**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Кафедра “фундаментальная информатика и информационные технологии”**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Функциональное программирование»**

**на тему «Анализ космических данных с использованием параллельных вычислений»**

**Вариант – 1**

Студент 23.Б16 Юдинцев С.А.

Преподаватель Киямов Ж.У.

**Санкт-Петербург**

**2024 г.**

**Содержание**

**1** [***Цель работы 3***](#_30j0zll)

**2 Постановка** [***задачи 3***](#_1fob9te)

**3** [***Основные шаги программы***](#_3znysh7) **5**

**4** [***Описание программы***](#_2et92p0) **6**

**5** [***Рекомендации по работе с программой 8***](#_3dy6vkm)

**6** [***Вывод 1***](#_tyjcwt)**2**

1. **Цель работы**

Целью данного проекта является создание программного обеспечения для анализа больших массивов космических данных, полученных с телескопов, с применением технологий параллельных вычислений. Разработанная программа должна обеспечивать высокую скорость обработки изображений, идентификацию астрофизических объектов и сбор статистической информации о них с использованием многопоточной обработки, что существенно сокращает время выполнения расчетов.

**Ключевые задачи:**

1. Внедрение параллельной обработки для повышения производительности анализа.

2. Точное обнаружение и классификация астрофизических объектов на изображениях.

3. Вычисление статистических параметров (яркость, площадь, координаты) для найденных объектов.

4. Обеспечение корректной синхронизации потоков для предотвращения конфликтов данных при параллельном выполнении.

1. **Постановка задачи**

**Входные данные**:

• Набор изображений космических объектов в форматах TIFF, JPEG или PNG.

**Выходные данные**:

• Аннотированные изображения с выделенными объектами.

• Текстовые файлы с информацией об объектах (координаты, яркость, размер, тип).

**Цели программы**:

1. Разделить входные изображения на части для параллельной обработки.

2. Проанализировать каждую часть изображения, выделяя объекты с их характеристиками.

3. Сохранить обработанные изображения и собранные статистические данные.

4. Обеспечить корректное взаимодействие между процессами для повышения производительности.

**Теоретическая часть**

1. **Бинаризация изображения**:

Для выделения объектов используется метод пороговой фильтрации с функцией cv2.threshold(). Пиксели с яркостью выше заданного порога считаются частью объектов.

2. **Фильтрация и сглаживание**:

Применяется фильтр повышения резкости через cv2.filter2D() и размытие методом Гаусса с помощью cv2.GaussianBlur() для устранения шумов.

3. **Сегментация объектов**:

Для нахождения контуров и выделения объектов используется метод cv2.findContours(). Полученные контуры позволяют определить область и координаты объектов.

4. **Классификация объектов**:

Объекты классифицируются на основе площади и яркости по следующим критериям:

• Маленькая область и высокая яркость: **звезда**

• Средняя яркость и маленький размер: **планета**

• Большая яркость и площадь: **галактика** или **квазар**

5. **Параллельная обработка**:

• Входные изображения разбиваются на фрагменты функцией split\_image().

• Для каждой части запускается отдельный процесс с использованием модуля multiprocessing.

• Результаты обработки собираются с использованием очереди Queue.

6. **Визуализация и аннотация**:

Для добавления текста на изображения используется библиотека **PIL**. Аннотированные изображения сохраняются в отдельной папке.

1. **Основные шаги программы**

# 

1. **Загрузка изображений**:

• Пользователь выбирает изображения с помощью диалогового окна (tkinter.filedialog).

2. **Разбиение изображения**:

• Изображение разделяется на несколько частей для параллельной обработки.

3. **Анализ фрагментов**:

• Каждый процесс выполняет:

• Фильтрацию и бинаризацию изображения.

• Выделение контуров объектов.

• Определение характеристик объектов (координаты, яркость, размер, тип).

• Результаты сохраняются в текстовый файл.

4. **Сбор результатов**:

• Части обработанных изображений собираются в одно целое.

• Итоговое изображение сохраняется с аннотациями.

5. **Сохранение выходных данных**:

• Обработанные изображения сохраняются в формате TIFF.

• Текстовые файлы с характеристиками объектов сохраняются в директории результатов.

1. **Описание программы**

# 

1. **Используемые библиотеки:**

• OpenCV (cv2): обработка изображений, бинаризация, нахождение контуров.

• NumPy (numpy): работа с массивами данных.

• PIL (Pillow): добавление текстовых аннотаций на изображения.

• multiprocessing: параллельное выполнение задач для ускорения обработки.

• tkinter: создание графического интерфейса для взаимодействия с пользователем.

• os: работа с файловой системой для сохранения результатов.

1. **Основные функции программы**:

*Таблица 4.1 Описание функций*

| Название функции | Входная переменная | описание |
| --- | --- | --- |
| analysing | image, number, queue, output\_directory | Обрабатывает часть изображения, выделяет объекты, классифицирует их и сохраняет результаты в файл. |
| classified | area, brightness | Классифицирует объекты по площади и яркости (звезда, планета, галактика или квазар). |
| split\_image | image, num\_parts | Разделяет изображение на части для параллельной обработки. |
| parallel\_processing | image\_paths | Управляет параллельной обработкой изображений, запускает процессы и объединяет результаты. |
| select\_images | - | Открывает диалоговое окно для выбора изображений и сохраняет их пути. |
| create\_ui | - | Создает и запускает графический пользовательский интерфейс с кнопками для обработки изображений. |

1. **Переменные программы**:

*Таблица 4.2 Описание переменных*

| Название переменной | Описание |
| --- | --- |
| file\_paths | Список для хранения путей к выбранным изображениям. |
| image\_with\_objects | Изображение с выделенными объектами и аннотациями (контуры и подписи объектов). |
| kernel | Ядро для применения фильтра (усиление контрастности) с помощью cv2.filter2D. |
| gray\_image | Черно-белое изображение, полученное с помощью преобразования в градации серого. |
| blurred\_image | Размытое изображение после применения гауссовского фильтра для уменьшения шума. |
| binary\_image | Бинарное изображение после пороговой обработки (выделение объектов). |
| contours | Список контуров, найденных на бинарном изображении с помощью cv2.findContours. |
| space\_objects | Список словарей с характеристиками обнаруженных объектов (яркость, координаты, размер). |
| num\_parts | Количество частей, на которые делится изображение для параллельной обработки. |
| image\_parts | Список, содержащий части изображений после обработки и их объединения. |
| image\_vstack | Список вертикально объединенных частей изображения. |
| lbl\_title, lbl\_instructions | Метки текста в пользовательском интерфейсе. |
| btn\_select, btn\_process | Кнопки для выбора изображений и запуска обработки в пользовательском интерфейсе. |

### 

### 

### 

### 

### 

### 

1. [**Рекомендации по работе с программой**](#_3dy6vkm)

**Рекомендация пользователю**

### 1. **Формат изображений**

### Программа поддерживает обработку изображений форматов .tif, .jpg и .png. Убедитесь, что изображения находятся в одном из этих форматов.

### 2. **Выбор изображений**

### • Нажмите кнопку **“Выбрать изображения”** и выберите одно или несколько изображений через открывшееся диалоговое окно.

### • Если изображения не были выбраны, программа покажет предупреждение.

### 3. **Запуск обработки**

### • После выбора изображений нажмите **“Начать обработку”**.

### • Дождитесь завершения процесса. По окончании работы появится сообщение о сохранении результатов.

### 4. **Результаты обработки**

### • Все результаты сохраняются в папке image\_result в подпапках, названных по имени обрабатываемого файла.

### • Для каждого изображения создается:

### • **new\_image.tif** – изображение с выделенными объектами и аннотациями.

### • Текстовые файлы с характеристиками объектов (координаты, яркость, размер и тип).

### • Части исходного изображения сохраняются в подпапке image\_crop.

### 5. **Системные требования**

### • Программа оптимизирована для многоядерных процессоров, поэтому для быстрой обработки рекомендуется использовать ПК с несколькими ядрами.

### • Убедитесь, что установлены **Python** версии 3.7 или выше и библиотеки: opencv-python, Pillow, numpy.

### 6. **Графический интерфейс**

### Программа имеет простой интерфейс с двумя кнопками:

### • **Выбрать изображения** – открывает окно выбора файлов.

### • **Начать обработку** – запускает анализ изображений.

**Рекомендация программисту**

**1. Совместимость и зависимости**

**• Проверьте, что все используемые библиотеки (cv2, numpy, Pillow, multiprocessing, tkinter) установлены.**

**• Убедитесь, что путь к шрифту (например, Arial.ttf) корректен для используемой операционной системы.**

**2. Многопроцессорная обработка**

**• Использование функции multiprocessing может вызывать ошибки в системах с ограниченными ресурсами.**

**• Для уменьшения нагрузки можно сократить количество частей изображения в функции split\_image (параметр num\_parts).**

**3. Пути сохранения файлов**

**• Убедитесь, что программа имеет права на создание папок и запись файлов в текущем рабочем каталоге.**

**• Для лучшей конфигурации можно добавить возможность выбора директории сохранения результатов через интерфейс.**

**4. Расширение классификации**

**• В функции classified типы объектов классифицируются по простой логике площади и яркости. Для улучшения можно внедрить машинное обучение или более сложные эвристики.**

**5. UI/UX улучшения**

**• Добавьте прогресс-бар или индикатор выполнения задач для отображения статуса обработки.**

**• Логику обработки можно вывести в фоновый поток, чтобы не замораживался интерфейс.**

**6. Обработка ошибок**

**• Добавьте обработку исключений при чтении файлов изображений (cv2.imread).**

**• Проверьте корректность ввода данных, например, наличие изображений перед началом обработки.**

**7. Оптимизация кода**

**• Разделение изображения на части (split\_image) может быть оптимизировано для более равномерного распределения пикселей.**

**• Добавьте параметры для настройки размера частей изображения или порога бинаризации.**

**8. Платформозависимость**

**• Протестируйте программу на разных операционных системах (Windows, macOS, Linux). Обратите внимание на пути к шрифтам и файлам.**

**• Для кроссплатформенности можно использовать встроенные шрифты вместо абсолютных путей.**

**9. Логирование**

**• Добавьте логирование для отслеживания прогресса и ошибок программы.**

**• Рекомендуется использовать модуль logging вместо print() для записи информации о процессе.**

1. **Вывод**

Разработанная программа эффективно анализирует изображения космических объектов с применением многопроцессорной обработки. Параллельная обработка позволяет значительно сократить время выполнения, а использование библиотек OpenCV и PIL обеспечивает точное выделение объектов и их аннотирование.

**Ключевые результаты**:

1. **Оптимизация обработки**: параллельная обработка фрагментов изображений.

2. **Точность анализа**: классификация объектов на основе их характеристик.

3. **Гибкость программы**: поддержка различных форматов изображений и наглядный графический интерфейс.

Программа может быть использована в астрофизических исследованиях для автоматического анализа и классификации космических объектов.

Материалы:

1. <https://nauchkor.ru/pubs/parallelnye-algoritmy-binarizatsii-izobrazheniy-587d36605f1be77c40d58df4>
2. <https://docs.python.org/3/library/threading.html>
3. <https://pypi.org/project/pillow/>