

**Thesis for obtaining the academic degree**  
*Bachelor of Science*

Finding optimal hyperparameters for image  
cleaning algorithms for the Cherenkov  
Telescope Array

Anno Knierim  
born in Hagen

2022

Chair of Experimental Physics V

Department of Physics

TECHNISCHE UNIVERSITÄT DORTMUND

Reviewer: Prof. Dr. Dr. Wolfgang Rhode  
Co-reviewer: Dr. Dirk Wiedner  
Submission date: 7 September 2022

This thesis is set in Libertinus (Serif, Sans and Math) and Fira Code typeset using L<sup>A</sup>T<sub>E</sub>X with LuaT<sub>E</sub>X from T<sub>E</sub>XLive 2021. The template for this thesis is an adapted version of Max Nöthes' *tudothesis*<sup>1</sup> template.  
Title graphic by Kai Brügge.

---

<sup>1</sup><https://github.com/maxnoe/tudothesis>

## Abstract

In recent years, gamma-ray astronomy has become a vastly studied field of astrophysics. Since then, many Imaging Air Cherenkov Telescope (IACT) experiments have been producing insights into sources of high-energy gamma rays. Continuing this trend, the Cherenkov Telescope Array (CTA), a next-generation array of IACTs, will provide new data targeting some of the highest energies of the gamma-ray spectrum.

In this thesis, I consider the four cleaning algorithms implemented in the low-level data processing software `ctapipe`, currently developed for CTA. Furthermore, I will determine optimized hyperparameters for each algorithm with the aim of improving their performance. A grid search achieves this objective over a range of reasonable hyperparameters. The data used in this work is diffuse gamma Monte Carlo (MC) simulation data for the Medium-Sized Telescopes (MSTs) at the northern site of CTA on the Canary island of La Palma.

The hyperparameters determined for the algorithms in `ctapipe` improve their performance over their default parameters significantly w.r.t. the angular resolution. This is especially true for the time-based `TimeConstrainedImageCleaner` (TCC) algorithm, which shows the most improvement w.r.t. the metrics, closely followed by the non-time-based `MARSImageCleaner` (MARS) algorithm. Overall, `FACTImageCleaner` (FACT) and MARS perform the best out of all cleaning algorithms in this work, with the latter being notable for its consistency in terms of the angular resolution over the whole energy range.

## Kurzfassung

In den vergangenen Jahrzehnten hat sich die Gammaastronomie zu einem viel erforschten Bereich der Astrophysik entwickelt. Seitdem haben bildgebende, atmosphärische Tscherenkow-Teleskope (IACTs) zahlreiche neue Erkenntnisse im Bereich der hochenergetischen Gammastrahlen und deren Quellen geliefert. Mit CTA, einem Observatorium bestehend aus dutzenden IACTs der nächsten Generation, werden neue Daten in einigen der höchsten Energiebereiche der Gammastrahlen erwartet.

In dieser Arbeit betrachte ich die vier Bildreinigungsalgorithmen, die in der Software `ctapipe` implementiert sind, welche sich zur Zeit für CTA in Entwicklung befindet. Zudem bestimme ich optimierte Hyperparameter für jeden Algorithmus mit dem Ziel, die Leistung der Bildreinigungsalgorithmen zu verbessern. Dies wird mittels einer Grid-Search über eine Auswahl von Hyperparametern durchgeführt. Die zu diesem Zweck verwendeten Daten bestehen aus diffusen Gammadaten, die mittels einer MC-Simulation für die MSTs des nördlichen Standorts CTAs auf der Kanarischen Insel La Palma erzeugt wurden.

Die Ergebnisse zeigen, dass die in dieser Arbeit gefundenen Hyperparameter die Leistung aller vier Algorithmen in Bezug auf die Winkelauflösung verbessern. Dies gilt insbesondere für den TCC-Algorithmus, der sich auch in Bezug auf die Metriken am meisten verbessert, gefolgt vom MARS-Algorithmus. Insgesamt arbeiten FACT und MARS von allen vier Algorithmen in dieser Arbeit am besten, wobei letzterer insbesondere bei der Winkelauflösung über den gesamten Energiebereich hinweg konstant gute Ergebnisse liefert.

# Contents

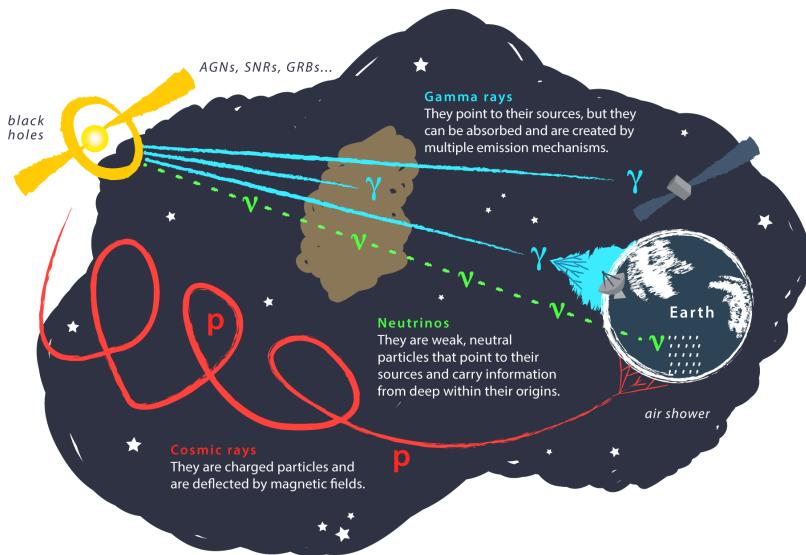
---

<b>1</b>	<b>Gamma-Ray Astronomy</b>	<b>1</b>
<b>2</b>	<b>IACTs and the Cherenkov Telescope Array</b>	<b>4</b>
2.1	Imaging Air Cherenkov Telescopes . . . . .	4
2.2	The Cherenkov Telescope Array . . . . .	6
<b>3</b>	<b>Data Processing with <code>ctapipe</code></b>	<b>8</b>
3.1	Data Simulation . . . . .	8
3.2	Data Levels in <code>ctapipe</code> . . . . .	9
3.3	This Work’s Data Processing Pipeline . . . . .	10
<b>4</b>	<b>Finding Optimal Hyperparameters for the Cleaning Algorithms</b>	<b>11</b>
4.1	Cleaning Algorithms . . . . .	11
4.2	Hyperparameters . . . . .	14
<b>5</b>	<b>Results</b>	<b>16</b>
5.1	Analysis of the Efficiency and the Angular Resolution . . . . .	16
5.2	Metrics of the Cleaning Algorithms on the Image Level . . . . .	19
5.3	Performance Compared to the Default Settings . . . . .	22
5.4	Comparison of the Cleaning Algorithms . . . . .	24
<b>6</b>	<b>Conclusions and Outlook</b>	<b>25</b>
	<b>Bibliography</b>	<b>26</b>
	<b>Glossary</b>	<b>29</b>
	<b>Appendix</b>	<b>31</b>
1	Configurations for <code>ctapipe</code> . . . . .	31
2	Hyperparameters . . . . .	35
3	Software Used . . . . .	36
4	Additional Plots and Tables . . . . .	37

# 1

## Gamma-Ray Astronomy

Astronomy, being one of the oldest sciences, is a vast field of study dating back to the earliest days of civilization, where astronomers studied the stars and the planets in order to understand the universe. It is, therefore, no surprise that astronomy spawned a great number of discoveries throughout the centuries. Whereas first observations were made using only the bare eye, we now have access to a multitude of experiments and telescopes that deepen our understanding of the universe. With the discovery of cosmic rays (CR) by Victor Hess in the early 20th century, the new field of astroparticle physics was born [26].



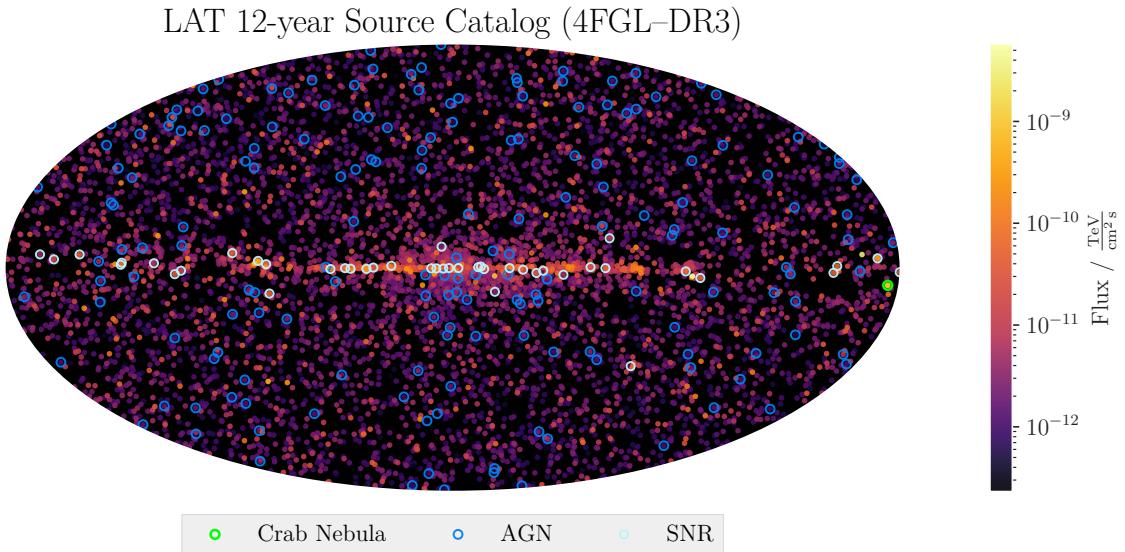
**Figure 1.1:** Different types of cosmic messengers on their way to Earth. Charged particles like protons and electrons are deflected by magnetic fields, thus making it hard to pinpoint the source. Only the origin of photons and neutrinos can be reconstructed directly since they are uncharged particles and therefore travel in straight lines. However, photons can be absorbed or created in multiple mechanisms. Since neutrinos only rarely interact with matter via the weak force, their detection is significantly harder than that of photons [6].

Only 50 years after Hess' discovery, in 1961, Explorer XI [25], the first satellite experiment to measure gamma rays from space was launched, providing measurements of gamma rays above 50 MeV, thus starting gamma-ray astronomy from space-based experiments. Just a few years earlier neutrinos were already discovered by Cowan and Reines [12], with solar neutrinos being discovered by Davis et. al. [16] in the 1960s at the Homestake experiment. The first detection of neutrinos from a supernova was made

by the Kamiokande experiment in 1987 [22]. The most recent discovery of a cosmic messenger has been that of gravitational waves in 2015 [1].

Cosmic messengers come in different types, that are either charged or uncharged, as shown in Figure 1.1. CR like electrons, protons or atomic nuclei are charged particles making it difficult to trace back their origin as they are deflected by cosmic electromagnetic fields. Uncharged particles like photons or neutrinos, however, travel in straight lines, making it easier to reconstruct their origins.

While photons can be absorbed by dust clouds on their way to Earth, they are easier to detect than neutrinos, the latter having a very small cross-section and interacting only weakly with matter. As a result, neutrino detectors and experiments, such as IceCube [5] or Super-Kamiokande [31], have to be large enough to detect neutrinos in sufficient quantities.



**Figure 1.2:** Mollweide projection of the *Fermi* Large Area Telescope (*Fermi*-LAT) 4FGL-DR3 catalog data of gamma-ray sources. The sky map shows the flux of gamma-ray sources in  $\text{TeV}/(\text{cm}^2 \text{s})$  observed over a span of 12 years from the experiment's launch in 2008. Note the concentration of high-energy (HE) gamma-ray sources around the galactic plane. Additionally, the Crab Nebula (bright green circle, right hand side)—being a standard candle gamma-ray astronomy—is marked along with supernova remnants (SNRs) (ice blue/light blue circles) and active galactic nuclei (AGNs) (blue circles) that *Fermi*-LAT observed [2, 3]. Notice, that the AGNs shown here are but a subset of all AGNs observed by the experiment—the non-blazar active galaxies. This figure was adapted from Kai Brügge's Ph.D. thesis [10].

In recent years, gamma-ray astronomy has become an important research field in astroparticle physics. The term gamma ray is generally denoted as photons with energies above 100 keV [18]. Due to this high-energy nature, gamma rays pose some of the most powerful cosmic messengers in the universe and since photons at such energies cannot be produced by thermal processes, their origin has to be described by higher order processes involving charged particles. Some of the brightest sources of gamma rays in our galaxy are supernova remnants (SNRs) such as the Crab Nebula. The Crab Nebula, in particular, poses as a so-called standard candle with its constant flux of HE gamma rays, allowing to test the performance of new experiments and telescopes. Other sources for gamma rays include gamma

---

ray bursts (GRBs), Pulsars and active galactic nuclei (AGNs), the latter being one of the most common types of extragalactic gamma-ray sources. AGNs themselves can be classified further depending on the existence of a relativistic jet, the viewing angle of the observer or how bright they are.

Gamma rays can be detected by a variety of experiments, either ground- or space-based. Space-based experiments like the *Fermi*-LAT are usually more sensitive to lower energies, but their performance is lower for higher energies, as their effective collection area is comparatively small. Ground-based experiments, however, are more sensitive to higher energies and rely on the scattering of secondary particles in extensive air showers (EASs) induced by the primary particle in Earth's atmosphere. The gamma-ray sky as observed by *Fermi*-LAT over a span of 12 years is shown in Figure 1.2.

For the past two decades, ground-based Imaging Air Cherenkov Telescope (IACT) experiments like the Major Atmospheric Gamma-Ray Imaging Cherenkov (MAGIC) telescopes, the Very Energetic Radiation Imaging Telescope Array System (VERITAS) and the High Energy Stereoscopic System (H.E.S.S.) have been monitoring these very-high-energy (VHE) gamma rays to gain an understanding of their production. This allowed determining different source classes inside and outside our galaxy, with some of the most important source classes inside our galaxy being SNRs such as the Crab Nebula, because of their brightness and the constant flux of CR in the (GeV) energy regime.

# IACTs and the Cherenkov Telescope Array

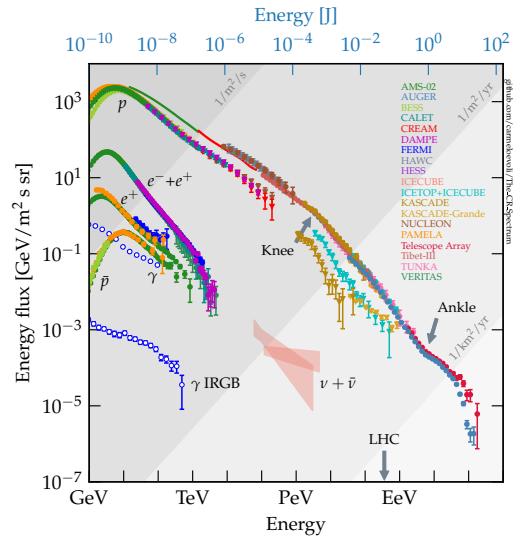
## 2

Modern gamma-ray observations are performed with either space-based experiments or with ground-based experiments. Examples include water Cherenkov tanks, such as the High Altitude Water Cherenkov (HAWC) [4] experiment, or Imaging Air Cherenkov Telescopes (IACTs), the latter being telescopes that use the Cherenkov light emitted by EASs in the atmosphere. In the following sections, I will introduce IACTs and the Cherenkov Telescope Array (CTA) and explain the mechanisms that enable the observations of gamma rays with these types of experiments.

### 2.1 Imaging Air Cherenkov Telescopes

Because of their ground-based setup, IACTs take advantage of the Earth's atmosphere to get a larger effective area than any space-based instrument. This is especially helpful for energies above 100 GeV, where the gamma-ray flux is already decreasing, whereas lower energies show higher fluxes. At such high energies, experiments with space-based instruments, such as *Fermi*-LAT, see fewer high-energy events compared to ground-based experiments, due to their comparatively small effective collection area [18, p. 256]. The cosmic ray flux in Figure 2.1 shows this well: The flux decreases rapidly with higher energies, with only one high-energy ( $>40$  GeV) photon per day and square meter reaching Earth from the Crab Nebula [27] and even fewer photons for energies in the EeV domain, where only 1 particle per square kilometer per year reaches Earth.

Earth's atmosphere is opaque to HE gamma rays, so ground-based experiments rely on cascades of sub-particles in so-called EASs. For gamma-ray astronomy, the electromagnetic component of these EASs, shown in Figure 2.2a, is of interest, as only this component can be produced by gamma rays. When a gamma ray interacts with Earth's atmosphere, it decays into an electron and a positron via pair production. These charged particles then emit more photons via bremsstrahlung. These photons, again, produce more charged particles, which in turn emit even more photons. This process continues until

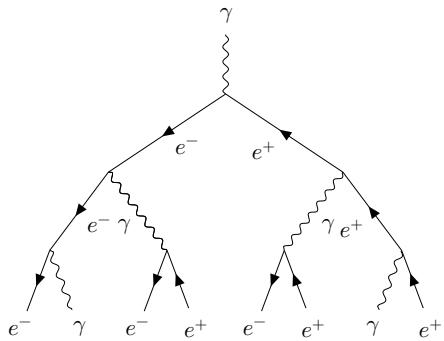


**Figure 2.1:** The cosmic ray flux as a function of energy. For very high energies the flux becomes very small with only 1 particle per square meter per year reaching Earth. For even higher energies in the domain of EeV this flux becomes 1 particle per square kilometer per year [17].

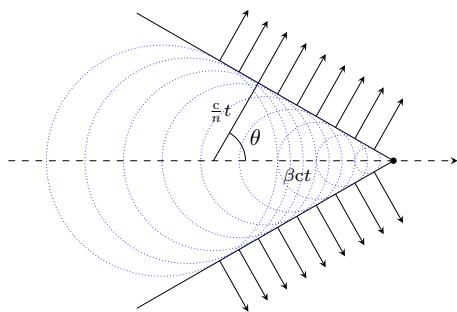
any of these processes reaches energies below a certain threshold, i. e. 1022 keV for the pair production process. For these processes to happen, the particles have to be influenced by the fields of the atoms in Earth's atmosphere. As the particles travel at speeds faster than light in the atmosphere, they emit Cherenkov light at a fixed angle  $\theta$  with respect to the refraction index  $n$  of the atmosphere and the factor  $\beta = v/c$ . The angle can be determined trigonometrically as

$$\cos \theta = \frac{1}{\beta n}, \quad (2.1)$$

as can be seen in Figure 2.2b.



**(a)** Simplified first steps of the Heitler model of the electromagnetic component of an extensive air shower. A gamma ray induces an electron and a positron via pair production that then emit more photons via bremsstrahlung.



**(b)** Cherenkov radiation (outward-pointing arrows) from a charged particle traveling at uniform velocity, faster than the speed of light in the medium. The dashed line shows the path of the particle over time.

**Figure 2.2:** The electromagnetic component of an extensive air shower (see (a)) is induced by gamma rays that then decay into electrons and positrons via pair production. These charged particles will emit more gammas via bremsstrahlung, which again will produce new electrons and positrons. This will continue until the energy of the gammas is below the threshold of 1022 keV for pair production. Another process is the production of Cherenkov light from the electrons and positrons, which is shown in (b). The Cherenkov light is the result of the charged particles traveling faster than the speed of light in the atmosphere, thus emitting photons in a fixed angle  $\theta$  w. r. t. the refraction index  $n$  of the medium and the factor  $\beta$ .

Cherenkov light emitted by EASs can then be collected by an IACT's mirrors and detected by its camera within a timeframe on the order of nanoseconds. The resulting image will show the shape of the air shower and can be used to determine the shower's primary particles' properties and reconstruct its origin. As the hadronic component of EASs produce electromagnetic subshowers, IACTs have a dominant hadronic background, which has to be separated from the gamma ray-induced showers, for example with machine learning algorithms. Before this separation, however, signal pixels in the camera image have to be separated from the electronic noise coming from the camera as well as from the Night Sky Background (NSB) with the help of cleaning algorithms. This will be the focus of this work. These cleaning algorithms pose a vital step in the analysis of an event, as only a fraction of all pixels in the camera frame are part of the signal.

## 2.2 The Cherenkov Telescope Array

CTA is a new generation of IACTs that will consist of two sites, one of which will be built at the Observatorio del Roque de los Muchachos (ORM) on the Canarian island of La Palma while the other one will be built in the southern hemisphere at the European Southern Observatorys (ESO) Paranal Observatory in the Atacama desert of northern Chile.

CTA consists of three types of telescopes, shown in Figure 2.3, each being sensitive to a different energy range [13]:

**Large-Sized Telescope (LST):** The largest telescopes in CTA have a primary deflector diameter of 23 m with a field of view of 4.3 deg and provide full system sensitivity in the energy regime from 20 GeV to 150 GeV.

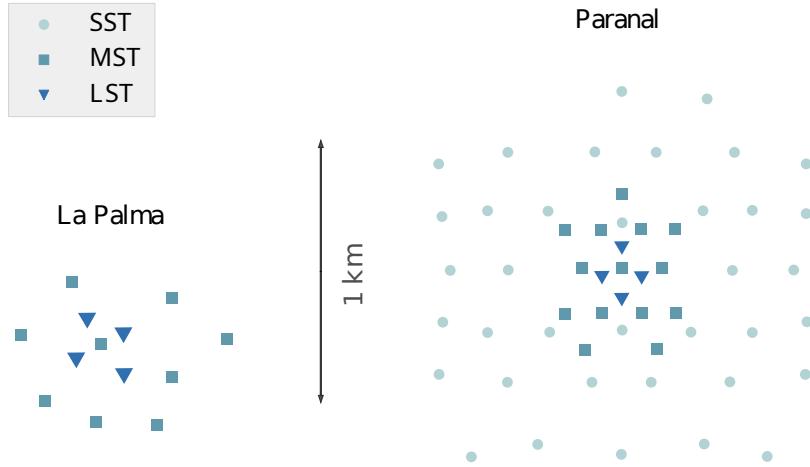
**Medium-Sized Telescope (MST):** The MSTs have a primary deflector diameter of 11.5 m with a field of view of 7.7 deg for their NectarCam and provide full system sensitivity in the energy regime from 150 GeV to 5 TeV.

**Small-Sized Telescope (SST):** Being the smallest telescope type in CTA, the SSTs have a primary deflector diameter of 4.3 m, a secondary deflector diameter of 1.8 m with a field of view of 10.5 deg and provide full system sensitivity in the energy regime from 5 TeV to 300 TeV.



**Figure 2.3:** A rendered image of the 3 telescope prototypes in CTA. From left to right: The SST, two types of MST (SCT and MST) and the LST. This work will only feature the MST, as the simulation data used contains solely MST data for the northern array (see chapter 5). Image credit: Gabriel Pérez Diaz, IAC [35].

Since the southern hemisphere has a better view of the galactic center than the northern hemisphere, only CTA south will feature the SSTs along the other two telescope types. CTA north will only feature MSTs and LSTs. In its full configuration, CTA would feature 70 SSTs, 25 MSTs and 4 LSTs at CTA south, and 15 MSTs and 4 LSTs at CTA north. However, a reduced layout—called Alpha Configuration—is the currently planned layout for both CTA north and south. This work will focus on the northern site, consisting of 9 MSTs and 4 LSTs [14]. Figure 2.4 shows the planned Alpha Configuration layout of both sites of CTA where CTA south is displayed with its 37 SSTs, 14 MSTs and the 4 LSTs in the center [15].



**Figure 2.4:** Alpha Configuration layout of the CTA north and CTA south site. CTA south would feature 37 SSTs, 14 MSTs and 4 LSTs [15] in its Alpha Configuration, while CTA north would feature 9 MSTs and 4 LSTs [14]. This figure was adapted from Kai Brügge in his Ph.D. thesis [10].

The MST's and LST's cameras are based on photo multiplier tube (PMT) photodetectors, while the SST's camera is based on a silicon photomultiplier (SiPM) photodetector. Both detector types provide fast response times on the order of nanoseconds. PMTs can detect small amounts of electromagnetic radiation due to their amplification process. However, this amplification process also amplifies the noise of the detector and requires higher voltages than SiPMs. The latter, on the other hand, has to be cooled, as SiPMs are prone to having a higher dark current with increasing temperatures.

With CTA probing the VHE gamma-ray sky and a resolution outperforming predecessors, like H. E. S. S. or MAGIC, it will be able to observe a broad variety of sources, allowing insights into interaction processes of VHE gamma rays as well as their origin.

Together with gravitational wave and neutrino observatories as well as with other gamma observatories, CTA will probe the universe in its high-energy (HE) domain, allowing to study a broad field of key questions in astrophysics. As such, CTA's scientific goals can be divided into three categories [11]:

**Origin and Role of relativistic Cosmic Particles:** CTA will try to find the sources of HE cosmic particle acceleration in the universe. This includes the search for the mechanisms that accelerate cosmic particles. Also, CTA will try to answer what role these particles play in feedback on star formation and galaxy evolution.

**Extreme Environments:** CTA aims to deepen our understanding of the processes that are at work close to neutron stars and black holes, the characteristics of relativistic jets and pulsar wind nebula (PWN). Also, it will probe the intensity of radiation fields and magnetic fields and how these would evolve over cosmic time scales.

**Frontiers in Physics:** CTA will try to answer questions about the nature of dark matter and its distribution in the universe. As such, the existence of axion-like particles will be probed. Furthermore, CTA will search for quantum gravitational effects on photon propagation.

# Data Processing with `ctapipe`

# 3

The analysis in this work was done with the open-source low-level data processing software for CTA, `ctapipe` [24], more specifically, the code from the master branch from July 6 (f26107fe). The goal of `ctapipe` is to provide a complete analysis framework ranging from data calibration and image extraction to the reconstruction of events and the analysis of their properties. In this chapter, I will first describe how the data was simulated in section 3.1 and then, in section 3.2, explain the different data levels and `ctapipe`'s various analysis steps. In section 3.3 I will explain the full pipeline created for this work.

## 3.1 Data Simulation

In its current in-development state, `ctapipe` is used and tested with simulated data, as a comparison to true data makes testing easier, although, in theory, real data can be tested, too. This data is created with the help of Monte Carlo (MC) simulations. The software used for these simulations is called Cosmic Ray Simulations for Kascade (CORSIKA) [21] and allows for detailed simulation of EASs initiated by HEs cosmic rays. The showers are observed by a virtual IACT array, the `sim_telarray` [9], that allows simulating the optics of various mirror and camera geometries via ray-tracing of incoming photons. This also includes simulations of imperfections or misalignments of the mirrors of each telescope. The resulting so-called `simtel` data is used in `ctapipe` and can be processed as described in section 3.2 and Figure 3.1. Some of the data properties are listed in Table 3.1.

This data is the basis for the analysis in this work. A total of around 27 k telescope events (~12 k array events) throughout 20 `simtel` runs of diffuse gamma data were used for the initial analysis, i. e. the testing of different parameters as described in section 4.2. Telescope events are the events that have been observed on a telescope level, while array events are the events where at least two telescopes observed the same EAS. After processing the individual runs and merging them into one large file, this file is processed for the aforementioned parameters.

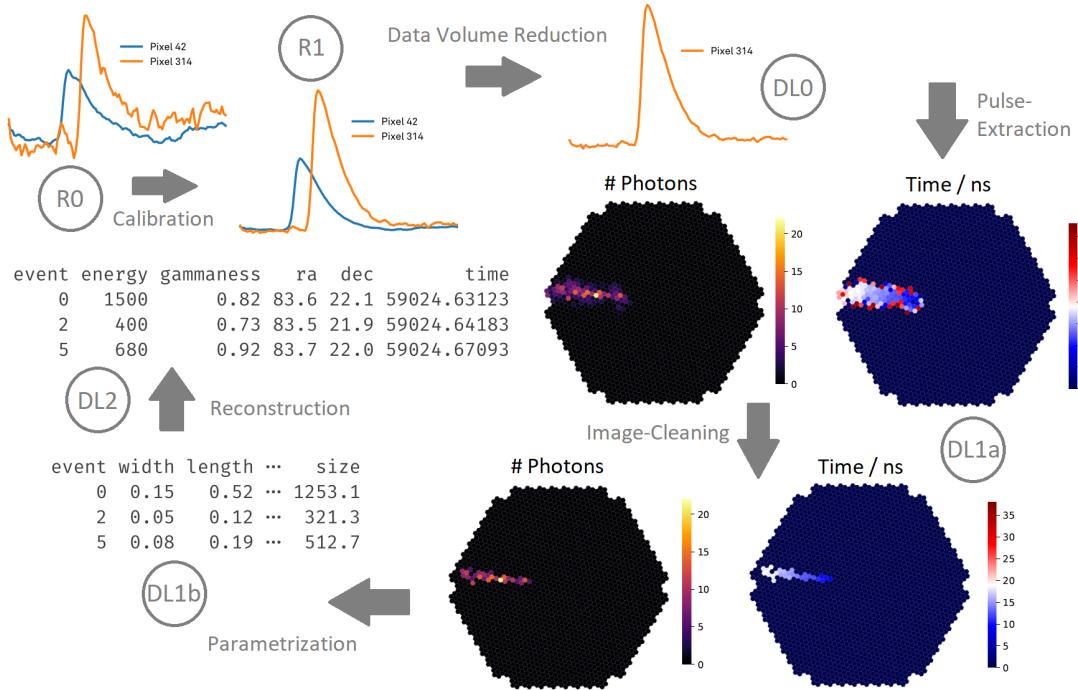
Once promising parameters are found, a larger dataset, containing around 1.3 M telescope events (~620 k array events) throughout 987 runs, is processed employing these selected settings, allowing for more statistics and a better comparison of the cleaning algorithms.

**Table 3.1:** Simtel data properties of the CORSIKA simulation used for the datasets in this work's analysis.

Diffuse Gamma data	
Energy range / TeV	0.003–330
Zenith angle / deg	20
View cone angle / deg	10
Number of showers	50 000
Spectral index	−2.0
Maximum scatter range / m	1900

## 3.2 Data Levels in ctaapipe

There are several data levels in ctaapipe, spanning from the raw data **R0** to the reconstructed events **DL2**, with the raw data levels being denoted by an **R** and the stored data levels by a **D**. Figure 3.1 shows a simplified overview of the data levels and the analysis steps. The raw data level **R0** is the data that comes directly from the photodetectors. The time-resolved signal of the data is calibrated from **R0** to **R1**. The data volume can be reduced from **R1** to **DL0** by selecting and saving waveforms, thus discarding pixels that likely do not contain any signal.



**Figure 3.1:** Data levels in ctaapipe. Raw data levels are denoted by an **R** and calibrated data levels by a **D**. The raw data first gets calibrated (**R0** → **R1**) and then reduced in volume by selecting waveforms (**R1** → **DL0**). From there the images are extracted (**DL0** → **DL1a**) and cleaned with a cleaning algorithm. This allows for parametrization of the events (**DL1a** → **DL1b**). The parametrized events can then be reconstructed (**DL1b** → **DL2**) [27, 19].

The **DL0** data level is the first level being stored and also the first level to be processed from the simulation datasets used in this work. From **DL0** to **DL1a**, the photon images are extracted by integrating the waveform peaks of each pixel, whilst the time images (“peak times”, compare Figure 3.1) are extracted by finding the rising edge of the pulse. The resulting images are cleaned by algorithms, to distinguish signal from NSB, which, in turn, allows for a parametrization of the events from **DL1a** to **DL1b**. The cleaning is a necessary step, especially for low-energy events, where the signal count is low compared to the NSB. The cleaning algorithms of ctaapipe are introduced in section 4.1 and can be roughly categorized into time-based and non-time-based algorithms.

While the latter are selecting pixels based on charge thresholds, which subsequently might lead to the need for a higher threshold, the former also takes the arrival times of pixels in the camera frame into account, allowing for a lower charge threshold.

The advantage of a lower charge threshold is, that it allows the selection of a larger portion of the signal at lower energies. Once the events are cleaned and parametrized, they can then be reconstructed (**DL1b** → **DL2**) and stored on the **DL2** data level, which consists of an event list with direction, energy and the reconstructed gammaness—a value that predicates how likely an event was produced by a gamma.

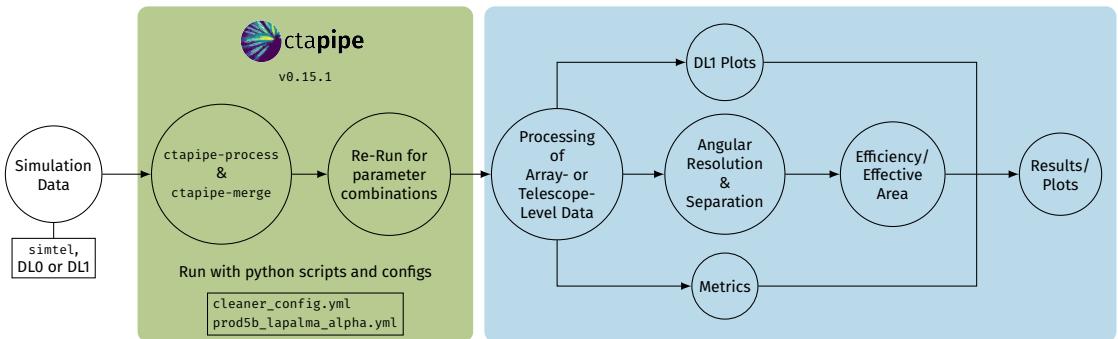
### 3.3 This Work’s Data Processing Pipeline

In this work, the data processing pipeline is as follows: First, several simulation data runs are selected and processed with *ctapipe*. Then, the datasets are merged and serve as a basis for a re-run of the combined dataset with various parameter combinations described in chapter 4.

The settings for *ctapipe* are saved in two configuration files: First, a file that sets up the cleaning process and what data to write to the output file. For the preprocessing step of this work’s pipeline, the cleaning settings are not relevant, as this step only serves to create a merged dataset. The second file contains a list of all the allowed telescope IDs. This allows a selection of the telescopes that are used in the analysis. This is especially helpful if one wants to only analyze MST-related data, like in this work.

For the re-run of the combined dataset, the cleaning settings are used, however, since this step is the heart of this work’s search for the optimal hyperparameters. The configuration files used are shown in appendix 1.

Each resulting dataset can then be processed on an array or telescope data level, resulting in **DL1a** image plots, values for the angular resolution and the efficiency as well as metrics for the performance. An in-detail description of how this helps to compare the performance of the different cleaning algorithms can be found in section 4.2. A schematic overview of the data processing pipeline of this work is shown in Figure 3.2.



**Figure 3.2:** Schematic overview of the data pipeline used for this work. Single runs of the simulation data are processed with *ctapipe-process* and then merged via the tool *ctapipe-merge* (green shaded area). The merged data is then processed on the array or telescope data level resulting in scores for metrics as well as **DL1** images and plots for the angular resolution and the effective area (blue shaded area).

# Finding Optimal Hyperparameters for the Cleaning Algorithms

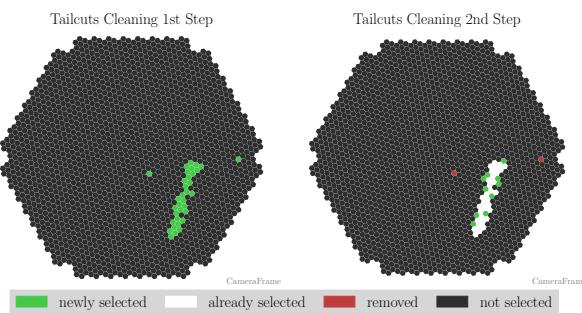
4

This work aims to find optimal hyperparameters for the cleaning algorithms used in `ctapipe`. Optimizing the cleaning step of the **DL1** data level in section 3.2 can most certainly lead to a better reconstruction of the events in a dataset. Therefore, a tweaking of the available parameters of each cleaner is necessary. In this chapter, I will first introduce the cleaning algorithms and their parameters in section 4.1 and then describe the procedure to find optimal hyperparameters in section 4.2.

## 4.1 Cleaning Algorithms

### TailcutsImageCleaner

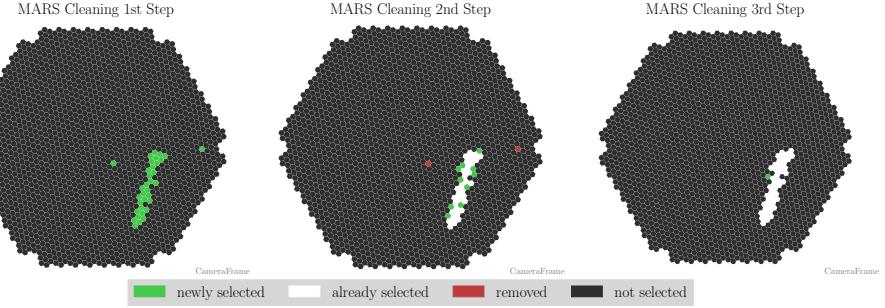
The used version of `ctapipe` features four cleaning algorithms, two of which are time-based. The `TailcutsImageCleaner` (Tailcuts) algorithm is the most basic algorithm of the four and serves as a good starting point for the development of new cleaning algorithms. Its first step is to select all pixels that are above a certain threshold, the core threshold  $Q_c$  (also called picture threshold in `ctapipe`). These pixels are the core part of the signal and are the brightest. The Tailcuts algorithm then selects all pixels that are above the boundary threshold  $Q_b$  and are neighboring the core pixels. A visualization of the algorithm is shown in Figure 4.1 for the default values of the algorithm.



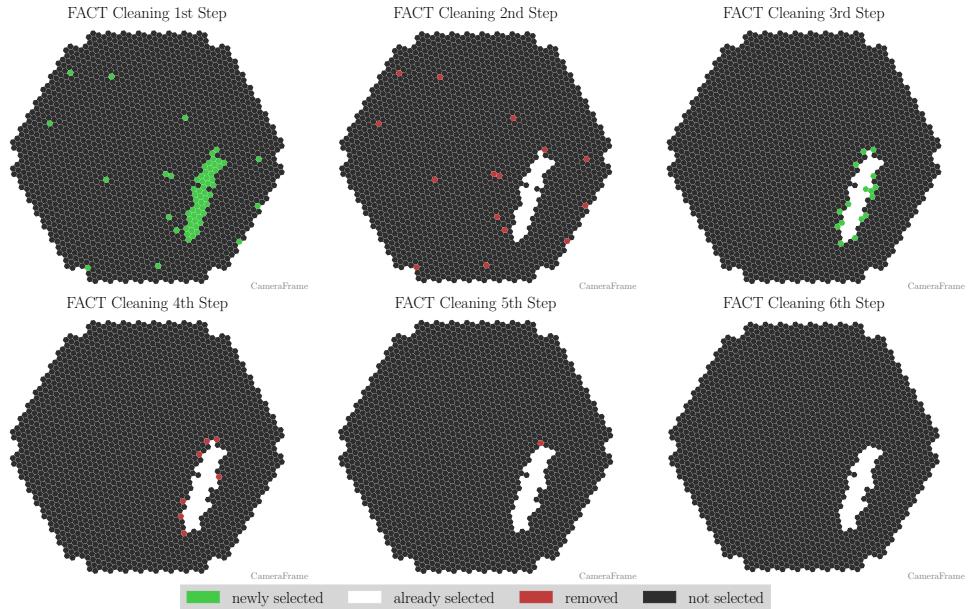
**Figure 4.1:** Visualization of the Tailcuts algorithm for a MST NectarCam image. First, all pixels above the core threshold are selected. Then, all pixels neighboring the core pixels that are above the boundary threshold are selected.

### MARSImageCleaner

The `MARSImageCleaner` (MARS) algorithm [36] is very much based on the Tailcuts algorithm and features an additional step, in that it also selects all neighbors of a neighbor of a core pixel, if they are above the boundary threshold. The three steps for the MARS algorithm are shown in Figure 4.2 for its default values.



**Figure 4.2:** Visualization of the MARS algorithm for a MST NectarCam image. The first two steps are identical to the Tailcuts algorithm. The third step selects all neighbors of a neighbor of a core pixel if they are above the boundary threshold  $Q_b$ .



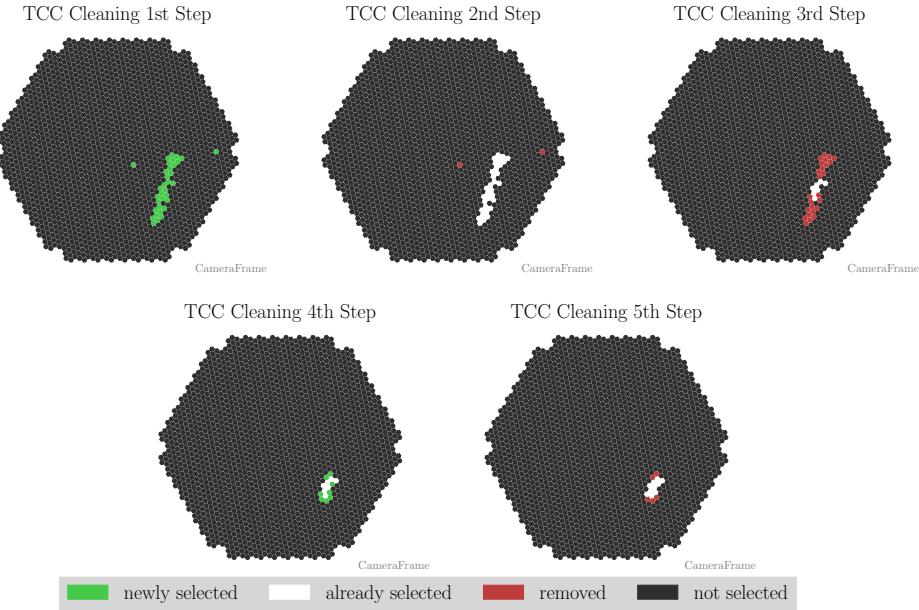
**Figure 4.3:** Visualization of the FACTImageCleaner (FACT) algorithm for a MST NectarCam image. The first step selects all pixels above the core threshold  $Q_c$ . The second step removes all pixels that have less than  $N$  neighbors. The third step selects all pixels neighboring the remaining pixels that are above the boundary threshold  $Q_b$ . The fourth step removes all pixels that have less than  $N$  neighbors, that have arrived within a given timeframe. The fifth and sixth steps are analogous to steps two and four respectively. Notice, that—per default in ctapipe—the values for  $Q_c$  and  $Q_b$  are lower than for the other cleaners (see Table 4.1), and hence together with the time limit more individual pixels than in the other cleaning algorithms are selected in the first step.

### FactImageCleaner

The FACT algorithm [33, 32] is a time-based cleaning algorithm that first selects all pixels that are above  $Q_c$ . Then, all pixels that have less than  $N$  neighbors are removed. For the third step, all pixels neighboring the remaining pixels that are above  $Q_b$  are selected. After that, all pixels that have less than  $N$  neighbors and that have arrived within a given timeframe are removed. Again, all pixels with less than  $N$  neighbors are removed. The last step once again removes pixels with less than  $N$  neighbors, arriving within the given time limit. The visualization of the algorithm is shown in Figure 4.3 for the default values of the algorithm.

### TimeConstrainedImageCleaner

The TimeConstrainedImageCleaner (TCC) algorithm is another time-based algorithm, coming from the MAGIC collaboration [30]. It first selects all pixels that are above  $Q_c$ . Then, all pixels that have less than  $N$  neighbors are removed. A minimum of 1 neighboring pixel is needed in the default settings. After that, all core pixels whose arrival times are within a given timeframe of the average arrival time. As a fourth step, the TCC algorithm finds all pixels above  $Q_b$ . Then, all pixels with less than  $N$  neighbors arriving within a given timeframe are removed. The visualization of the algorithm is shown in Figure 4.4 for the default values of the algorithm. For a better comparison, the resulting cleaned images of each cleaner are shown in Figure 1 in appendix 4.



**Figure 4.4:** Visualization of the TCC algorithm for a MST NectarCam image. The first step selects all pixels above the core threshold  $Q_c$ . The second step removes all pixels that have less than  $N$  neighbors. The third step selects all pixels whose arrival times are within a given timeframe of the core time limit  $t_c$ . The fourth step selects all pixels above the boundary threshold  $Q_b$ . The fifth step removes all pixels with less than  $N$  neighbors, that have arrived within a given timeframe of the boundary time limit  $t_b$ . Notice, that with the default values TCC tends to select fewer pixels than the other algorithms.

## 4.2 Hyperparameters

The hyperparameters of each cleaning algorithm are set in a specific configuration file. There, the user can set and change the parameters, shown in Table 4.1. Tailcuts and MARS can be set-up with only three parameters: the `picture_threshold`, the `boundary_threshold` and `min_number_picture_neighbors`. The time-based algorithms have additional parameters: First of all a `time_limit` for FACT and then a `time_limit_core` as well as a `time_limit_boundary` for TCC.

**Table 4.1:** The four cleaning algorithms and their hyperparameters. Being the most basic algorithms, Tailcuts and MARS have only three parameters, while the FACT and TCC algorithms have one and two additional time-based parameters, respectively. This table shows the default values and, if the parameters have them, also their units, as they are implemented in the ctapipe source code.

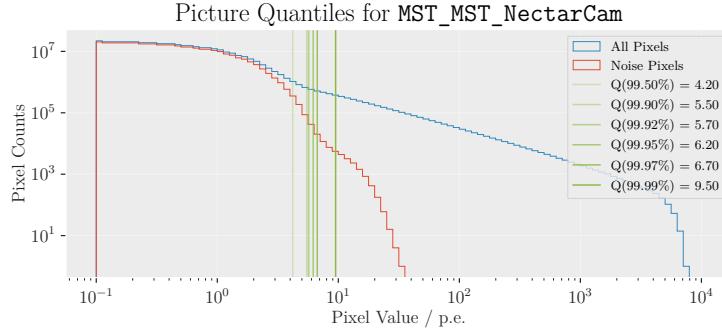
Cleaning Algorithm	Hyperparameter	Default Values
Tailcuts	<code>picture_threshold</code>	7 p.e.
	<code>boundary_threshold</code>	5 p.e.
	<code>min_number_picture_neighbors</code>	0
MARS	<code>picture_threshold</code>	7 p.e.
	<code>boundary_threshold</code>	5 p.e.
	<code>min_number_picture_neighbors</code>	0
FACT	<code>picture_threshold</code>	4 p.e.
	<code>boundary_threshold</code>	2 p.e.
	<code>time_limit</code>	5 ns
	<code>min_number_picture_neighbors</code>	2
TCC	<code>picture_threshold</code>	7 p.e.
	<code>boundary_threshold</code>	5 p.e.
	<code>time_limit_core</code>	4.5 ns
	<code>time_limit_boundary</code>	1.5 ns
	<code>min_number_picture_neighbors</code>	1

The full configuration files for the default settings of each cleaning algorithm are listed in appendix 1. There, the configuration files for the allowed telescopes as described in section 3.3 are shown, too.

To find the optimal hyperparameters, I utilized `scikit-learn` [29] for a grid search. This allows one to manually set subsets of hyperparameters and search for the best combination of these. Table 1 in appendix 2 shows the subsets of hyperparameters used in this work.

I used noise quantiles, i. e. the percentage of all *noise* pixels that are not part of the true signal, to determine the core threshold  $Q_c$ . This would mean, that e. g. 99.99 % of the noise pixels would fall under a threshold of 9.50 p.e. for the data used in this work. The quantiles are calculated directly from the images by discriminating the noise pixels from the image with the help of the true image. This approach has the advantage, that the resulting thresholds are more precisely optimized<sup>1</sup>. Since the boundary thresholds are calculated as a quotient of  $Q_c$ , this benefits here as well. The quantiles used in this work are shown in Figure 4.5.

<sup>1</sup>As opposed to setting the thresholds by hand, e. g. 4 to 10 in 0.5 increments.



**Figure 4.5:** Picture noise quantiles for a MST NectarCam dataset. The quantiles are calculated as the percentage of noise pixels in the image. The resulting threshold can then be used as a candidate for the core threshold  $Q_c$  in a grid search.

The grid search results in 150 parameter combinations for Tailcuts and MARS each, as well as another 1050 combinations for FACT and further 3000 combinations for TCC. Since it is impossible to find the optimal settings just by looking at the cleaned images of the datasets, a combined metric is necessary to evaluate each cleaner's performance for the parameter combinations.

In this work, I chose to first look at the efficiency

$$\text{Eff} = \frac{n_{\text{reco}}}{n_{\text{total}}}, \quad (4.1)$$

where  $n_{\text{total}}$  is the total number of events in the dataset and  $n_{\text{reco}}$  the number of reconstructed events. The latter depends on the cleaning, as only a successfully cleaned event can be parametrized and therefore reconstructed. By first binning the efficiency per energy and then taking the sum over all energy bins, one can sort the datasets by setting upper and lower bounds for the efficiency. The intervals' lengths were chosen as 0.05, i. e. 5 %, resulting in a total of 20 intervals between 0 % and 100 %.

For each interval, the mean angular resolution is calculated by determining the 68 % containment of the angular distance distribution, i. e. the angular distance between the reconstructed origin and the true origin of the shower. The dataset with the lowest mean angular resolution is the best performing for each interval. If more low-energy events are reconstructed, not only will the efficiency be higher, but also the angular resolution, as those events are more likely to include pixels that are part of the NSB and also include fewer signal pixels than a high-energy event. As a result, a trade-off between efficiency and angular resolution is necessary.

For the performance analysis of the algorithms, I calculated metrics such as, but not limited to, the True Positive Rate (TPR) or recall, the False Positive Rate (FPR) or fall-out and the True Negative Rate (TNR) or specificity. A list of all used metrics can be seen in Table 2 in appendix 4.1.

# Results

# 5

In this chapter, the results of the analysis are presented. First, I present the results of the efficiency analysis in section 5.1. The initial tests of parameter combinations are based on a small dataset consisting of 20 runs, i. e. 12 668 events. This is due to the number of parameter combinations that are tested, as more runs would increase the processing time immensely. Furthermore, I show the results of the angular resolution for a combined metric with the efficiency. Then, in section 5.2, the metrics of each resulting combination of hyperparameters are plotted. In section 5.3, the performance of each cleaning algorithm compared to the default settings is shown. Finally, a comparison of the different cleaning algorithms is provided in section 5.4.

## 5.1 Analysis of the Efficiency and the Angular Resolution

To narrow down possible candidates for optimal hyperparameters, I first analyze the efficiency of the different cleaning algorithms. The efficiency is determined by the number of events that are reconstructed after cleaning. For this work I chose 20 intervals within 0 and 1 with a step size of 0.05. The number of reconstructed events  $n_{\text{reco}}$  and the total number of events  $n_{\text{total}}$  are binned per energy and the efficiency is then calculated according to Equation 4.1.

For each interval only those datasets are selected, where the efficiency lies between the lower and upper bound of the interval. Then the minimum angular resolution is determined for each interval. The parameters of these datasets are then selected to be the optimal parameters for each cleaning algorithm. This not only allows for a comparison of the cleaners but also a decision on a trade-off between the efficiency and the angular resolution, namely having a better angular resolution, but a lower efficiency, or a higher efficiency but also a higher and therefore worse angular resolution. The results for the efficiency are listed in Table 5.1 and the results for the mean angular resolution in Table 5.2.

As one can see, not all cleaning algorithms have valid values for each interval, peaking at a maximum of around 40 % to 45 % of successfully reconstructed events. This is because not all events are stereo events, i. e. events, where two or more telescopes were triggered, which is a requirement for the reconstruction. The remaining events are therefore mono events and do not contribute to either the efficiency or the angular resolution. The efficiency would be higher, of course, for a full-array analysis, but that would mean also including LSTs data<sup>1</sup>, which for this work would not help find the optimal parameters, since it is better to analyze the telescope types separately. The reason for the latter is, that optimizing the telescopes by type would lead to better results for the hyperparameters for the respective telescopes.

The results listed in the tables show that there is a clear trade-off in choosing between efficiency and angular resolution. As such, for further comparison of the cleaning algorithms, the corresponding datasets for the efficiency and the angular resolution are plotted in Figure 5.1 for the intervals [0.10, 0.15]

<sup>1</sup>which would result in additional long run times for the pipeline and was thus left out due to time limitations, also see chapter 6.

**Table 5.1:** The results of the analysis for the efficiency of each cleaning algorithm taken over all energy bins. The table lists the lower and upper limits of each efficiency interval. The efficiency is calculated according to Equation 4.1 and each listed efficiency is the one where the mean angular resolution is minimal for the given interval. Notice how not all cleaning algorithms have valid results for all efficiency intervals, due to not all events being stereo events.

		Efficiency			
Eff <sub>lower</sub>	Eff <sub>upper</sub>	Tailcuts	MARS	FACT	TCC
0.00	0.05		0.007		
0.05	0.10		0.051	0.070	
0.10	0.15	0.121	0.122	0.141	0.121
0.15	0.20	0.177	0.179	0.178	0.176
0.20	0.25	0.242	0.244	0.250	0.248
0.25	0.30	0.296	0.297	0.254	0.263
0.30	0.35	0.307	0.306	0.340	0.311
0.35	0.40	0.365	0.380	0.388	0.375
0.40	0.45	0.401	0.410	0.405	0.401

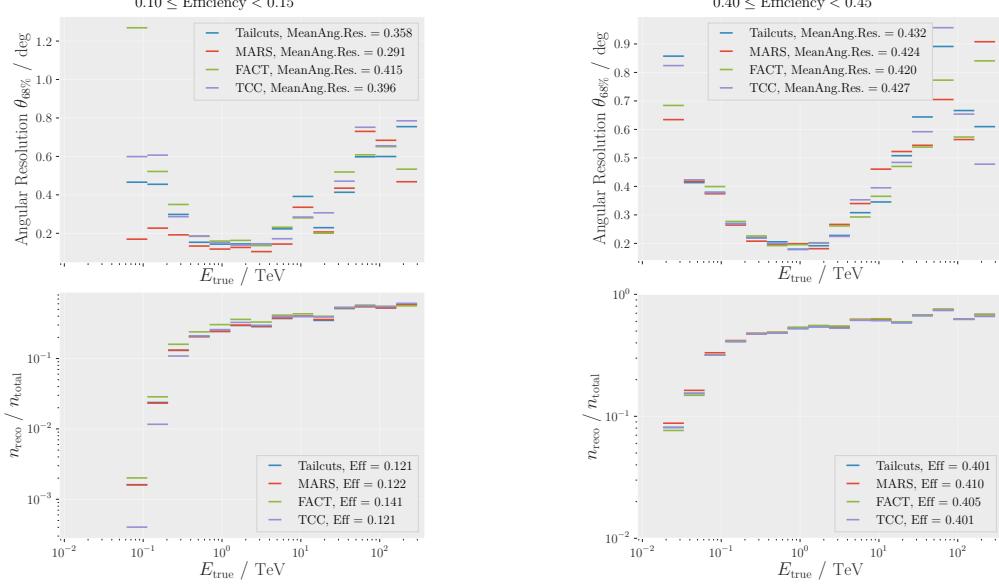
**Table 5.2:** The results of the analysis for the mean angular resolution of each cleaning algorithm. The table lists the lower and upper limits of each efficiency interval. The angular resolution listed is the minimum mean angular resolution of the respective efficiency interval. The corresponding efficiency values are listed in Table 5.1. Notice how not all cleaning algorithms have valid results for all efficiency intervals, due to not all events being stereo events.

		Mean Angular Resolution / deg			
Eff <sub>lower</sub>	Eff <sub>upper</sub>	Tailcuts	MARS	FACT	TCC
0.00	0.05		0.244		
0.05	0.10		0.428	0.413	
0.10	0.15	0.358	0.291	0.415	0.396
0.15	0.20	0.308	0.282	0.367	0.340
0.20	0.25	0.334	0.332	0.403	0.388
0.25	0.30	0.346	0.356	0.366	0.386
0.30	0.35	0.357	0.343	0.365	0.383
0.35	0.40	0.395	0.395	0.404	0.390
0.40	0.45	0.432	0.424	0.420	0.427

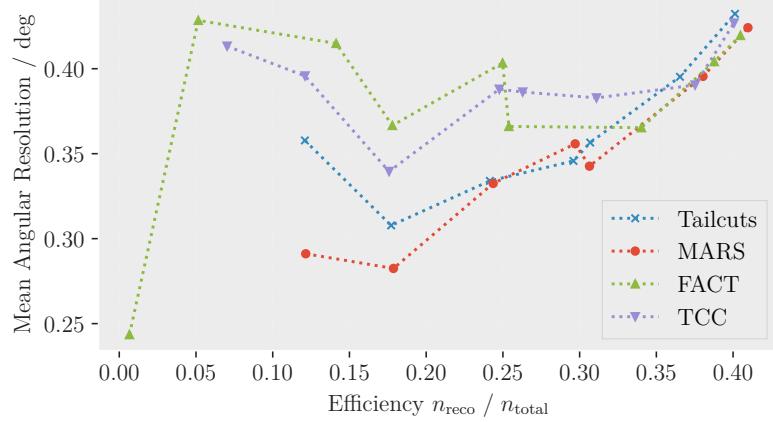
## 5 Results

---

and  $[0.40, 0.45]$ . The reason for this is, that these are the minimum and maximum intervals w.r.t. the efficiency, where all cleaners have valid values. Furthermore, the mean angular resolution is plotted against the efficiency in Figure 5.2.



**Figure 5.1:** Mean angular resolution and efficiency for the MST simulation binned per energy. Notice the decrease in the scattering of the mean angular resolution and the efficiency at medium to medium-high energies at higher efficiencies in the plots on the right.



**Figure 5.2:** Mean angular resolution plotted against the efficiency between 0.00 and 0.45 for all four cleaning algorithms. For further analysis, I prioritize a higher efficiency over the angular resolution as this would leave me with more reconstructed events and thus more statistics. This, however, results in a higher and therefore worse angular resolution compared to e.g. the values around an efficiency of 0.18.

## 5.2 Metrics of the Cleaning Algorithms on the Image Level

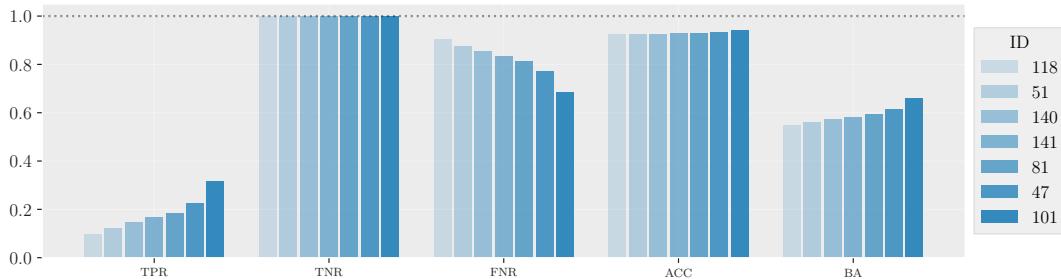
Concerning the metrics on the image level, I first analyze the parameter combinations of all settings found in section 5.1. First, the number of the True Positive (tp), False Positive (fp), False Negative (fn) and True Negative (tn) values is calculated per event. Then, those values are summed up for each setting from the datasets in the previous section and the metrics are calculated as shown in Table 2.

The metrics for each parameter combination are first compared for each cleaning algorithm respectively, to determine the best setting within a set of parameter combinations. The results are then plotted in Figures 5.3, 5.4, 5.5 and 5.6 and are shown with specific IDs they were given when the datasets were first processed with `ctapipe`. This improves readability as opposed to writing out the full settings. The hyperparameters for the best performing IDs of every algorithm are listed in Table 5.7.

While all settings perform well w. r. t., for example, the TNR, one can see from the results, that for all cleaning algorithms, the best parameter combination is the one where the TPR, the Accuracy (ACC) and the Balanced Accuracy (BA) are the highest. These settings are also the ones that correspond to the highest efficiency. The values for each respective metric and cleaning algorithm are listed in Tables 5.3, 5.4, 5.5 and 5.6 respectively, while a comparison of the cleaners against each other is shown in section 5.4.

**Table 5.3:** Results for the metrics of Tailcuts. One can see, that the best results are obtained for the setting with ID 101.

ID	TPR	TNR	FNR	ACC	BA
118	0.0972	1.0000	0.9028	0.9256	0.5486
51	0.1231	1.0000	0.8769	0.9256	0.5615
140	0.1460	1.0000	0.8540	0.9256	0.5730
141	0.1660	1.0000	0.8340	0.9283	0.5830
81	0.1854	1.0000	0.8146	0.9288	0.5927
47	0.2268	1.0000	0.7732	0.9332	0.6134
101	0.3169	0.9997	0.6831	0.9418	0.6583



**Figure 5.3:** Metrics for Tailcuts. It can be seen, that the cleaning setting with ID 101 performs best in terms of the true positive rate, accuracy and balanced accuracy, making it the best setting of the five parameter combinations w. r. t. the metrics.

**Table 5.4:** Results for the metrics of MARS. The best results w. r. t. the metrics are obtained for the setting with ID 13.

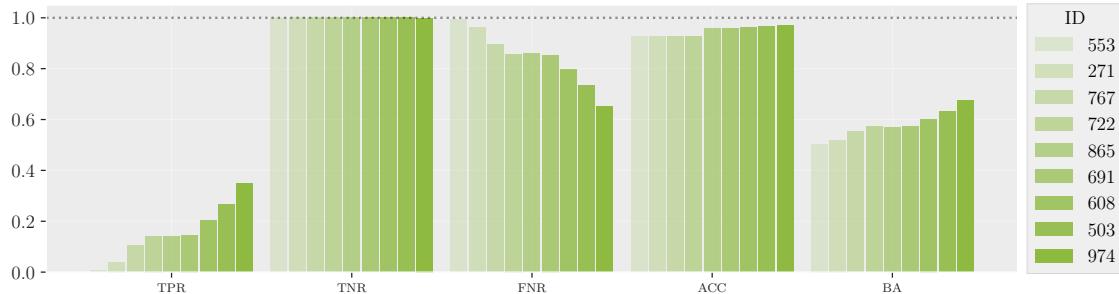
ID	TPR	TNR	FNR	ACC	BA
54	0.1069	1.0000	0.8931	0.9256	0.5534
59	0.1369	1.0000	0.8631	0.9256	0.5684
76	0.1565	1.0000	0.8435	0.9256	0.5782
60	0.1625	1.0000	0.8375	0.9283	0.5812
135	0.1738	1.0000	0.8262	0.9607	0.5869
15	0.2433	1.0000	0.7567	0.9640	0.6216
13	0.3460	0.9995	0.6540	0.9684	0.6727



**Figure 5.4:** Metrics for MARS. The results show that the cleaning setting with ID 13 performs best in terms of the true positive rate, accuracy and balanced accuracy, making it the best setting of the five parameter combinations w. r. t. the metrics.

**Table 5.5:** Results for the metrics of FACT. One can see, that the best results are obtained for the settings with ID 974.

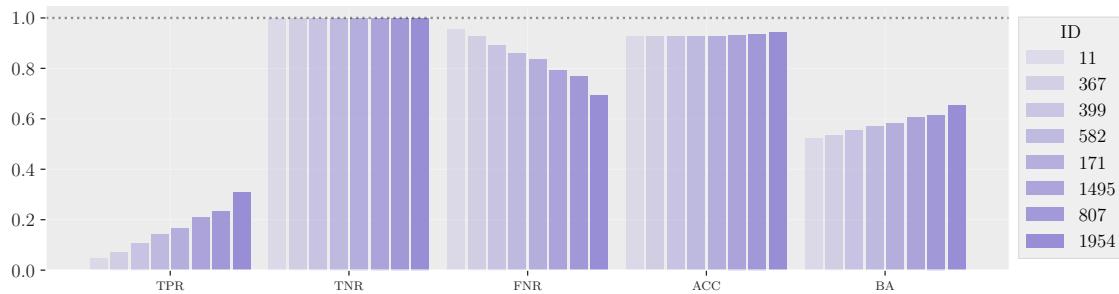
ID	TPR	TNR	FNR	ACC	BA
553	0.0077	1.0000	0.9923	0.9256	0.5039
271	0.0396	1.0000	0.9604	0.9256	0.5198
767	0.1042	1.0000	0.8958	0.9256	0.5521
722	0.1425	1.0000	0.8575	0.9256	0.5712
865	0.1396	1.0000	0.8604	0.9591	0.5698
691	0.1464	1.0000	0.8536	0.9594	0.5732
608	0.2040	1.0000	0.7960	0.9621	0.6020
503	0.2655	1.0000	0.7345	0.9650	0.6327
974	0.3493	0.9996	0.6507	0.9687	0.6745



**Figure 5.5:** Metrics for FACT. The cleaning setting with ID 947 performs best in terms of the number of true positives, accuracy and balanced accuracy, making it the best setting of the nine parameter combinations.

**Table 5.6:** Results for the metrics of TCC. The best results are obtained with the setting with ID 1954.

ID	TPR	TNR	FNR	ACC	BA
11	0.0452	1.0000	0.9548	0.9256	0.5226
367	0.0717	1.0000	0.9283	0.9256	0.5359
399	0.1071	1.0000	0.8929	0.9256	0.5535
582	0.1403	1.0000	0.8597	0.9267	0.5701
171	0.1653	1.0000	0.8347	0.9278	0.5827
1495	0.2088	1.0000	0.7912	0.9321	0.6044
807	0.2314	1.0000	0.7686	0.9364	0.6157
1954	0.3065	0.9998	0.6935	0.9418	0.6531



**Figure 5.6:** Metrics for the TCC. The cleaning setting with ID 1954 performs best in terms of the number of true positives, accuracy and balanced accuracy, making it the best setting of the eight parameter combinations.

**Table 5.7:** Best-performing IDs w. r. t. the metrics and the corresponding hyperparameters for each respective cleaner. Listed are the core threshold  $Q_c$ , the boundary threshold  $Q_b$ , the minimum number of neighbors and, where applicable, the time limit  $t$  as well as the core and boundary time limits  $t_c$  and  $t_b$ .

Cleaning Algorithm	ID	$Q_c$ / p.e.	$Q_b$ / p.e.	Min. Neigh.	$t$ / ns	$t_c$ / ns	$t_b$ / ns
Tailcuts	101	6.700	1.675	1			
MARS	13	5.500	1.833	2			
FACT	974	4.200	1.400	2	6.0		
TCC	1954	6.700	1.675	1		15.0	9.0

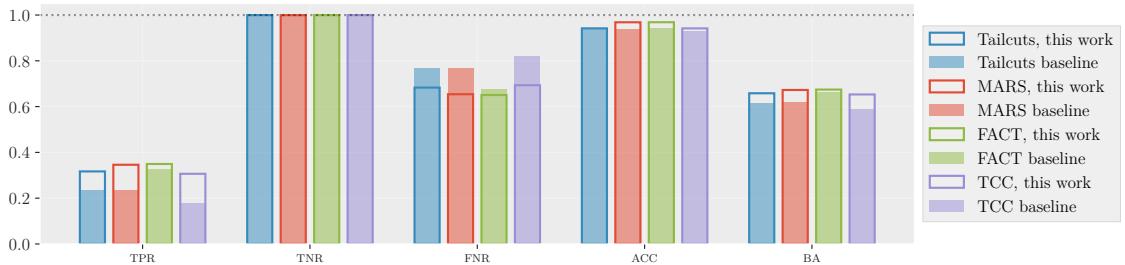
### 5.3 Performance Compared to the Default Settings

Now, with a setting selected for each cleaner, it is reasonable to compare the performance to the default settings, that are implemented in the `ctapipe` source code (see Table 4.1). First, the relative angular resolution

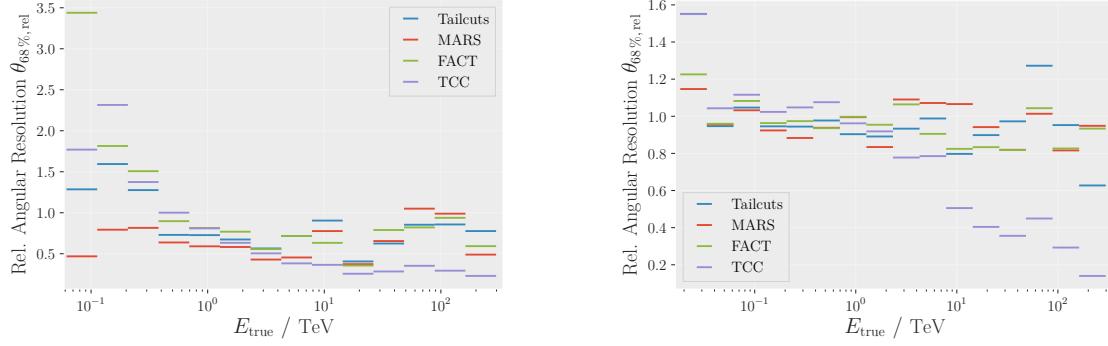
$$\theta_{68\%, \text{rel}} = \frac{\theta_{68\%, \text{rel}}}{\theta_{68\%, \text{base}}} \quad (5.1)$$

is computed for the default (base) settings and the selected settings for each cleaner. The results are shown in Figure 5.8 for the same efficiency intervals as in Figure 5.1. One can see, that especially for the interval [0.40, 0.45] and at medium to high energies, TCC performs fairly well compared to the default settings. The other algorithms perform only slightly better than the default settings at a performance gain of up to 50 % e. g. for MARS. Notably for MARS is how it performs consistently across the whole energy range.

Furthermore, the metrics of the hyperparameters determined in this work can be compared to the metrics of the default settings. The results are shown in Figure 5.7 with the values of the metrics being listed in Table 5.8. The values for the selected settings in this work are listed in Table 5.9. With the hyperparameters found in this work, all four algorithms perform better than with the default settings. Especially TCC and Tailcuts and MARS gain a significant improvement, most notably in the TPR and BA scores. The improvement for TCC is even more notable than for MARS and Tailcuts. FACT also improved in terms of the TPR and ACC, but not as much as the rest.



**Figure 5.7:** Metrics for the default (baseline) settings compared to the settings determined in this work. One can see that a significant improvement was reached for TCC and Tailcuts and MARS, especially in terms of TPR and BA. FACT did only slightly improve, especially in terms of ACC.



**Figure 5.8:** Relative angular resolution for the default settings and the selected settings for each cleaner. TCC outperforms the default values with the new settings, especially at higher energies. MARS seems to perform better across the whole energy range with the found hyperparameters. FACT and Tailcuts, too, perform better, especially for medium to high energies.

**Table 5.8:** Metrics for the default (baseline) settings. The found hyperparameters listed in Table 5.7 improve the performance of each cleaner over the default settings. Especially TCC and Tailcuts gained better scores w. r. t. the metrics.

Cleaning Algorithm	TPR	TNR	FNR	ACC	BA
Tailcuts	0.2316	1.0000	0.7684	0.9358	0.6158
MARS	0.2346	1.0000	0.7654	0.9358	0.6173
FACT	0.3261	0.9998	0.6739	0.9429	0.6629
TCC	0.1790	1.0000	0.8210	0.9299	0.5895

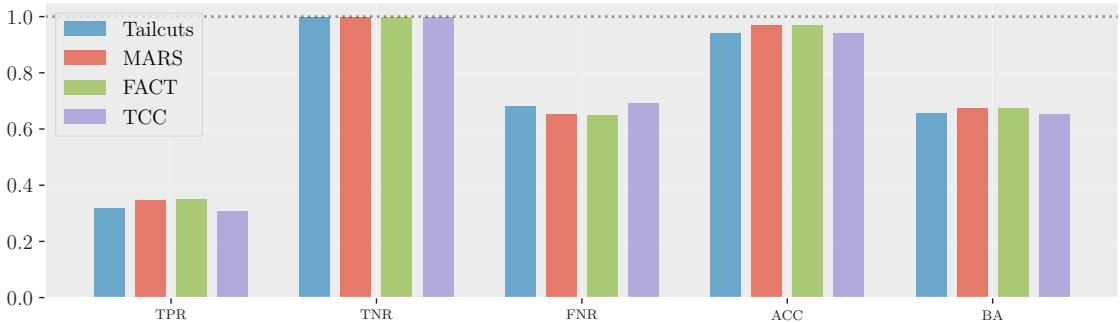
## 5.4 Comparison of the Cleaning Algorithms

As of writing this thesis, the cleaning algorithms discussed in this thesis are the only ones implemented in the `ctapipe` source code. With the determined hyperparameters, I decided to compare the performance of all four cleaners. I once again decided to use the settings selected in section 5.2 for each algorithm and look at the metrics. The metrics of all four algorithms are shown in Figure 5.9. The corresponding values are listed in Table 5.9.

One can see, that FACT performs best out of all algorithms w. r. t. the metrics, closely followed by MARS. Both algorithms perform well in terms of TPR and BA. Tailcuts and TCC, although performing slightly worse than FACT and MARS, still perform fairly well. Both have the same scores for the ACC and a similar score for the BA.

**Table 5.9:** Metrics for the selected settings of each cleaning algorithm. Out of these four algorithms, FACT performs best in terms of TPR and BA with MARS following closely. Tailcuts and TCC, although slightly worse than the other two, still perform fairly well. MARS, due to its consistency for the angular resolution, is a viable alternative to FACT.

Cleaning Algorithm	TPR	TNR	FNR	ACC	BA
Tailcuts	0.3169	0.9997	0.6831	0.9418	0.6583
MARS	0.3460	0.9995	0.6540	0.9684	0.6727
FACT	0.3493	0.9996	0.6507	0.9687	0.6745
TCC	0.3065	0.9998	0.6935	0.9418	0.6531



**Figure 5.9:** Bar plot visualizing the metrics for the selected settings of all four cleaning algorithms. FACT performs best out of all cleaning algorithms, followed by MARS. The latter, however performs consistently w. r. t. the angular resolution, as shown in Figure 5.8 and can therefore be considered a viable alternative to FACT. Tailcuts and FACT perform slightly worse than the other two, but still fairly well.

# Conclusions and Outlook

---

In the scope of this work, the hyperparameters for each of the four currently implemented cleaning algorithms were searched via a grid search. The resulting hyperparameters were then probed w. r. t. a combined metric of the angular resolution and the efficiency. These results allowed for a selection of hyperparameters for further study. Finally, a comparison between the optimized and the default parameters as well as a comparison between each respective cleaner was performed.

Most of the data (pre-)processing in this work was done with CTA’s open-source low-level data processing pipeline software `ctapipe`. The data used was PROD5 MC simulations, which was processed from raw `simtel` data (**R0**) up to cleaned (**DL1a**) and parametrized (**DL1b**) levels for the metrics as well as reconstructed (**DL2**) data for the efficiency and angular resolution. Further high-level processing was done with this works pipeline as described in section 3.3. The four implemented cleaning algorithms can be roughly categorized into time-based (FACT and TCC) and non-time-based algorithms (Tailcuts and MARS). The time-based algorithms have the advantage to also set limits on the arrival time of each pixel. This allows for a lower core threshold  $Q_c$  for the photon count per pixel, which in turn can lead to better results in the cleaning process. A good cleaning allows for better parameterization and in turn a better reconstruction of the events.

The main results of this thesis are the hyperparameters for the cleaning algorithms. Due to the long runtimes of the grid searches and the time limitation of this thesis, however, the hyperparameters for cleaning algorithms could only be optimized for the MSTs. The optimal hyperparameters of each cleaning algorithm for the MSTs are listed in Table 5.7. A further grid search for the LST data would be necessary to find the remaining core and boundary thresholds  $Q_c$  and  $Q_b$  as well as the time limits.

The comparison of the performance of the optimized algorithms with the default algorithms has shown that the optimized algorithms performed better than the default algorithms w. r. t. the angular resolution—especially for medium to high energies. This means that the found hyperparameters could help with origin reconstructions. When looking at the metrics, all algorithms but FACT improved significantly, with the latter improving just slightly over its performance with default settings. The reason is, that the parameters selected in Table 5.7 are very close to the default parameters (see Table 4.1). The parameters for the other algorithms differ more from their default settings, thus leading to a more significant improvement. Additionally, a comparison between the optimized algorithms themselves was performed. This led to FACT and MARS being the best-performing algorithms overall w. r. t. the metrics, with MARS showing a consistency w. r. t. the angular resolution. Tailcuts and TCC, however, still performed fairly well.

In this work, I prioritized the efficiency over the angular resolution as this led to a better performance w. r. t. the image level metrics. This is due to the fact that a higher efficiency means that more events got reconstructed and therefore better overall statistics. Another user, however, might still want to choose the angular resolution over the efficiency if it fits their purpose. The parameters for all other parameter combinations can be found in Table 3 in appendix 4.

Finally, although the results already promising, further analysis and a broader grid search may lead to even better-suited hyperparameters and therefore better the performance of all algorithms. This is especially true for the LST data, which was left out due to the aforementioned long run times of the pipeline.

# Bibliography

---

1. B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration Collaboration). “Observation of Gravitational Waves from a Binary Black Hole Merger.” *Phys. Rev. Lett.* 116, 6 2016, page 061102.  
DOI: [10.1103/PhysRevLett.116.061102](https://doi.org/10.1103/PhysRevLett.116.061102)
2. S. Abdollahi et al. “*Fermi* Large Area Telescope Fourth Source Catalog.” *The Astrophysical Journal Supplement Series* 247:1, 2020, page 33.  
DOI: [10.3847/1538-4365/ab6bcb](https://doi.org/10.3847/1538-4365/ab6bcb)
3. S. Abdollahi et al. “Incremental *Fermi* Large Area Telescope Fourth Source Catalog.” *The Astrophysical Journal Supplement Series* 260:2, 2022, page 53.  
DOI: [10.3847/1538-4365/ac6751](https://doi.org/10.3847/1538-4365/ac6751). ARXIV: [2201.11184 \[astro-ph.HE\]](https://arxiv.org/abs/2201.11184)
4. A. U. Abeysekara et al. “Daily Monitoring of TeV Gamma-Ray Emission from Mrk 421, Mrk 501, and the Crab Nebula with HAWC.” *The Astrophysical Journal* 841:2, 2017, page 100.  
DOI: [10.3847/1538-4357/aa729e](https://doi.org/10.3847/1538-4357/aa729e)
5. A. Achterberg et al. “First year performance of the IceCube neutrino telescope.” *Astroparticle Physics* 26:3, 2006, pages 155–173.  
DOI: [10.1016/j.astropartphys.2006.06.007](https://doi.org/10.1016/j.astropartphys.2006.06.007). ARXIV: [astro-ph/0604450 \[astro-ph\]](https://arxiv.org/abs/astro-ph/0604450)
6. J. A. Aguilar, J. Yang, and S. Bravo. *Neutrinos and gamma rays, a partnership to explore the extreme universe*. IceCube/WIPAC. 2016.  
<https://icecube.wisc.edu/news/view/455> visited on 2022-08-05
7. Astropy Collaboration et al. “Astropy: A community Python package for astronomy.” *Astronomy & Astrophysics* 558, A33, 2013, A33.  
DOI: [10.1051/0004-6361/201322068](https://doi.org/10.1051/0004-6361/201322068). ARXIV: [1307.6212 \[astro-ph.IM\]](https://arxiv.org/abs/1307.6212)
8. Astropy Collaboration et al. “The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package.” *The Astronomical Journal* 156:3, 123, 2018, page 123.  
DOI: [10.3847/1538-3881/aabc4f](https://doi.org/10.3847/1538-3881/aabc4f). ARXIV: [1801.02634 \[astro-ph.IM\]](https://arxiv.org/abs/1801.02634)
9. K. Bernlöhr. “Simulation of imaging atmospheric Cherenkov telescopes with CORSIKA and sim\_telarray.” *Astroparticle Physics* 30:3, 2008, pages 149–158.  
DOI: [10.1016/j.astropartphys.2008.07.009](https://doi.org/10.1016/j.astropartphys.2008.07.009). <https://www.sciencedirect.com/science/article/pii/S0927650508000972>
10. K. Brügge. “Unmasking the Gamma-Ray Sky. Comprehensive and Reproducible Analysis for Cherenkov Telescopes.” PhD thesis. Technische Universität Dortmund, 2019

11. Cherenkov Telescope Array Consortium et al. *Science with the Cherenkov Telescope Array*. WORLD SCIENTIFIC, 2019.  
DOI: [10.1142/10986](https://doi.org/10.1142/10986). ARXIV: [1709.07997v2 \[astro-ph.IM\]](https://arxiv.org/abs/1709.07997v2)
12. C. L. Cowan et al. “Detection of the Free Neutrino: a Confirmation.” *Science* 124:3212, 1956, pages 103–104.  
DOI: [10.1126/science.124.3212.103](https://doi.org/10.1126/science.124.3212.103)
13. *CTA Technology*. Cherenkov Telescope Array Consortium.  
[https://www.cta-observatory.org/wp-content/uploads/2019/12/CTA-Specifications\\_v08\\_formatted.pdf](https://www.cta-observatory.org/wp-content/uploads/2019/12/CTA-Specifications_v08_formatted.pdf) visited on 2022-08-10
14. *CTAO’s site in the northern hemisphere*. Cherenkov Telescope Array Consortium.  
<https://www.cta-observatory.org/about/array-locations/la-palma/> visited on 2022-08-10
15. *CTAO’s site in the southern hemisphere*. Cherenkov Telescope Array Consortium.  
<https://www.cta-observatory.org/about/array-locations/chile/> visited on 2022-08-10
16. R. Davis. “Solar Neutrinos. II. Experimental.” *Phys. Rev. Lett.* 12, 11 1964, pages 303–305.  
DOI: [10.1103/PhysRevLett.12.303](https://doi.org/10.1103/PhysRevLett.12.303)
17. C. Evoli. *The Cosmic-Ray Energy Spectrum*. 2020.  
DOI: [10.5281/zenodo.4396125](https://doi.org/10.5281/zenodo.4396125)
18. S. Funk. “Ground- and Space-Based Gamma-Ray Astronomy.” *Annual Review of Nuclear and Particle Science* 65:1, 2015, pages 245–277.  
DOI: [10.1146/annurev-nucl-102014-022036](https://doi.org/10.1146/annurev-nucl-102014-022036)
19. J. Hackfeld. “Analyzing the Data Volume Reduction for the LST-1 Prototype of the Cherenkov Telescope Array.” MA thesis. Ruhr-Universität Bochum, 2021
20. C. R. Harris et al. “Array programming with NumPy.” *Nature* 585:7825, 2020, pages 357–362.  
DOI: [10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2)
21. D. Heck et al. *CORSIKA: a Monte Carlo code to simulate extensive air showers*. 1998.  
<https://ui.adsabs.harvard.edu/abs/1998cmcc.book.....H>.  
Note: Provided by the SAO/NASA Astrophysics Data System
22. K. Hirata et al. “Observation of a neutrino burst from the supernova SN1987A.” *Phys. Rev. Lett.* 58, 14 1987, pages 1490–1493.  
DOI: [10.1103/PhysRevLett.58.1490](https://doi.org/10.1103/PhysRevLett.58.1490)
23. J. D. Hunter. “Matplotlib: A 2D graphics environment.” *Computing in Science & Engineering* 9:3, 2007, pages 90–95.  
DOI: [10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55)
24. K. Kosack et al. *cta-observatory/ctapipe: v0.15.0 – 2022-05-23*. Version v0.15.0. 2022.  
DOI: [10.5281/zenodo.6572720](https://doi.org/10.5281/zenodo.6572720)
25. W. L. Kraushaar and G. W. Clark. “Search for Primary Cosmic Gamma Rays with the Satellite Explorer XI.” *Phys. Rev. Lett.* 8, 3 1962, pages 106–109.  
DOI: [10.1103/PhysRevLett.8.106](https://doi.org/10.1103/PhysRevLett.8.106)

## Bibliography

---

26. M. S. Longair. *High energy astrophysics: an informal introduction for students of physics and astronomy*. Cambridge University Press, Cambridge [England] ; New York, 1981.  
ISBN: 9780521235136, 9780521280136
27. M. Nöthe. “Monitoring the High Energy Universe. Open, Reproducible, Machine Learning Based Analysis for the first G-APD Cherenkov Telescope.” PhD thesis. Technische Universität Dortmund, 2020
28. M. Nöthe et al. *cta-observatory/pyirf: v0.7.0 – 2022-04-19*. Version v0.7.0. 2022.  
DOI: [10.5281/zenodo.6469902](https://doi.org/10.5281/zenodo.6469902)
29. F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12, 2011, pages 2825–2830
30. M. Shayduk. “A new image cleaning method for the MAGIC Telescope,” 2005
31. Y. Suzuki. “The Super-Kamiokande experiment.” *The European Physical Journal C* 79, 2019.  
DOI: [10.1140/epjc/s10052-019-6796-2](https://doi.org/10.1140/epjc/s10052-019-6796-2)
32. F. Temme. “FACT - Data Analysis. Analysis of Crab Nebula Data using PARFACT, a newly Developed Analysis Software for the First G-APD Cherenkov Telescope.” Diploma Thesis. Technische Universität Dortmund, 2013
33. F. Temme. “On the Hunt for Photons. Analysis of Crab Nebula Data obtained by the First G-APD Cherenkov Telescope.” PhD thesis. Technische Universität Dortmund, 2016
34. The pandas development team. *pandas-dev/pandas: Pandas*. 2020.  
DOI: [10.5281/zenodo.3509134](https://doi.org/10.5281/zenodo.3509134)
35. *The technology behind the next generation very-high energy gamma-ray detector*. Cherenkov Telescope Array Consortium.  
<https://www.cta-observatory.org/project/technology/> visited on 2022-08-11
36. R. Zanin et al. “MARS, The MAGIC Analysis and Reconstruction Software.” In: *International Cosmic Ray Conference*. Vol. 33. International Cosmic Ray Conference. 2013, page 2937

# Glossary

---

**ACC** Accuracy. 19–24, 37

**AGN** active galactic nucleus. 2, 3

**BA** Balanced Accuracy. 19–24, 37

**CORSIKA** Cosmic Ray Simulations for Kascade. 8

**CR** cosmic rays. 1–3

**CTA** Cherenkov Telescope Array. iii, 4, 6–8, 25

**EAS** extensive air shower. 3–5, 8

**ESO** European Southern Observatory. 6

**FACT** FACTImageCleaner. iii, 12–15, 20–25, 33, 36, 38, 39

**Fermi-LAT** *Fermi* Large Area Telescope. 2–4

**fn** False Negative. 19, 37

**FNR** False Negative Rate. 19–21, 23, 24, 37

**fp** False Positive. 19, 37

**FPR** False Positive Rate. 15

**GRB** gamma ray burst. 2

**HAWC** High Altitude Water Cherenkov. 4

**HE** high-energy. 2, 4, 7, 8

**H.E.S.S.** High Energy Stereoscopic System. 3, 7

**IACT** Imaging Air Cherenkov Telescope. iii, 3–6, 8

## *Glossary*

---

**LST** Large-Sized Telescope. 6, 7, 16, 25, 35

**MAGIC** Major Atmospheric Gamma-Ray Imaging Cherenkov. 3, 7, 13

**MARS** MARSIImageCleaner. iii, 11, 12, 14, 15, 20, 22–25, 32, 38, 39

**MC** Monte Carlo. iii, 8, 25

**MST** Medium-Sized Telescope. iii, 6, 7, 10–13, 15, 25, 35

**NSB** Night Sky Background. 5, 9, 15

**ORM** Observatorio del Roque de los Muchachos. 6

**PMT** photo multiplier tube. 7

**PWN** pulsar wind nebula. 7

**SiPM** silicon photomultiplier. 7

**SNR** supernova remnant. 2, 3

**SST** Small-Sized Telescope. 6, 7

**Tailcuts** TailcutsImageCleaner. 11, 12, 14, 15, 19, 22–25, 32, 38, 39

**TCC** TimeConstrainedImageCleaner. iii, 13–15, 21–25, 34, 36, 38, 39

**tn** True Negative. 19, 37

**TNR** True Negative Rate. 15, 19–21, 23, 24, 37

**tp** True Positive. 19, 37

**TPR** True Positive Rate. 15, 19–24, 37

**VERITAS** Very Energetic Radiation Imaging Telescope Array System. 3

**VHE** very-high-energy. 3, 7

# Appendix

---

## 1 Configurations for ctapipe

The listing below shows the configuration file used for preprocessing the datasets, i. e. before the application of the different cleaning settings.

```
DataWriter:  
    transform_image: true  
    transform_peak_time: true  
    write_images: true  
    write_parameters: false  
    write_raw_waveforms: false  
    write_showers: true  
  
ProcessorTool:  
    progress_bar: true  
  
CameraCalibrator:  
    image_extractor_type: NeighborPeakWindowSum  
  
ImageProcessor:  
    image_cleaner_type: TailcutsImageCleaner  
  
    TailcutsImageCleaner:  
        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"]  
  
ShowerProcessor:  
    reconstructor_type: HillasReconstructor  
    HillasReconstructor:  
        StereoQualityQuery:  
            quality_criteria:  
                - [enough intensity, "parameters.hillas.intensity > 50"]  
                - [Positive width, "parameters.hillas.width.value > 0"]  
                - [enough pixels, "parameters.morphology.num_pixels > 3"]  
                - [not clipped, "parameters.leakage.intensity_width_2 < 0.5"]
```

```
SimTelEventSource:  
    focal_length_choice: effective  
    skip_calibration_events: true
```

The listings below show the contents of the default configuration files used with ctapipe for each cleaning algorithm respectively.

**TailcutsImageCleaner (Tailcuts):**

```
DataWriter:  
    transform_image: true  
    transform_peak_time: true  
    write_images: false  
    write_parameters: true  
    write_raw_waveforms: false  
    write_showers: true  
  
ProcessorTool:  
    progress_bar: true  
  
CameraCalibrator:  
    image_extractor_type: NeighborPeakWindowSum  
  
ImageProcessor:  
    image_cleaner_type: TailcutsImageCleaner  
  
TailcutsImageCleaner:  
    picture_threshold_pe:  
        - [type, "LST*", 7.0]  
        - [type, "MST*NectarCam", 7.0]  
    boundary_threshold_pe:  
        - [type, "LST*", 5.0]  
        - [type, "MST*NectarCam", 5.0]  
    keep_isolated_pixels: false  
    min_picture_neighbors: 0  
  
ImageQualityQuery:  
    quality_criteria:  
        - ["enough_pixels", "np.count_nonzero(image) > 2"]  
        - ["enough_charge", "image.sum() > 50"]  
  
ShowerProcessor:  
    reconstructor_type: HillasReconstructor  
    HillasReconstructor:  
        StereoQualityQuery:  
            quality_criteria:  
                - [enough intensity, "parameters.hillas.intensity > 50"]  
                - [Positive width, "parameters.hillas.width.value > 0"]  
                - [enough pixels, "parameters.morphology.num_pixels > 3"]  
                - [not clipped, "parameters.leakage.intensity_width_2 < 0.5"]
```

**MARSImageCleaner (MARS):**

```
DataWriter:  
    transform_image: true
```

```

transform_peak_time: true
write_images: false
write_parameters: true
write_raw_waveforms: false
write_showers: true

ProcessorTool:
    progress_bar: true

CameraCalibrator:
    image_extractor_type: NeighborPeakWindowSum

ImageProcessor:
    image_cleaner_type: MARSImageCleaner

    MARSImageCleaner:
        picture_threshold_pe:
            - [type, "LST*", 7.0]
            - [type, "MST*NectarCam", 7.0]
        boundary_threshold_pe:
            - [type, "LST*", 5.0]
            - [type, "MST*NectarCam", 5.0]
        keep_isolated_pixels: false
        min_picture_neighbors: 0

    ImageQualityQuery:
        quality_criteria:
            - ["enough_pixels", "np.count_nonzero(image) > 2"]
            - ["enough_charge", "image.sum() > 50"]

ShowerProcessor:
    reconstructor_type: HillasReconstructor
    HillasReconstructor:
        StereoQualityQuery:
            quality_criteria:
                - [enough intensity, "parameters.hillas.intensity > 50"]
                - [Positive width, "parameters.hillas.width.value > 0"]
                - [enough pixels, "parameters.morphology.num_pixels > 3"]
                - [not clipped, "parameters.leakage.intensity_width_2 < 0.5"]

```

**FACTImageCleaner (FACT):**

```

DataWriter:
    transform_image: true
    transform_peak_time: true
    write_images: false
    write_parameters: true
    write_raw_waveforms: false
    write_showers: true

ProcessorTool:
    progress_bar: true

CameraCalibrator:
    image_extractor_type: NeighborPeakWindowSum

ImageProcessor:
    image_cleaner_type: FACTImageCleaner

```

```
FACTImageCleaner:
    picture_threshold_pe:
        - [type, "LST*", 7.0]
        - [type, "MST*NectarCam", 7.0]
    boundary_threshold_pe:
        - [type, "LST*", 5.0]
        - [type, "MST*NectarCam", 5.0]
    keep_isolated_pixels: false
    min_picture_neighbors: 2
    time_limit_ns: 5.0

ImageQualityQuery:
    quality_criteria:
        - ["enough_pixels", "np.count_nonzero(image) > 2"]
        - ["enough_charge", "image.sum() > 50"]

ShowerProcessor:
    reconstructor_type: HillasReconstructor
    HillasReconstructor:
        StereoQualityQuery:
            quality_criteria:
                - [enough intensity, "parameters.hillas.intensity > 50"]
                - [Positive width, "parameters.hillas.width.value > 0"]
                - [enough pixels, "parameters.morphology.num_pixels > 3"]
                - [not clipped, "parameters.leakage.intensity_width_2 < 0.5"]
```

**TimeConstrainedImageCleaner (TCC):**

```
DataWriter:
    transform_image: true
    transform_peak_time: true
    write_images: false
    write_parameters: true
    write_raw_waveforms: false
    write_showers: true

ProcessorTool:
    progress_bar: true

CameraCalibrator:
    image_extractor_type: NeighborPeakWindowSum

ImageProcessor:
    image_cleaner_type: TimeConstrainedImageCleaner

    TimeConstrainedImageCleaner:
        picture_threshold_pe:
            - [type, "LST*", 7.0]
            - [type, "MST*NectarCam", 7.0]
        boundary_threshold_pe:
            - [type, "LST*", 5.0]
            - [type, "MST*NectarCam", 5.0]
        keep_isolated_pixels: false
        min_picture_neighbors: 1
        time_limit_core_ns: 4.5
        time_limit_boundary_ns: 1.5
```

```

ImageQualityQuery:
    quality_criteria:
        - ["enough_pixels", "np.count_nonzero(image) > 2"]
        - ["enough_charge", "image.sum() > 50"]

ShowerProcessor:
    reconstructor_type: HillasReconstructor
    HillasReconstructor:
        StereoQualityQuery:
            quality_criteria:
                - [enough intensity, "parameters.hillas.intensity > 50"]
                - [Positive width, "parameters.hillas.width.value > 0"]
                - [enough pixels, "parameters.morphology.num_pixels > 3"]
                - [not clipped, "parameters.leakage.intensity_width_2 < 0.5"]

```

The following listing shows the contents of the `prod5b_lapalma_alpha.yml` configuration file, which is used to set the allowed telescope IDs for `ctapipe-process`.

```

# telescope ids of the 4 LST + 9 MST La Palma alpha configuration in Prod5b
# (out of 84 telescopes total)
EventSource:
    allowed_tels: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 19, 35]

```

Since the comparison of the cleaning algorithms also required differentiating between Medium-Sized Telescopes (MSTs) and Large-Sized Telescopes (LSTs), the following two listings show the contents of the `prod5b_lapalma_mst.yml` and `prod5b_lapalma_lst.yml` configuration files, respectively:

MSTs:

```

# telescope ids of the 9 MST of the La Palma alpha configuration
# in Prod5b
EventSource:
    allowed_tels: [5, 6, 7, 8, 9, 10, 11, 19, 35]

```

LSTs:

```

# telescope ids of the 4 LST of the La Palma alpha configuration
# in Prod5b
EventSource:
    allowed_tels: [1,2,3,4]

```

## 2 Hyperparameters

The following table shows the subsets of hyperparameters used in the grid search:

**Table 1:** Hyperparameters for the cleaning algorithms used in the grid search. The picture quantiles determine what percent of all pixels per event will be below the core threshold  $Q_c$ . The boundary threshold is calculated by multiplying the core threshold by the boundary threshold ratio. The time limit  $t$  is the parameter for FACT, while the time limits  $t_c$  and  $t_b$  are the time limits for TCC.

Picture Quantiles	Boundary Threshold Ratio	Minimum Neighbors	$t$ / ns	$t_c$ / ns	$t_b$ / ns
0.995	0.25	1	1.0	9.0	4.5
0.999	0.33	2	2.0	12.0	9.0
0.9992	0.5	3	4.0	15.0	12.0
0.9995	0.66	4	5.0	18.0	15.0
0.9997	0.75	5	6.0	20.0	
0.9999			10.0		
			12.0		

### 3 Software Used

This work was written and built with L<sup>A</sup>T<sub>E</sub>X and LuaT<sub>E</sub>X from T<sub>E</sub>XLive 2021 on both the Windows Subsystem for Linux (WSL, Ubuntu 20.04) on Windows 10 and Ubuntu 20.04. Also, I relied heavily on the python programming language, of which the most important libraries used for this work are listed below.

- numpy [20]
- pandas [34]
- matplotlib [23]
- astropy [7, 8]
- pyirf [28]
- scikit-learn [29]

For the processing of the datasets, I used the development version of ctapipe, more specifically, the code from the master branch from July 6 (f26107fe). A complete listing of all used python libraries can be found below:

```

name: cta-dev
channels:
- anaconda
- defaults
- conda-forge
dependencies:
- sphinx_rtd_theme=1.0.0
- sphinx=3.5.4
- jupyter=1.0.0
- matplotlib=3.5.1
- iminuit=2.11.2
- tqdm=4.64.0
- traitlets=5.1.1
- joblib=1.1.0
- sphinx-autodocapi=0.14.1
- pytest-runner=6.0.0
- pytables=3.7.0
- tomli=2.0.1
- psutil=5.9.0
- pytest-cov=3.0.0
- python=3.8.13
- pyyaml=6.0
- numba=0.55.1
- pytest=7.1.2
- pip=22.0.4
- jinja2=3.0.3
- xz=5.2.5
- nbsphinx=0.8.8
- scipy=1.8.0
- cython=0.29.28
- eventio=1.9.1
- pandas=1.4.2
- zlib=1.2.11
- black=22.3.0
- h5py=3.6.0
- numpydoc=1.2.1
- vitables=3.0.2
- wheel=0.37.1
- ipywidgets=7.7.0
- scikit-learn=1.0.2
- graphviz=3.0.0
- bokeh=2.4.2
- zstandard=0.17.0
- pre-commit=2.18.1
- numpy=1.21.6
- astropy=5.0.4
- setuptools=62.1.0
- ipython=8.2.0
- jupyterlab=3.3.4
- seaborn=0.11.2
- ipympl=0.9.1
- ca-certificates=2022.6.15
- certifi=2022.6.15
- openssl=3.0.5
- gammipy=0.20
- pylint=2.14.3
- ctapipe=0.15.1.dev166+gf26107f
- colour=0.1.5
- flake8=4.0.1
- pip:
  - pyirf==0.7.0
  - pytimedinput==2.0.1
  - runipy==0.1.5
  - setuptools-scm==6.4.2
  - git+https://github.com/cta-observatory/ctapipe@f26107fe

```

## 4 Additional Plots and Tables

### 4.1 Metrics

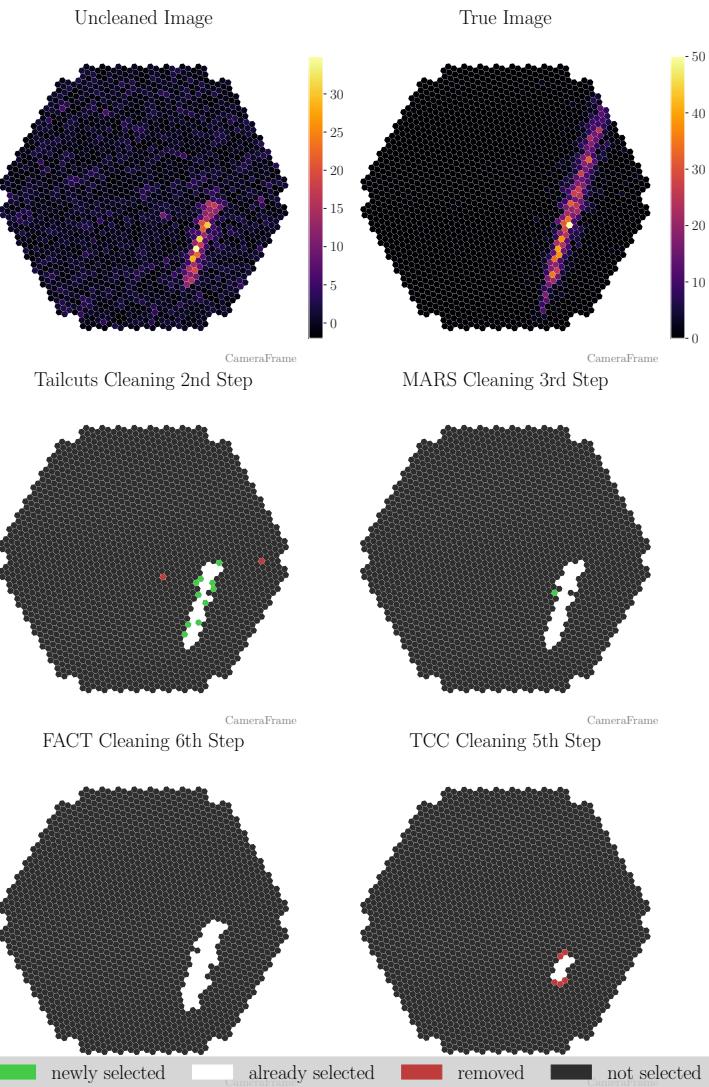
Table 2 below shows the metrics used in this work as well as their formulas. Also shown is a confusion matrix for a binary (positive/negative) classification problem.

**Table 2:** The metrics used for this works analysis as well as their calculation. Right: Confusion matrix for a binary (positive/negative) classification.

Metric	Calculation	Prediction			
		positive	negative		
Label	pos.	True Positive (tp)	False Negative (fn)	neg.	True Negative (tn)
True Positive Rate (TPR)	$\frac{tp}{tp + fn}$				
True Negative Rate (TNR)	$\frac{tn}{tn + fp}$				
False Negative Rate (FNR)	$\frac{fn}{fn + tp}$				
Accuracy (ACC)	$\frac{tp + tn}{tp + fp + tn + fn}$				
Balanced Accuracy (BA)	$\frac{tpr + tnr}{2}$				

## 4.2 A comparison of the last steps of the cleaning algorithms

Figure 1 below shows the last steps of the cleaning algorithms Tailcuts, MARS, FACT and TCC for the default parameters set in `ctapipe`.



**Figure 1:** The last steps of the cleaning algorithms Tailcuts, MARS, FACT and TCC for the default parameters set in `ctapipe`. Also shown are the uncleaned image and the true image for better comparison.

### 4.3 Resulting Settings for Every Cleaning Algorithm

Table 3 below shows all settings determined for every cleaning algorithm. Additionally, the angular resolution and the efficiency are shown to help with selecting the best settings for a user's needs.

**Table 3:** All resulting settings for all four cleaning algorithm. Listed are the core threshold  $Q_c$ , the boundary threshold  $Q_b$ , the minimum number of neighbors and, where applicable, the time limit  $t$  as well as the core and boundary time limits  $t_c$  and  $t_b$ . Besides this, the angular resolution and the efficiency are also shown.

Cleaning Algorithm	ID	$Q_c$ / p.e.	$Q_b$ / p.e.	Min. Neigh.	$t$ / ns	$t_c$ / ns	$t_b$ / ns	Eff	Mean Ang. Res / °
Tailcuts	118	9.500	4.750	5				0.121	0.358
	51	9.500	6.333	4				0.177	0.308
	140	9.500	6.333	3				0.242	0.334
	141	9.500	6.333	2				0.296	0.346
	81	9.500	4.750	2				0.307	0.357
	47	6.200	4.650	2				0.365	0.395
	101	6.700	1.675	1				0.401	0.432
MARS	54	9.500	7.125	5				0.122	0.291
	59	9.500	6.333	4				0.179	0.282
	76	9.500	6.333	3				0.244	0.332
	60	9.500	7.125	2				0.297	0.356
	135	9.500	6.333	2				0.306	0.343
	15	6.700	4.467	1				0.380	0.395
	13	5.500	1.833	2				0.410	0.424
FACT	553	9.500	6.333	5	2.0			0.007	0.244
	271	9.500	4.750	4	4.0			0.051	0.428
	767	9.500	6.333	3	5.0			0.141	0.415
	722	9.500	3.167	3	5.0			0.178	0.367
	865	9.500	7.125	2	2.0			0.250	0.403
	691	9.500	7.125	2	4.0			0.254	0.366
	608	9.500	4.750	1	6.0			0.340	0.365
	503	6.700	3.350	1	12.0			0.388	0.404
	974	4.200	1.400	2	6.0			0.405	0.420
TCC	11	9.500	6.333	5		15.0	15.0	0.070	0.413
	367	9.500	7.125	4		20.0	12.0	0.121	0.396
	399	9.500	6.333	3		18.0	12.0	0.176	0.340
	582	9.500	7.125	2		20.0	12.0	0.248	0.388
	171	5.700	4.275	3		18.0	4.5	0.263	0.386
	1495	4.200	2.800	3		15.0	15.0	0.311	0.383
	807	6.700	4.467	1		15.0	12.0	0.375	0.390
	1954	6.700	1.675	1		15.0	9.0	0.401	0.427