**Введение**

Программное обеспечение, создаваемое в ЛЯП ОИЯИ, предназначено для анализа Широких атмосферных ливней (ШАЛ) из данных Атмосферных черенковских телескопов (АЧТ). Анализ происходит посредством вычисления для каждого зарегистрированного ШАЛ т.н. параметров Хилласа - набора параметров, описывающих изображение ШАЛ в камере, которая расположена в фокусе зеркала АЧТ.

Камера АЧТ состоит из матрицы фотоэлектронных умножителей (ФЭУ) и readout-системы. Основным элементом readout-системы является плата MAROC-3 с 64 входными каналами. Так как для расширения динамического диапазона данные с каждого ФЭУ приходят на два канала - анодный и динодный - то каждый MAROC-3 может получать данные не более, чем с 32 ФЭУ. Поэтому для удобства все ФЭУ разбиты на кластеры по 28. Таким образом 56 каналов MAROC-3 - с 4 по 59 - связаны с выходными каналами ФЭУ, а каналы с номерами 0-3 и 60-63 не используются.

Детектируемый камерой ШАЛ представляет собой область из сработавших ФЭУ камеры, и данное изображение ливня, полученное в камере АЧТ - считается одним событием. Данное изображение аппроксимируется эллипсом, и в итоге рассчитываются следующие параметры Хилласа:

* Size: суммарная амплитуда всех сработавших ФЭУ в фотоэлектронах;
* Length: длина большой полуоси;
* Width: длина малой полуоси;
* Distance: расстояние от центра масс изображения до центра камеры;
* Alpha: угол между большой полуосью эллипса и отрезком, соединяющим центр камеры с центром масс изображения;

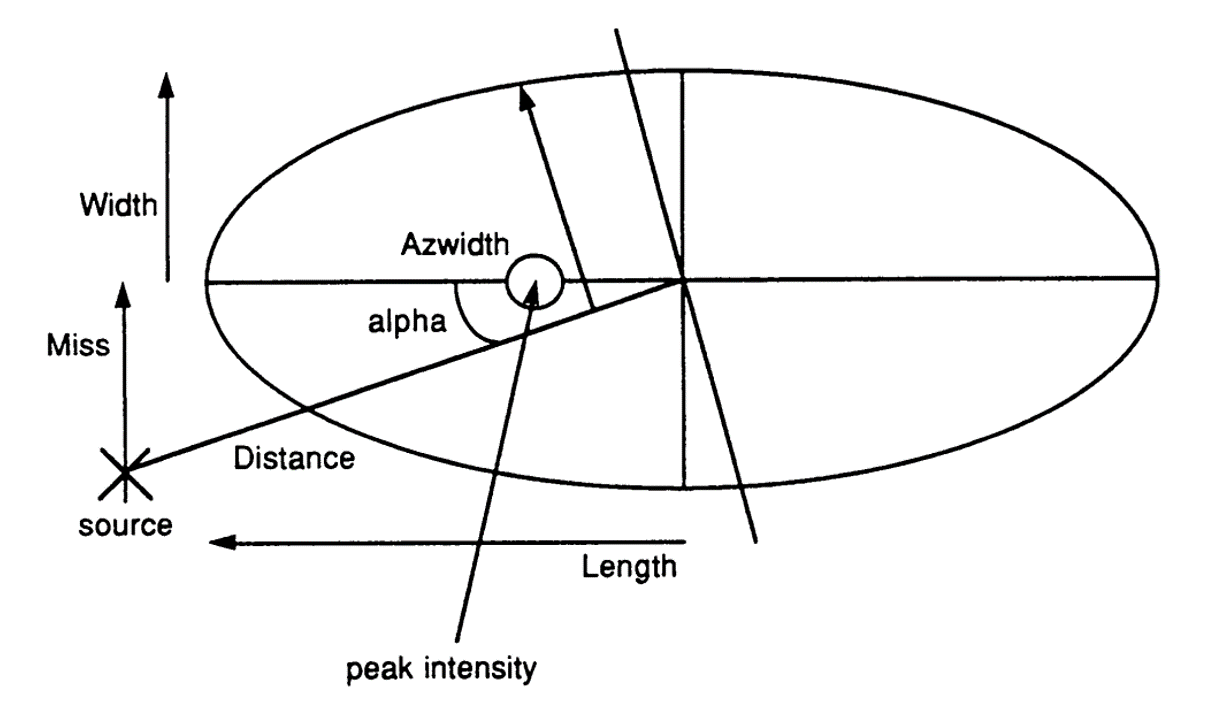


Рис. 1. Схематическое изображение параметров Хилласа. Геометрический центр эллипса на рисунке соответствует центру масс изображения ШАЛ.

Помимо расчета параметров Хилласа для событий, зафиксированных АЧТ, данное ПО также позволяет строить их распределения. Также важной опцией является возможность визуализировать статистику работы камеры АЧТ, т.е. строить наложенные на плоскость камеры средние частоты срабатывания ФЭУ, средние амплитуды за выбранный период времени и проч. Это помогает впоследствии визуально оценивать работу различных ФЭУ, например, находить шумящие ФЭУ и маскировать их и.т.д. На Рис. 2 приведен пример подобной картинки - статистика частоты срабатываний всех ФЭУ камеры, наложенная на картинку плоскости камеры, за первые две минуты первого RUN-а за 28.10.17.

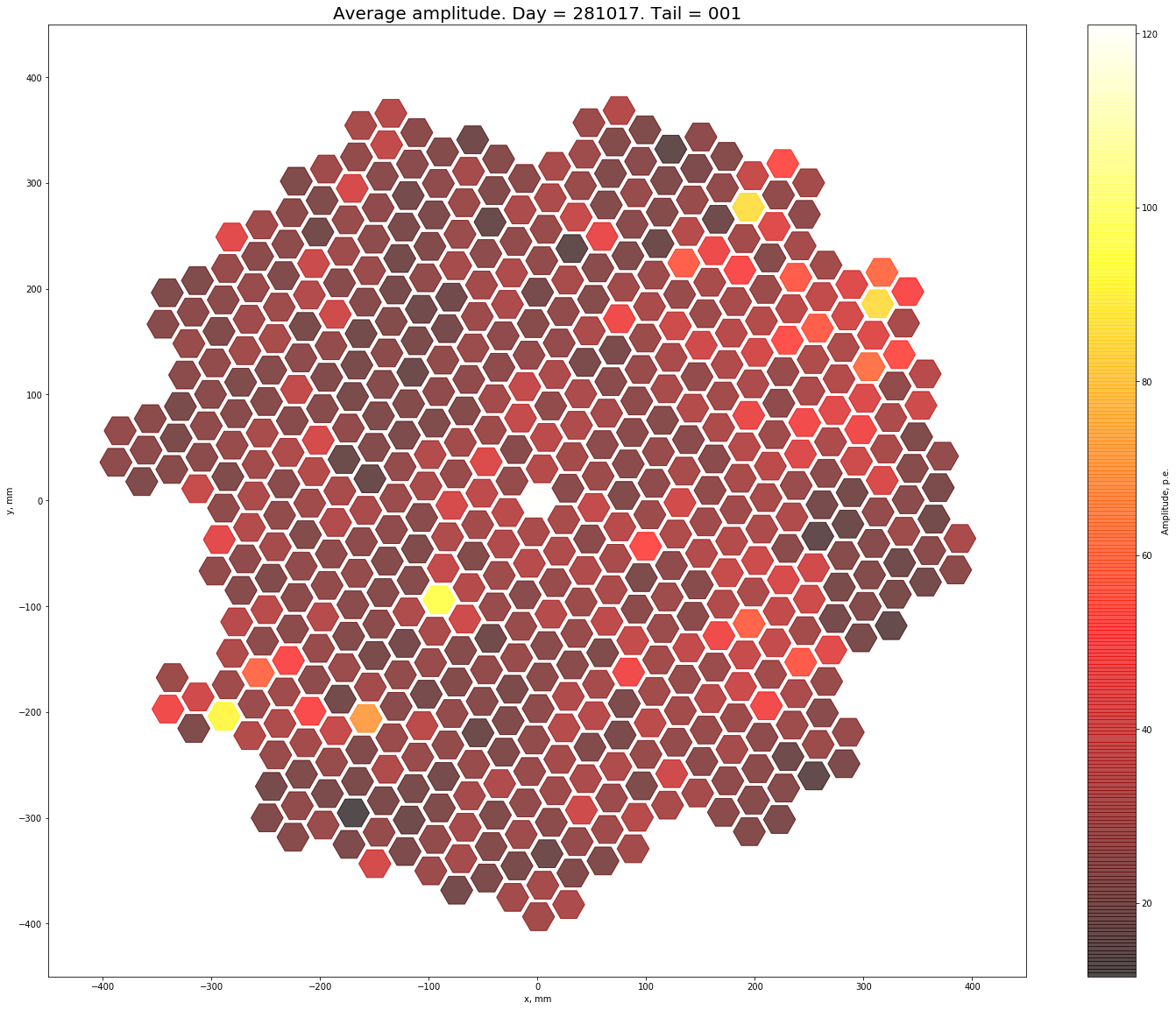


Рис. 2. Распределение средних амплитуд пикселей по всей камере, для первой двухминутной порции RUN-a 281017. АЧТ направлен на источник.

Каждый акт набора данных составляет т.н. RUN. Данные из одного RUN-а записываются без остановки и на жестком диске расположены в директории, именованную определенным образом. К примеру, первый по счету RUN, проведенный 14.06.2021 года, имеет имя 140621, второй по счету - 140621.01 и проч.

Данные каждого кластера записываются в виде бинарных файлов в отдельную директорию, именованную номером кластера, начиная с первого - BSM01, BSM02 … BSM22. Каждый две минуты данные начинают записываться в новый файл. Каждому файлу присваивается имя вида 140621021.005, где первые 6 цифр соответствуют имени RUN-а, следующие 3 цифры - номеру кластера, а 3 цифры после точки - номеру двухминутной порции. Внутри ПО номера двухминутной порции именуются переменной Tail - от 001 до последней в данном RUN-е.

Входными данными данного ПО являются сырые данные, полученные с АЦП АЧТ. Выходными данными - параметры Хилласа для всех событий ШАЛ, содержащихся в сырых данных. ПО создано для работы в операционных системах на базе Linux. Тестирование ПО осуществлялось в ОС Ubuntu 18.04 LTS. Требование к системе: установленный пакет python 3 (или выше).

ПО состоит из трех подпрограмм: Manticore, Scorpicore и модуля конвертации форматов. Отдельные модули созданы для реализации следующих задач:

1. Manticore - декодирование сырых бинарных файлов с АЧТ, вычитание пьедесталов (фоновый сигнал ФЭУ) из амплитуд, перевод кодов АЦП в фотоэлектроны, формирование и вывод финальных файлов с амплитудами в фотоэлектронах отдельно для каждого события. Далее результат работы Manticore будет называться предобработкой, а ее выходные файлы - предобработанными.
2. Scorpicore - анализ предобработанных файлов. Вывод распределений параметров Хилласа, а также визуализация статистики работы камеры.
3. Модуль конвертации форматов - сравнение предобработанных файлов с файлами, предобработанными в НИИЯФ МГУ (сравниваются амплитуды в каждом событии). Конвертация предобработанных файлов в формат, аналогичный формату файлов НИИЯФ МГУ.

Далее - подробно о каждом модуле.

**Manticore**

Manticore - модуль, написанный на языке программирования python, предназначенный для чтения и предобработки сырых данных, набранных с АЧТ.

Интерфейс Manticore реализован с помощью файла input\_card.conf, в которую пользователь предварительно вносит желаемые параметры обработки.

На Рис. 3 приведен скриншот input\_card.conf для модуля Manticore.

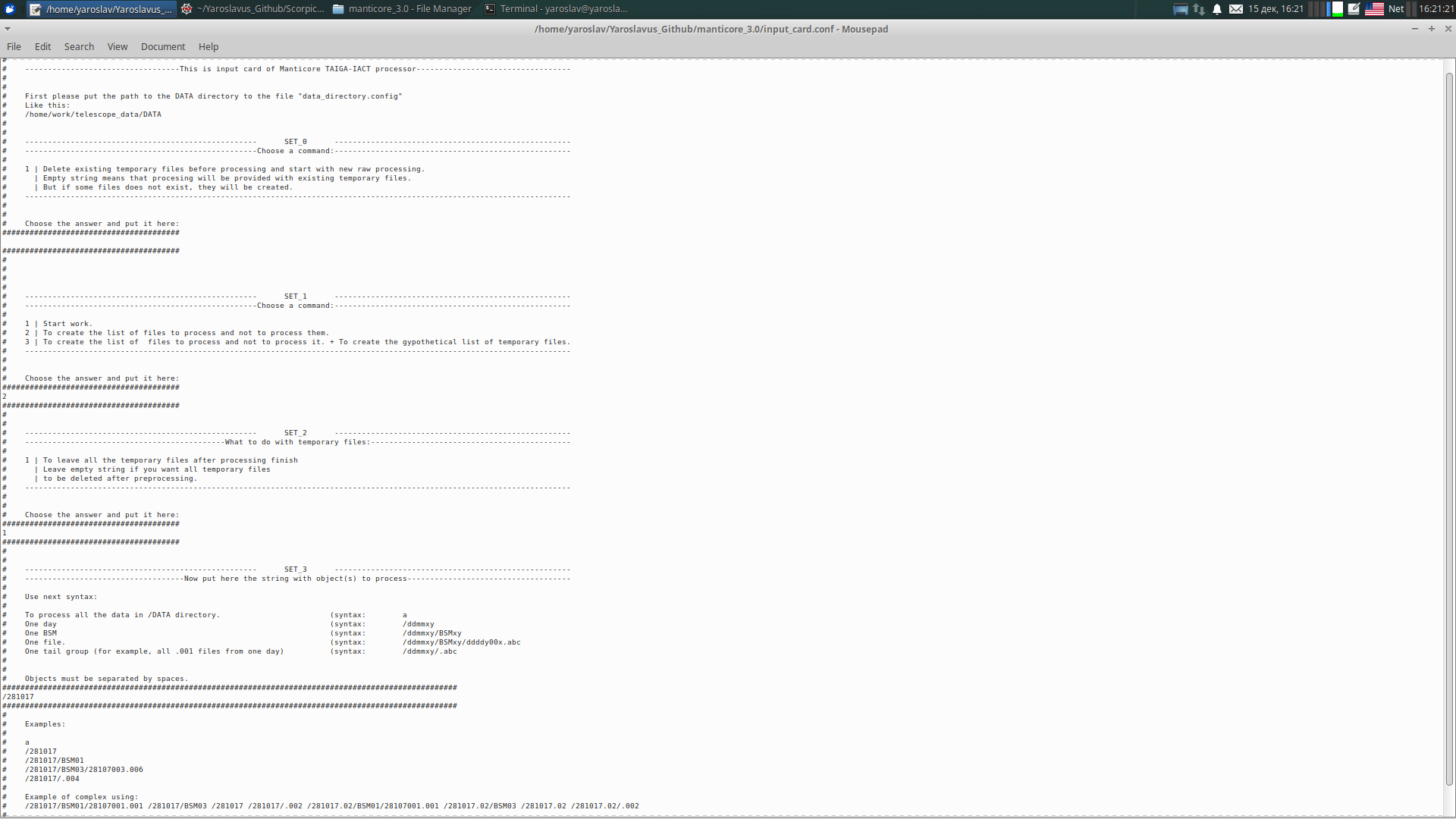


Рис. 3. Скриншот input\_card для Manticore. Параметры SET\_0 и SET\_1 отвечают за удаление (либо да, либо нет) временных файлов, которые генерирует ПО. Параметр SET\_2 определяет режим работы предобработки. Параметр SET\_3 определяет какие файлы будут предобработаны. Например, а - все доступные файлы для предобработки; /281017 - данные всего Run-a от 28.10.2017; /281017/.004 - только 4-я двухминутка Run-a от 28.10.2017;

input\_card.conf принимает следующие параметры: режим работы сборщика мусора, режим предобработки и набор данных, подлежащий предобработке.

1. Режим работы сборщика мусора.

Опции:

Удалять или оставлять после окончания предобработки временные файлы, созданные в ее процессе (для тестирования промежуточных этапов предобработки). Также данный модуль может запускаться и перед началом предобработки, если известно, что после предыдущей предобработки в рабочей директории остались временные файлы.

1.1. Да.

1.2. Нет.

2. Режим предобработки.

Опции:

2.1. Полная предобработка.

2.2. Создание списка файлов, подлежащих предобработке.

3. Набора данных, подлежащий предобработке:

3.1. Все доступные файлы для предобработки.

3.2. Конкретный файл.

3.3. Данные конкретного кластера за конкретный RUN.

3.4. Один двухминутный блок данных из одного RUN-а.

Последняя опция доступна потому, что данные записаны последовательными порциями по 2 минуты (см. Введение). Также она позволяет обрабатывать не все данные за заданный период, а при необходимости игнорировать порции с заведомо плохими данными, сократив при этом время предобработки.

Подробный синтаксис записи параметров предобработки и входного набора данных содержится в самой input\_card.conf.

На Рис. 4 представлена схема работы ПО

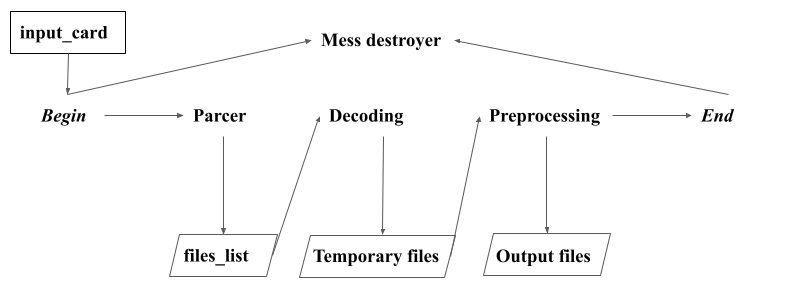


Рис. 4. Блок схема ПО Manticora. Блоки Begin и End - логические точки старта и остановки работы основного алгоритма модуля. Основной алгоритм перед стартом опционно включает сборщик мусора (Mess destroyer), принимает входные параметры из input\_card.conf и передает их в Parser. Дальше по схеме. В точке End основной алгоритм завершается, снова опционно запускает сборщик мусора (Mess destroyer) и выдает предобработанные файлы.

Manticora состоит из 3-х библиотек:

* Parser
* Decoding
* Preprocessing

Кратко о каждом из них:

* **Parser** (от англ. “Parse” - структурный анализ) принимает набор данных, подлежащих предобработке, из input\_card.conf. Так как сырые данные структурированы специфическим образом в большом количестве директорий с различным уровнем вложений, парсер на выходе формирует одиночный текстовый файл files\_list.txt с упорядоченным списком конкретных одиночных сырых файлов для предобработки.
* **Decoding** выполняет операцию декодирования с каждым файлом из files\_list.txt по порядку. Процесс декодирования одного файла состоит в следующем:

1. Расшифровка бинарного файла с амплитудами.
2. Расшифровка бинарного файла со статическими пьедесталами. Статические пьедесталы измеряются единожды за ночь и не меняются для всех двухминутных порций.
3. Формирование выходного бинарного файла с амплитудами, из которых вычтены статические пьедесталы.
4. Формирование бинарного файла с динамическими пьедесталами. Динамические пьедесталы рассчитываются для каждого предобрабатываемого файла отдельно по всем нетриггерным событиям.
5. Формирование выходного бинарного файла с амплитудами, из которых вычтены динамические пьедесталы.
6. Расчет стандартных отклонений значений пьедестала для каждого канала для данного файла (у каждого ФЭУ два канала - динодный и анодный).
7. Расчет стандартного отклонения от стандартного отклонения значений пьедесталов - для выявления каналов, значения которых лежат внутри доверительного интервала. В результате каждому каналу присваивается т.н. “игнор-статус”. Равен 0, если пьедестал канала лежит вне доверительного интервала, и 1 - если внутри него.

В результате для каждого файла генерируются временные файлы с расширениями:

* .spd бинарный файл содержащий значения статических пьедесталов
* .sig бинарный файл содержащий игнор-статусы статических пьедесталов
* .ssg бинарный файл содержащий значения стандартных отклонений статических пьедесталов
* .dpd бинарный файл содержащий значения динамических пьедесталов
* .dig бинарный файл содержащий игнор-статусы динамических пьедесталов
* .dsg бинарный файл содержащий значения стандартные отклонения динамических пьедесталов
* .amp бинарный файл, который содержит фактические значения амплитуд каналов в кодах
* .asp бинарный файл, который содержит значения амплитуд каналов в кодах АЦП, из которых вычтены статические пьедесталы
* .adp бинарный файл, который содержит значения амплитуд каналов в кодах АЦП, из которых вычтены динамические пьедесталы
* .hdr бинарный файл, который содержит номера событий и время их регистрации

В процессе того, как Decoding обрабатывает файлы из files\_list.txt, генерируется файл total\_dict\_of\_days.txt, который содержит информацию о минимальном и максимальном номере события для каждой двухминутной порции. Это необходимо, так как данный файл, в дальнейшем, используется функциями из библиотеки Preprocessing, что позволяет сэкономить время и место на диске, в процессе работы с ней.

* **Preprocessing** объединяет временные файлы, сформированные модулем decoding, в выходной файл, который содержит последовательно информацию о каждом событии (расширение: “.out”). Соответственно, таких файлов будет 3 для каждого tail - двухминутной порции: с чистыми амплитудами, с вычтенными статическими пьедесталами, с вычтенными динамическими пьедесталами.

Сам процесс происходит следующим образом:

1. Для конкретного tail - идентификатора двухминутки (например, 006) - из файла total\_dict\_of\_days.txt считываются номера максимального и минимального номеров событий (min\_number, max\_number), вне зависимости, в каком кластере этот номер зарегистрирован.
2. Создается трехмерная матрица размерности (max\_number - min\_number)\*22\*32, где 22 - количество кластеров в камере, 32 - количество амплитуд, считываемого с одного кластера (не подключенные каналы также считываются).
3. Данная матрица заполняется данными событий, где каждая ячейка содержит амплитуду, номер канала, на основе которого она рассчитана (анодный - 0, динодный - 1).
4. Матрица записывается в соответствующий файл с следующим именем, например для двухминутки 006: 006\_static.out, 006\_dynamic.out или 006\_clean.out.

На Рис. 5-7 представлен пример такого файла (001\_clean.out), где 1 - это номер события, 2 - номер двухминутки, в которой это событие было зарегистрировано, 3 - идентификатор RUN-а и 4 - это 22 строки, каждая из которых соответствует конкретному кластеру (нумерация начинается с нуля). Непустые строки - содержат информацию, разделенную табуляцией, о сработавших кластерах: первое слово в строке - это номер кластера, следующее - время его срабатывания, далее - последовательная информация об амплитуде каждого ФЭУ в кодах АЦП, где первое значение - это амплитуда , а второе идентификатор канала с которого амплитуда была снята (0 - анод, 1 - динод).

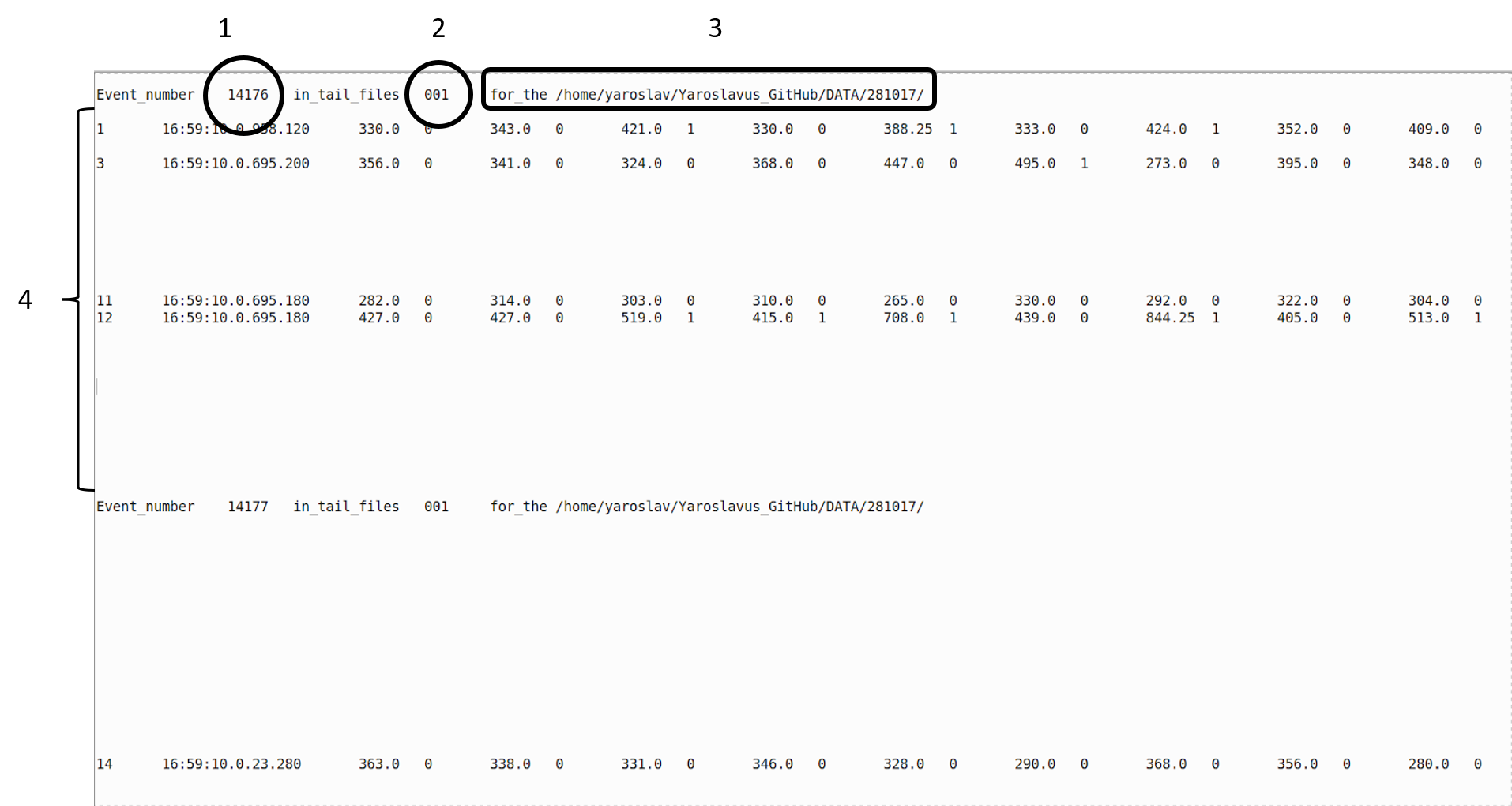


Рис. 5. Пример выходного файла модуля preprocessing ПО manticore. Вывод всех событий с амплитудами за первую двухминутную порцию данных (tail = “001”)



Рис. 6. Пример выходного файла модуля preprocessing ПО manticore. Вывод всех событий с амплитудами, из которых вычтены статические пьедесталы, за первую двухминутную порцию данных (tail = “001”)



Рис. 7. Пример выходного файла модуля preprocessing ПО manticore. Вывод всех событий с сырыми амплитудами, из которых вычтены динамические пьедесталы, за первую двухминутную порцию данных (tail = “001”)

На Рис. 8-10. показан пример визуализации события 14176 Run-a от 28.10.2017. На Рис. 8 показана визуализация сырых амплитуд - без вычитания пьедесталов. На Рис. 9 и 10 визуализировано то же самое событие, но с вычтенными статическими и динамическими пьедесталами соответственно.

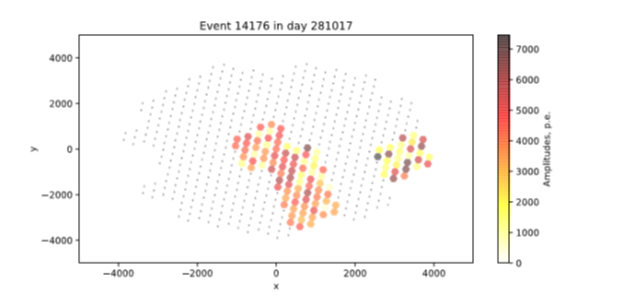


Рис. 8. Сырые амплитуды ШАЛ в камере.

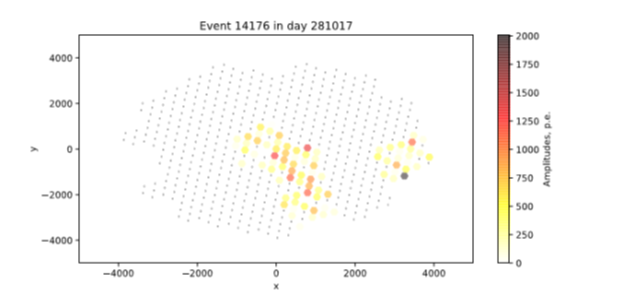


Рис. 9. Амплитуды ШАЛ с вычтенными статическими пьедесталами.

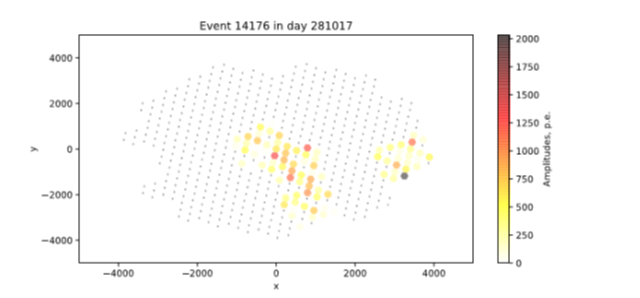


Рис. 10. Амплитуды ШАЛ с вычтенными динамическими пьедесталами.

Также manticore формирует выходной файл со статистикой совпадений. На Рис. 11. представлен скриншот такого файла.

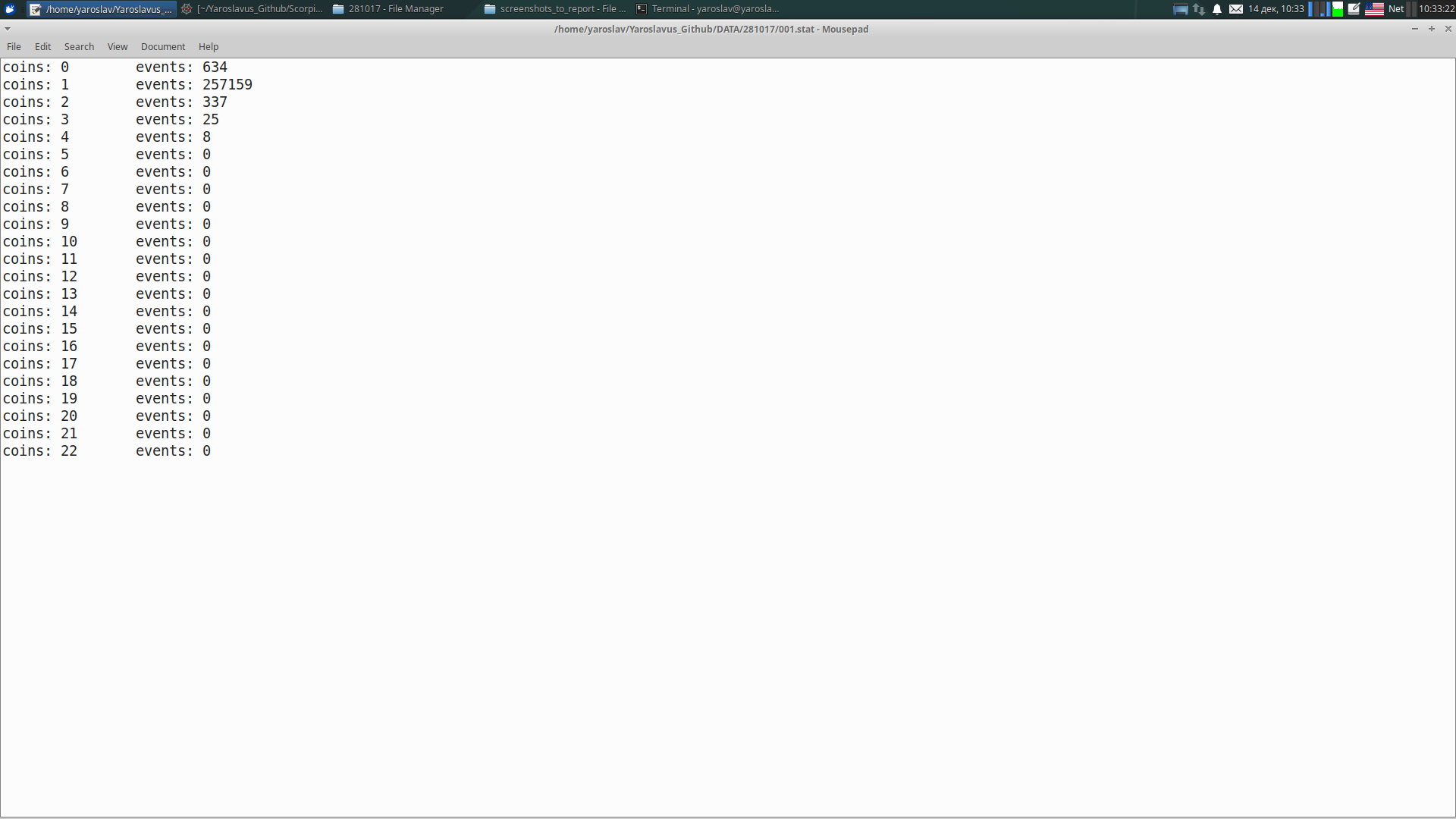


Рис. 11. Пример вывода статистики всех событий за первую двухминутную порцию сырых данных (идентификатор “001”). Первый столбец - число сработавших кластеров в событии, а второй - количество таких событий.

**Scorpicore**

Scorpicore - модуль, написанный на языке программирования python, предназначенный для анализа предобработанных, сформированных Manticore.

Scorpicore может работать в двух режимах:

1. Рассчитать параметры Хилласа конкретного события по заданному номеру.
2. Рассчитать параметры Хилласа для всех событий заданного RUN-а.

Интерфейс Scorpicore реализован с помощью файла input\_card.conf, в который предварительно вносятся желаемые параметры анализа: номер события или флаг, который характеризует необходимость обработать весь RUN.

В процессе работы Scorpicore оперирует объектами класса Pmt, реализованного в нем отдельным модулем. Начальная задача Scorpicore - инициализировать для каждого реального ФЭУ один объект класса Pmt, содержащий информацию о его координатах (например, центральный ФЭУ имеет координаты (0,0)) и информацию о расчетных коэффициентах (код АЦП/фотоэлектрон, усиление и проч.). Так как реальные ФЭУ пронумерованы двумя числами - номером кластера и номером ФЭУ, а для навигации по объектам класса Pmt удобна сквозная нумерация, каждому объекту присваивается персональный номер (“global\_number”) таким образом, что его легко “расшифровать обратно”, получив номер кластера и номер ФЭУ.

Для каждого события Scorpicore производит инициализацию всех ФЭУ камеры, как объектов класса Pmt, а затем пропускает все данные объекты через цепь операций “Cleaning -> Hillas”. Затем происходит запись строки с рассчитанными параметрами Хилласа данного события в выходной файл. Затем Scorpicore переходит к следующему событию, снова пишет в файл и т.д.

Scorpicore состоит из 3-х модулей:

* Cleaning
* Hillas
* Plotter

Кратко о каждом из них:

* **Cleaning** предназначен для очищения (клининга) изображения ШАЛ от сигналов шумящих ФЭУ.. Шумящие ФЭУ характеризуется тем, что их амплитуды в конкретном событии, засоряют изображение ШАЛ. Например, это может быть связано с тем, что в поле зрения телескопа попала звезда; был зарегистрирован свет от фар проезжающей машины и.т.п. Другими словами в камере помимо изображения ШАЛ, присутствуют изображения сторонних светящихся объектов. Соответственно необходимо реализовать некие алгоритмы клининга, которые будут их находить и удалять. В данном модуле реализованы 4 функции, каждая из которых соответствует определенной методике клининга:
  + Пороговая. При использовании данной методики амплитуды всех ФЭУ, которые ниже некоего, заранее заданного порога, зануляются. Алгоритмическая сложность такого клининга для элементов.

Следующие две методики работают с понятием соседей - соседних ФЭУ. Каждый ФЭУ, за исключением расположенных на краю камеры имеет 6 соседей. Соседи данного ФЭУ могут принадлежать как тому же кластеру, что и он, так и другому.

* + По сумме амплитуд соседних ФЭУ. При использовании данной методики зануляются амплитуды тех ФЭУ, у которых суммарная амплитуда соседних ФЭУ не превышает некоторого, заранее заданного порога. Алгоритмическая сложность такого клининга для элементов.
  + По количеству соседних ФЭУ. При использовании данной методики зануляются амплитуды тех ФЭУ, у которых суммарное количество соседних ФЭУ с амплитудой, ниже некоторого порога, не более, чем некоторое, заранее заданное число. Алгоритмическая сложность такого клининга для элементов.

На Рис. 12 показан пример изображения ШАЛ до процедуры клининга. На Рис. 13-14 показано изображение того же самого ШАЛ, которое подверглось различным процедурам клининга - по сумме амплитуд соседних ФЭУ и по количеству соседних ФЭУ.

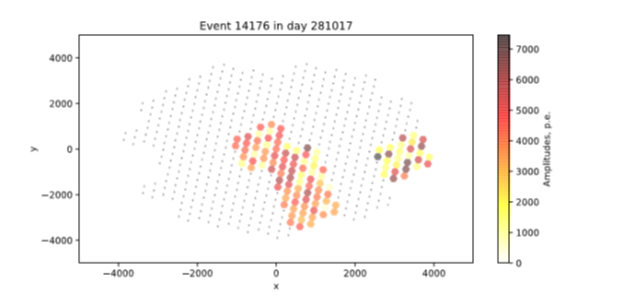


Рис. 12. Изображение ШАЛ до процедуры клининга.

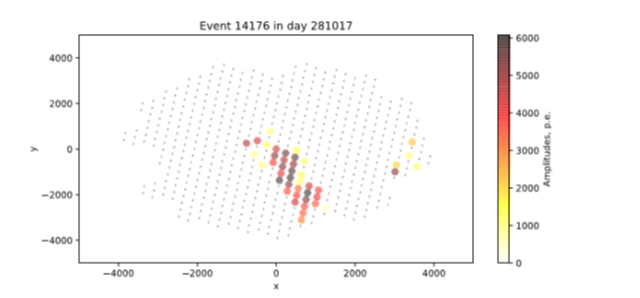


Рис. 13. Изображение ШАЛ после клининга по сумме амплитуд соседних ФЭУ.

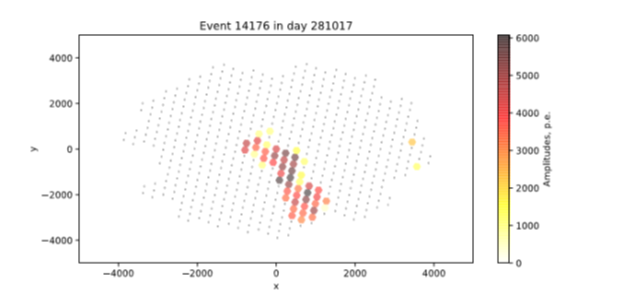


Рис. 14 Изображение ШАЛ после клининга по количеству соседних ФЭУ.

Отдельным образом следует упомянуть о функции, которая осуществляет т.н. “топологический клининг”.

На Рис. 13-14 видно, что даже применение вышеупомянутых типов клининга не гарантирует полного выделения события. Это видно по оставшимся в правом краю камеры нескольким не очищенным ФЭУ, которые находятся вне основного “пятна” события. Значит, необходим такой тип клининга, который не только работает с пороговыми значениями, но и оперирует с пятнами, как отдельными друг от друга объектами. Благодаря этому можно не только анализировать каждое пятно в отдельности, но и по необходимости “выключать” в событии все пятна, которые не представляют интереса. Для каждого события необходимо оставить только одно пятно, а именно - содержащее в себе самое большое количество ФЭУ в сравнении с другими пятнами.

Основная идея клининга такова:

1. Вначале необходимо среди сработавших ФЭУ выделить пятна - замкнутые области из ФЭУ, амплитуда которых больше, чем некоторое заданное наперед пороговое значение в фотоэлектронах. По умолчанию это значение - ноль, однако можно применить обычный пороговый клининг, повысив это значение. К примеру, при обработке изображений ШАЛ в НИИЯФ МГУ часто берется значение в 7 фотоэлектронов.
2. Далее необходимо разделить пятна друг от друга. Итогом этой операции является то, что каждое пятно будет существовать в программе, как отдельный объект, содержащий всю информацию о содержащихся в нем ФЭУ и только о них. Также каждому пятну присваивается свой уникальный номер.
3. На финальном шаге, выделяется только одно пятно, которое с наибольшей вероятностью соответствует зарегистрированному ШАЛ. Остальные просто стираются: зануляется амплитуды всех ФЭУ, которые им принадлежат.

Преимущества данного типа клининга состоят в следующем:

* эмпирический опыт использования показывает, что данный клининг даже сам по себе работает эффективнее, чем клининги по количеству соседей и по сумме амплитуд соседей. Он в принципе не оставляет отдельно стоящих шумящих ФЭУ или их островков. Предварительное применение обычного порогового клининга дополнительно улучшает результат;
* В силу глубоко рекурсивной природы реализации этого клининга в библиотеке scorpicore\_cleaning, его алгоритмическая сложность для элементов.
* **Hillas** производит расчет параметров Хилласа для одного события. Опционно может работать с использованием поправок на wobble-режим работы АЧТ.
* **Plotter** - модуль, отвечающий за графическое представление полученных данных. Содержит две функции: первая - для построения гистограмм с параметрами Хилласа, вторая - для отображения событий или различной статистики на модели камеры (см. Рис. 2).

Для демонстрации работы вышеизложенного ПО был взят один из нормировочных RUN-ов - 231119.01 от 23.11.2019 . На рисунках 15-19 приведены распределения параметров Хилласа за взятый период. Гистограммы синего цвета построены на основе данных, полученных с помощью топологического клининга с нижним порогом в 0 фотоэлектронов, т.е. были очищены только те ФЭУ, амплитуды которых оказались ниже, чем соответствующие пьедесталы, а затем выделены отдельные пятна и рассчитаны их параметры Хилласа. Гистограммы красного цвета построены на основе данных, полученных с помощью топологического клининга с нижним порогом в 7 фотоэлектронов, т.е. после зануления по пьедесталам проводилась очистка всех ФЭУ, амплитуды которых ниже, чем 7 фотоэлектронов. А затем выделялись пятна и считались их параметры Хилласа.

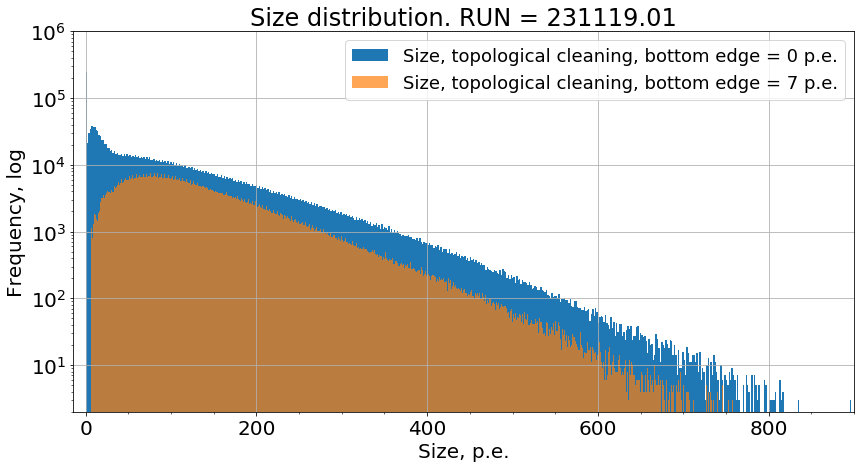


Рис. 15. Распределение параметра Хилласа Size для RUN-a 231119.01. Выделяющийся не острый пик в районе нуля на синей гистограмме соответствует острому пику в нуле на оранжевой гистограмме. Это свидетельствует о том, что многие пятна, прошедшие клининг с порогом в 0 фотоэлектронов, и имевшие при этом ненулевой Size, были очищены после клининга с порогом в 7 фотоэлектронов.

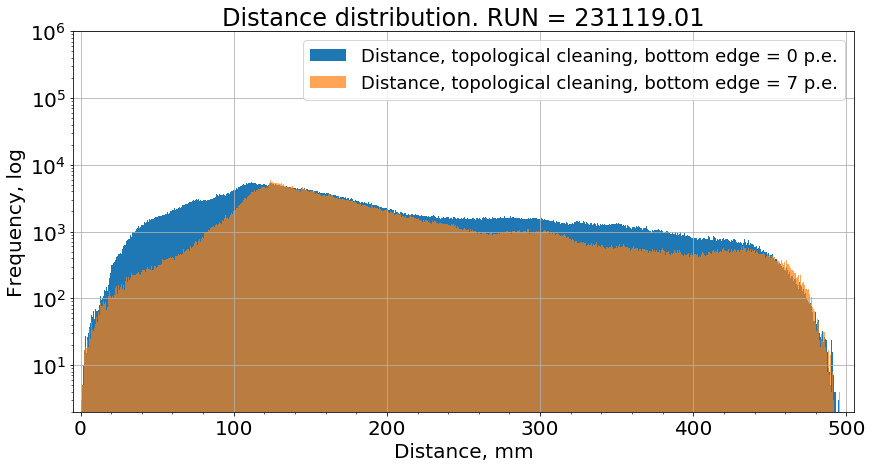


Рис. 16. Распределение параметра Хилласа Distance для RUN-a 231119.01.

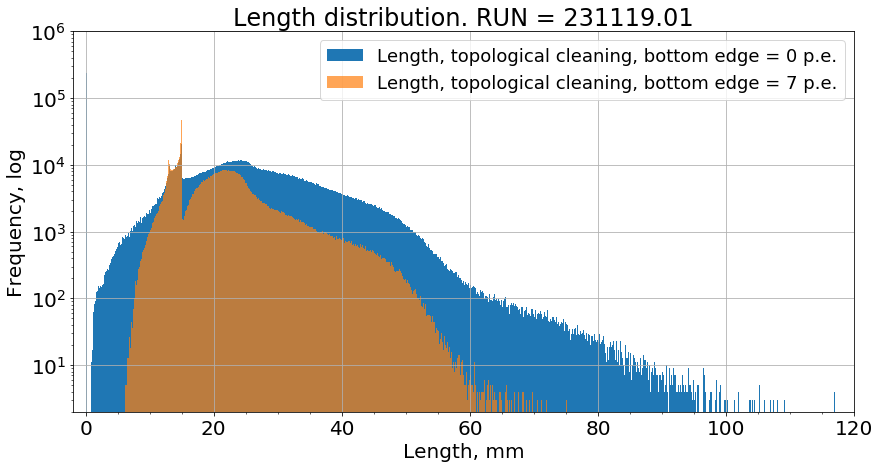


Рис. 17. Распределение параметра Хилласа Length для RUN-a 231119.01. Острый пик в нуле оранжевой гистограммы соответствует событиям, амплитуды которых выше пьедестальных, но не прошли чистку с порогом в 7 фотоэлектронов (см. описание к Рис. 15).

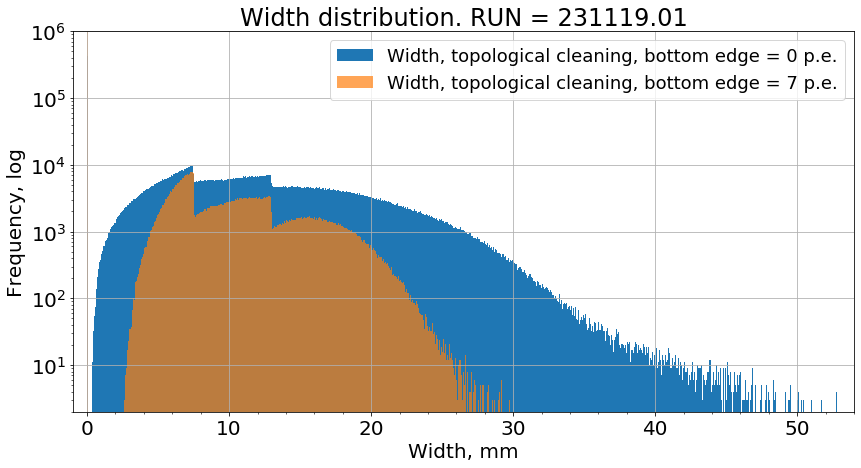


Рис. 18. Распределение параметра Хилласа Width для RUN-a 231119.01. Синяя гистограмма - после топологического клининга с порогом 0 фотоэлектронов, оранжевая - после топологического клининга с порогом 7 фотоэлектронов.

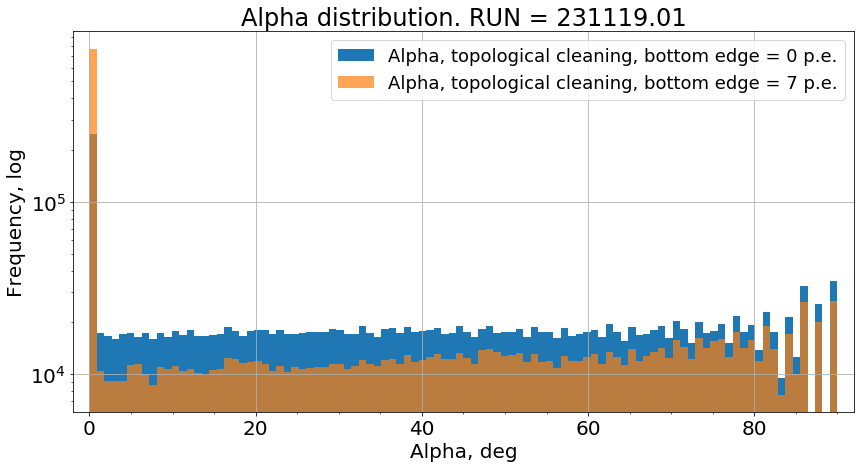


Рис. 19. Распределение параметра Хилласа Alpha для RUN-a 231119.01. Видно, что преобладают события, направленные к центру камеры, а промежутки в бинах в районе 90 градусов говорят о малом количестве таких событий. В остальных областях гистограммы распределение равномерно.

На рисунках 20-22 визуализирована статистика по работе камеры для RUN-a 231119.01. На Рис. 20 - темпы счета, на Рис. 21 - суммарные амплитуды. На обоих графиках видна кластерная структура камеры. Однако пороги (отдельно на каждом кластере) выставляются вручную таким способом, чтобы средние амплитуды были одного порядка по всей камере (Рис. 22).

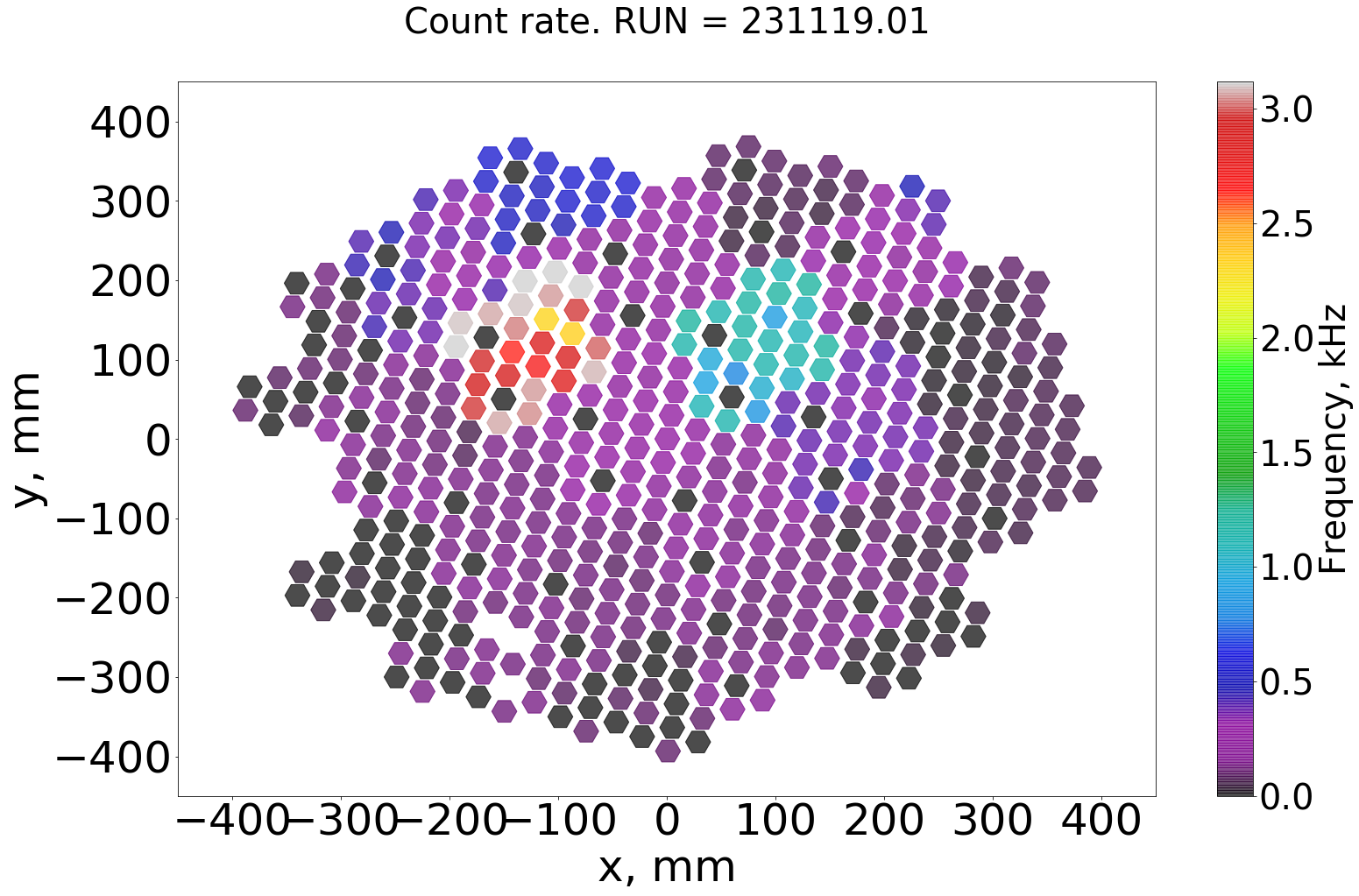


Рис. 20. Распределение количества срабатываний по всей камере в Hz для RUN-a 231119.01. Заметна кластерная структура, что отражает независимость настроек порогов по кластерам.

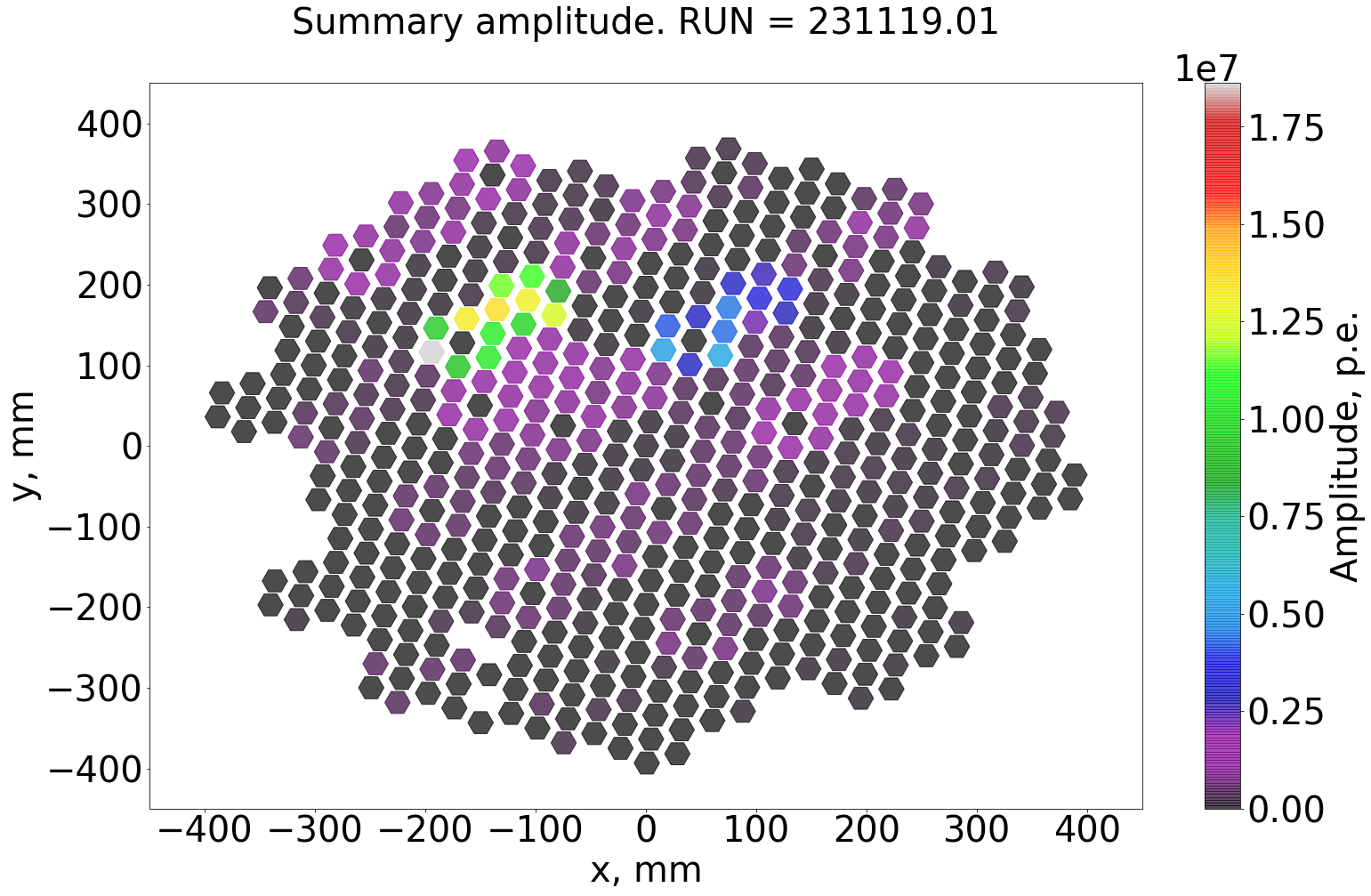


Рис. 21. Распределение суммарных амплитуд пикселей по всей камере, для RUN-a 231119.01.

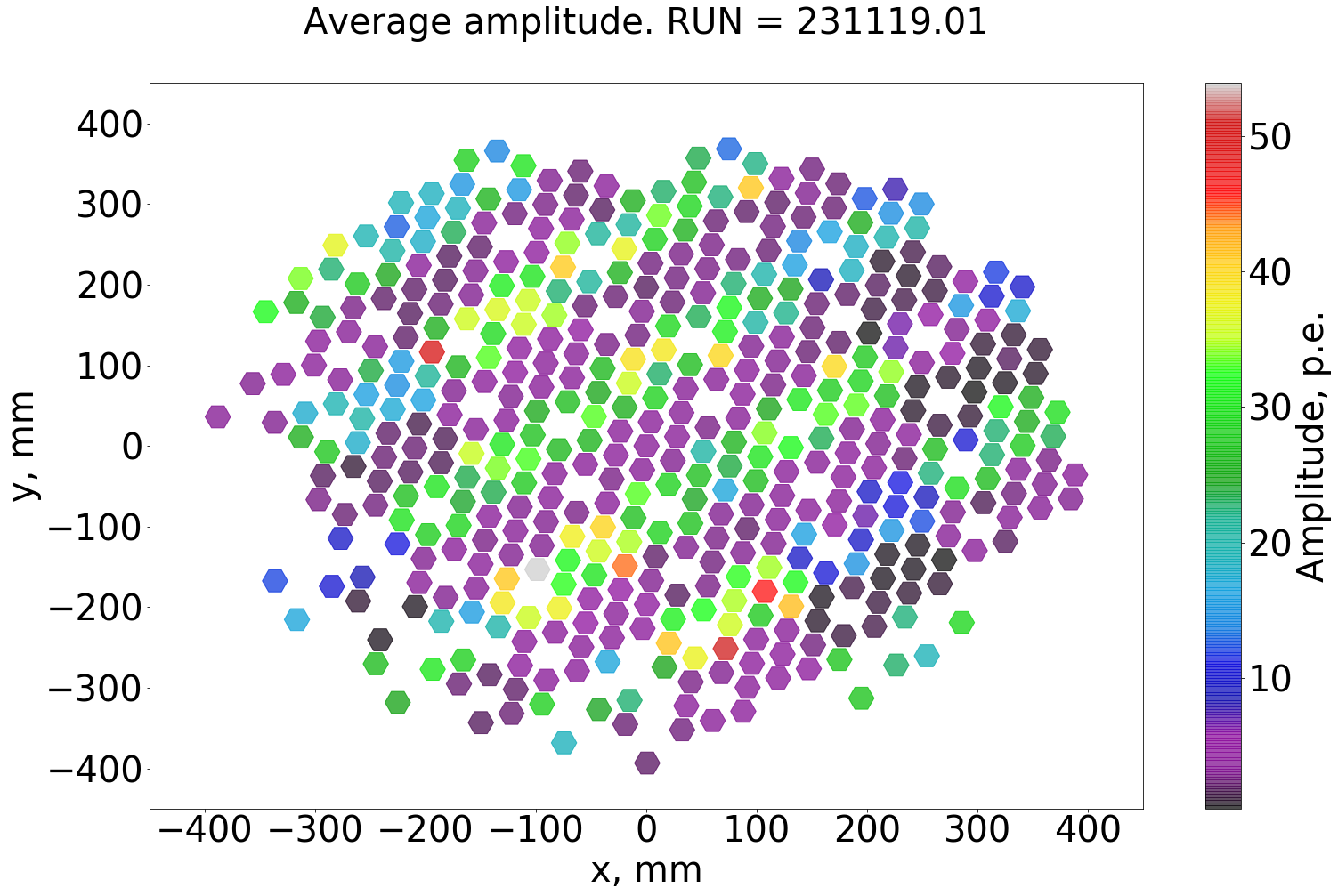


Рис. 22. Распределение средних амплитуд пикселей по всей камере, для RUN-a 231119.01.

**Модуль конвертации форматов**

Данный модуль был создан для того, чтобы сравнивать результат работы manticore с результатами полученными группой НИИЯФ МГУ им. Скобельцына, занимающейся предобработкой сырых данных. Данный модуль конвертирует результат manticore в файлы аналогичные по формату файлам, полученным в НИИЯФ МГУ, с некоторыми различиями в содержании:

* Файлы manticore содержат амплитуды только одного канала для каждого ФЭУ: либо анодного, либо динодного. Если амплитуда анодного канала в кодах АЦП ниже заданного порогового значения (постоянная величина для RUN-a), то для дальнейшей обработки берется она, в противном случае берется амплитуда с динодного канала. Так как файлы МГУ всегда содержат амплитуды ФЭУ с обоих каналов, то конвертированные файлы manticore , при конвертации, искусственно заполняются следующими значениями. Если есть анодная амплитуда, но нет динодной, то первая вносится в файл без изменений, а вторая обозначается как -11 (-11 - произвольно выбранное число). В противоположном случае, когда нет анодной, но есть динодная амплитуда, аноду присваивается амплитуда “пороговое значение + 1”, а динодная амплитуда вносится в файл без изменений.
* В файлы manticore для каждого ФЭУ пишется идентификатор канала (0 - анод, 1 - динод), тогда как файлы МГУ, вместо этих значений, содержат trigger status (0 - канал триггерный, 1 - канал не триггерный),

В силу того, что конвертированные файлы имеют тот же формат, что и файлы МГУ, с учетом вышеупомянутых различий, они могут быть использованы сторонним ПО, которое создано для работы с файлами МГУ.

**Дальнейшая разработка**

Планируется в дальнейшем:

* Структуризация кода, т.е. разработка единого пространства переменных, разработка единого стиля написания кода и единого стиля комментирования, планирование и реорганизация некоторых модулей и их взаимодействий, реализация обработки исключений;
* Подготовка документации к использованию;
* Создание дополнительных модулей для запуска данного ПО на ресурсах, предоставленных Многофункциональным информационно-вычислительным комплексом ОИЯИ;
* Создание модулей, предназначенных на агрегацию данных с двух и более телескопов;
* Внедрение в данное ПО возможностей использования параллельных вычислений.

**Выводы**

На текущий момент реализованы и протестированы все функциональные модули ПО для чтения сырых данных с АЧТ, для их декодирования в физические данные, для расчета параметров Хилласа, а также для построения их распределений. Было произведено сравнение предобработанных данных разрабатываемого ПО с предобработанными данными НИИЯФ МГУ, показавшее их сходимость.