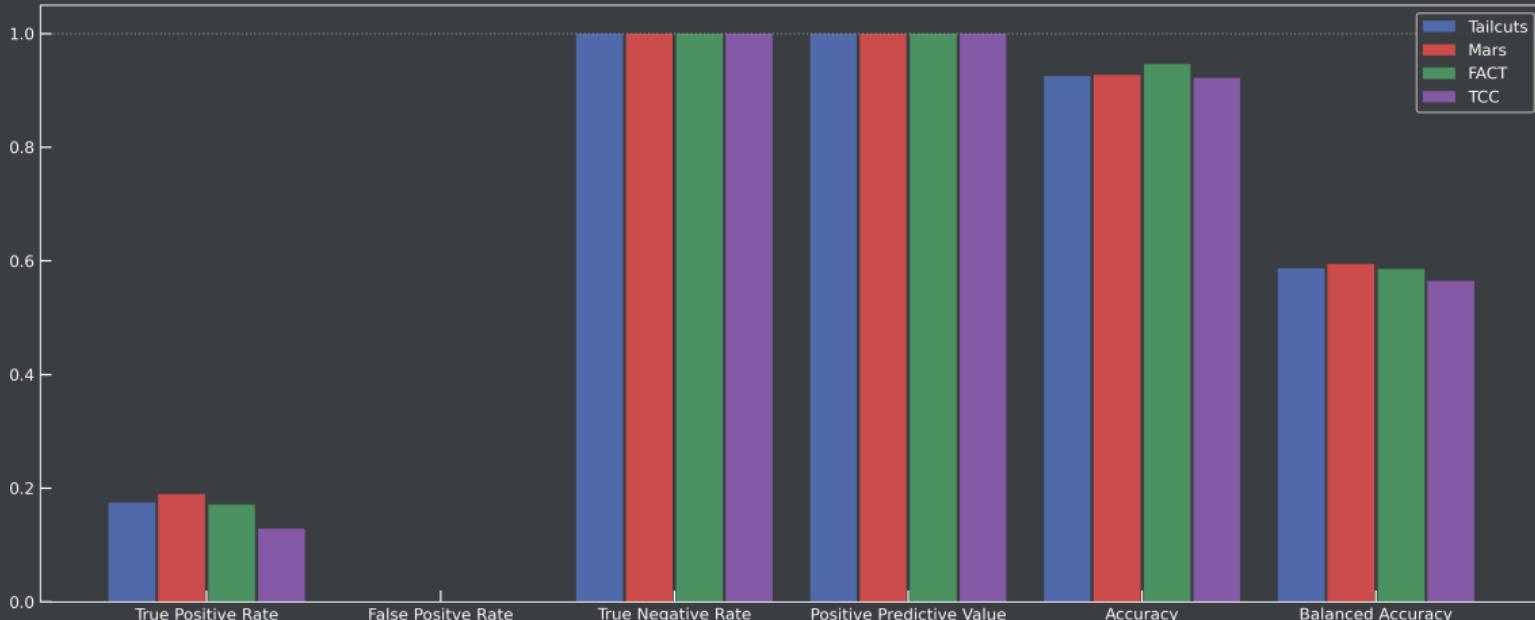
Metrics

Metrics for Tailcuts, Mars, FACT and TCC
(6.0 / 3.0)

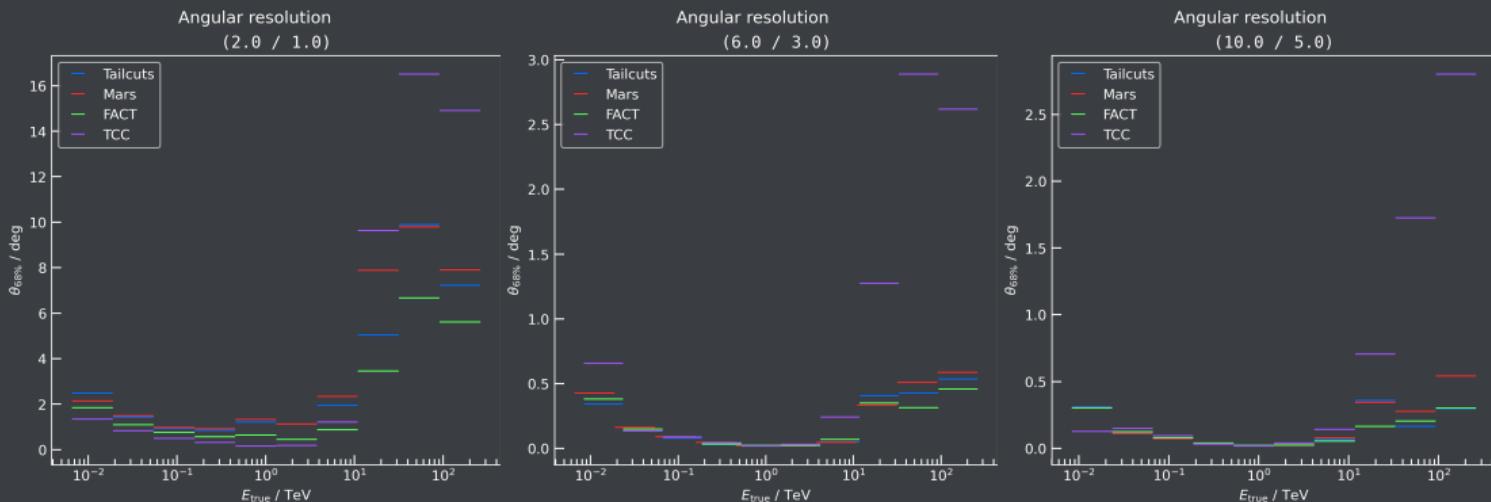


Metrics

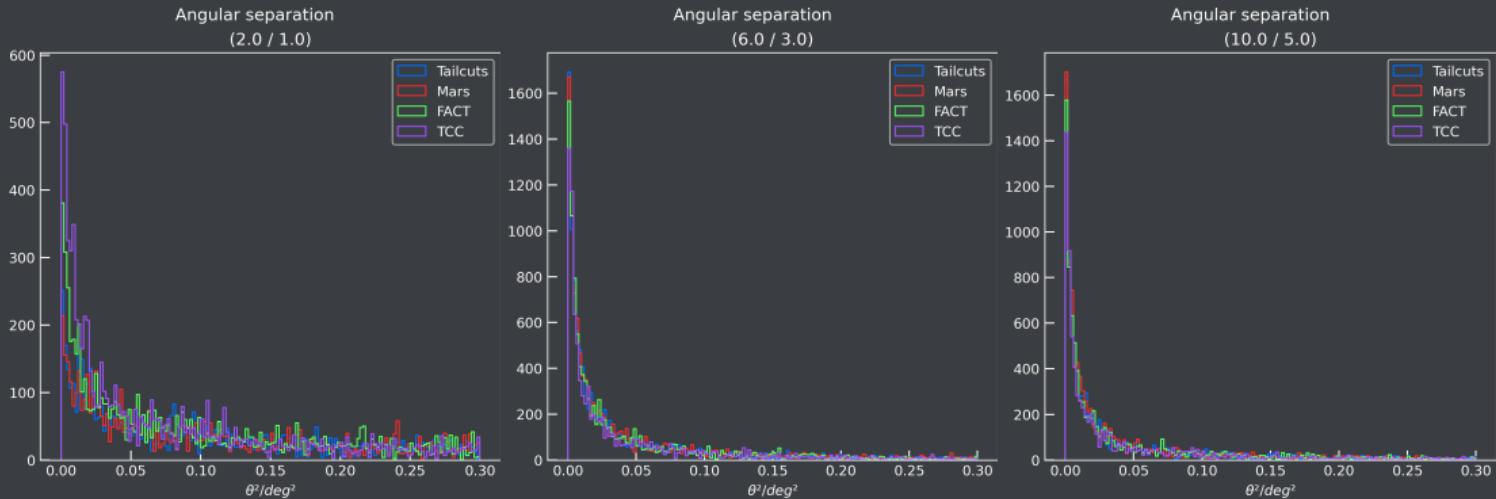
Metrics for Tailcuts, Mars, FACT and TCC
(10.0 / 5.0)



Angular Resolution



Effective Area



Outlook and Summary

Outlook

- Compare cleaners for other parameters than the picture and boundary thresholds
 - Use `sklearn.model_selection.Parametergrid` to find the best parameters for each cleaner
- Instead of letting the picture threshold vary from 0 to 10, use quantiles
- Vary the boundary thresholds as 0.25, 0.33, 0.5 and 0.75 of the picture threshold



Problems

- Run time and number of datasets increase with the number of parameters
 - For TailcutsImageCleaner and MarsImageCleaner alone, this results in 32 possible combinations of parameters:

```
params = {  
    "picture_quantiles": (0.9, 0.99, 0.995, 0.999),  
    "boundary_threshold_ratio": (0.25, 0.33, 0.5, 0.75),  
    "min_number_picture_neighbors": (1, 2)  
}
```

- Add only two parameters for FACTImageCleaner and this number increases to 64 possible combinations

```
fact_params["time_limit"] = (2, 5)
```

Summary

- So far, a picture threshold of ≈ 6.0 seems to be the best choice w.r.t. the metrics
 - Has to be tested again for combinations with other parameters
- Testing other ratios than 0.5 for the boundary thresholds seems to be a rational next step
- With sklearns ParameterGrid, it is hopefully possible to find the best parameters for each cleaner

