212-Кончугаров-Методичка

Методичка.

Оглавление:

1. Введение

Данная программа предназначена для обработки и визуализации двумерных данных с использованием различных алгоритмов обработки изображений, кластеризации и построения путей. Программа может работать как с синтетически сгенерированными данными (гауссовы распределения), так и с реальными изображениями в формате BMP.

Программа предназначена для комплексной обработки двумерных данных с возможностью:
- Генерации искусственных ландшафтов с помощью гауссовых распределений
- Анализа и обработки реальных изображений в формате BMP
- Выделения значимых компонент на изображениях
- Кластеризации данных
- Построения оптимальных путей с обходом препятствий
- Визуализации результатов в различных форматах

2. Общее описание программы

Программа представляет собой консольное приложение, которое:
- Считывает конфигурационный файл с параметрами
- Создает поле заданного размера
- Добавляет на поле гауссовы распределения
- Выполняет операции обработки изображений (бинаризация, выделение компонент связности)
- Производит кластеризацию данных
- Строит триангуляцию Делоне
- Находит оптимальный путь между точками с учетом препятствий
- Сохраняет результаты в различных форматах (BMP, данные для gnuplot)

3. Структура программы

Программа состоит из нескольких основных классов:
- Server: логирование серверной части
- Client: логирование клиентской части
- Field: представление поля данных

- Gauss: гауссово распределение
- Srez: операции с разрезами и алгоритмы обработки
- Control: управляющий класс (диспетчер)
- Interface: пользовательский интерфейс

3.1 Класс Field (Поле данных)

Назначение:
Основной контейнер для хранения и обработки матричных данных. Обеспечивает взаимодействие между различными модулями программы.

Ключевые методы:
а) apply_gauss(const Gauss& g) - добавляет гауссово распределение
- Параметры: объект класса Gauss
- Воздействие: модифицирует матрицу данных и матрицу весов
б) normalize() - нормализует данные
- Делит каждое значение на соответствующий вес
- Обеспечивает корректное наложение гауссов
в) bmp_write() - сохранение в BMP
- Поддержка 3 режимов(ч/б изображение, цветные компоненты, кластеры с центроидами)

Внутренняя структура:

```
std::vector<std::vector<double>> matrix;    // Основные данные
std::vector<std::vector<double>> weights;   // Весовые коэффициенты
std::vector<std::vector<Color>> colors;     // Цветовая информация
```

3.2 Класс Srez (Срез и обработка)

Назначение:
Обработка бинаризованных срезов, выделение компонент и анализ.

Ключевые методы:
а) bin(int h, Field& f) - бинаризация
- Параметр h - пороговое значение
- Создает бинарную копию данных
б) wave(int n) - алгоритм волны
- n - порог размера компоненты
- Возвращает вектор значимых компонент

В) kMeans(int k, int p) - кластеризация
- k - число основных кластеров
- p - число подкластеров

Особенности:
- Использует 8-связность для компонент
- Хранит кэшированные результаты операций
- Оптимизирован для работы с большими матрицами

3.3 Класс Gauss (Гауссово распределение)

Назначение:
Моделирование гауссовых "холмов" на поле.
Формула:

$$G(x,y) = h * \exp(-((x-x0)^2/(2\sigma x^2) + (y-y0)^2/(2\sigma y^2)))$$

Поля:

```
double h;      // Амплитуда
double x0, y0; // Центр
double sigma_x, sigma_y; // Разброс
```

Методы:
а) calculate(x,y) - вычисление значения в точке
б) Операторы сравнения для сортировки

3.4 Класс Server (Серверное логирование)

Назначение:
Централизованное логирование системных событий.

Методы:
а) enable_logging(filename) - активация логирования
б) log(message) - запись сообщения
в) current_time() - форматирование времени

Формат лога:

```
[DD.MM.YYYY HH:MM:SS] - Сообщение
```

Скриншот 3.4.1: Пример лог-файла

```
11.04.2025 23:06:15 - Field created with size 100x100
11.04.2025 23:06:15 - Control passed 'init' command to Field
11.04.2025 23:06:17 - Control passed Gauss command to Field with (h=1.000000, x0=50.000000, y0=50.000000,
11.04.2025 23:06:18 - Control passed 'generate' command to Field
11.04.2025 23:06:18 - Control is applying Gauss #1 to Field
11.04.2025 23:06:18 - Gauss #1applied
11.04.2025 23:06:18 - Gauss completed applying all Gausses
11.04.2025 23:06:18 - A section on height 0 is obtained
11.04.2025 23:09:02 - Field created with size 100x100
11.04.2025 23:09:02 - Control passed 'init' command to Field
11.04.2025 23:09:02 - Control passed Gauss command to Field with (h=1.000000, x0=50.000000, y0=50.000000,
11.04.2025 23:09:02 - Control passed 'generate' command to Field
11.04.2025 23:09:02 - Control is applying Gauss #1 to Field
11.04.2025 23:09:02 - Gauss #1applied
11.04.2025 23:09:02 - Gauss completed applying all Gausses
11.04.2025 23:09:02 - A section on height 0 is obtained
11.04.2025 23:18:32 - Field created with size 100x100
11.04.2025 23:18:32 - Control passed 'init' command to Field
11.04.2025 23:18:32 - Control passed Gauss command to Field with (h=1.000000, x0=50.000000, y0=50.000000,
```

3.5 Класс Client (Клиентское логирование)

Назначение:
Логирование пользовательских действий и интерфейсных событий.

Отличия от Server:
- Более подробные сообщения
- Фиксация пользовательского ввода
- Визуальное форматирование

Скриншот 3.5.1: Пример лог-файла

```
11.04.2025 23:06:09 - The noise value is set to10
11.04.2025 23:06:15 - Interface received 'INI' command with parameters: length=100, width=100
11.04.2025 23:06:17 - Interface received 'g' command with parameters: h=1.000000, x0=50.000000, y0=50.0000
11.04.2025 23:06:18 - Interface received 'generate' command.
11.04.2025 23:06:18 - Interface received 'BIN' command with h = 0
11.04.2025 23:06:18 - Interface received 'WAVE' command with pixel noise:10
11.04.2025 23:08:01 - The noise value is set to10
11.04.2025 23:09:02 - The noise value is set to10
11.04.2025 23:09:02 - Interface received 'INI' command with parameters: length=100, width=100
11.04.2025 23:09:02 - Interface received 'g' command with parameters: h=1.000000, x0=50.000000, y0=50.0000
11.04.2025 23:09:02 - Interface received 'generate' command.
11.04.2025 23:09:02 - Interface received 'BIN' command with h = 0
11.04.2025 23:09:02 - Interface received 'WAVE' command with pixel noise:10
11.04.2025 23:18:32 - The noise value is set to10
11.04.2025 23:18:32 - Interface received 'INI' command with parameters: length=100, width=100
11.04.2025 23:18:32 - Interface received 'g' command with parameters: h=1.000000, x0=50.000000, y0=50.0000
11.04.2025 23:18:32 - Interface received 'generate' command.
11.04.2025 23:18:32 - Interface received 'BIN' command with h = 0
11.04.2025 23:18:32 - Interface received 'WAVE' command with pixel noise:10
11.04.2025 23:18:32 - Interface received 'TRIANG' command with filename: triang.bmp
```

3.6 Класс Control (Управляющий модуль)

Назначение:
Координация работы всех компонентов системы.
Основные функции:

- Управление жизненным циклом объектов
- Обработка ошибок
- Маршрутизация команд

3.7 Класс Interface (Пользовательский интерфейс)

Назначение:
Взаимодействие с пользователем и обработка ввода.

Режимы работы:
а) Пакетный (из файла)
б) Интерактивный (из консоли)
в) Автоматический (по конфигурации)

Методы:
а) run() - основной цикл
б) parse_command() - разбор ввода
в) show_help() - вывод справки

3.8 Вспомогательные структуры

Point/PointD:
- Хранение координат (целые/вещественные)
- Операторы сравнения

Color:
- RGB представление
- Методы обработки цвета

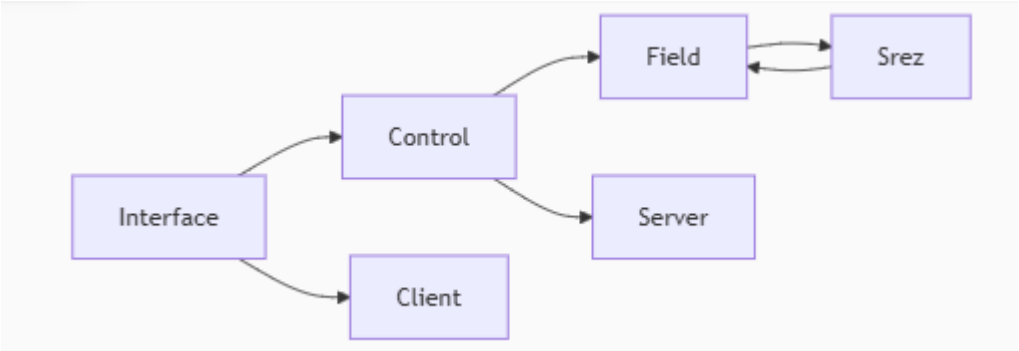
Triangle/Edge:
- Для триангуляции Делоне
- Геометрические операции

3.9 Взаимодействие классов

Типичный сценарий:

- Interface получает команды
- Control создает необходимые объекты
- Field хранит данные
- Srez выполняет обработку
- Server/Client логируют процесс

Потоки данных:



Каждый компонент системы имеет четко определенную зону ответственности и взаимодействует с другими через стандартизированные интерфейсы, что обеспечивает модульность и легкость сопровождения кода. Визуализация структур и процессов помогает лучше понять архитектурные решения, заложенные в программе.

4. Детальное описание алгоритмов программы

4.1 Алгоритм волны (Wave Algorithm)

Что делает:
Алгоритм волны предназначен для выделения связных компонент на бинарном изображении. Он находит все пиксели, соединенные между собой и имеющие одинаковое значение (обычно 0 для объектов и 255 для фона).

Как работает:
а) Поочередно проверяет каждый пиксель изображения
б) При обнаружении непосещенного пикселя, принадлежащего объекту (значение 0):
- Запускает волну (обход в ширину)
- Помечает все связанные пиксели как посещенные
- Сохраняет компоненту, если ее размер превышает заданный порог
в)Использует 8-связность (учитывает диагональные соседи)

Зачем нужен:
- Для выделения отдельных объектов на изображении
- Удаления шума (маленьких компонент)
- Подготовки данных для дальнейшего анализа

Скриншот 4.1: Результат работы

4.2 Алгоритм k-средних (k-means)

Что делает:
Разделяет набор данных на k кластеров, минимизируя суммарное квадратичное отклонение точек кластеров от центроид этих кластеров.

Как работает:
а) Инициализация: случайный выбор k точек в качестве начальных центроид
б) Основной цикл:
- Назначение точек: каждая точка назначается ближайшей центроиде
- Обновление центроид: пересчет положения центроид как среднего всех точек кластера
- Проверка сходимости (изменение положения центроид < ε)
в) В программе реализован двухуровневый вариант:
- Сначала обычная кластеризация на k кластеров

- Затем каждый кластер делится на p подкластеров

Зачем нужен:
- Для автоматической группировки похожих точек
- Выявления структуры данных
- Подготовки к дальнейшему анализу

Скриншот 4.2: Конечный результат кластеризации

4.3 Алгоритм Бойера-Ватсона (Bowyer-Watson)

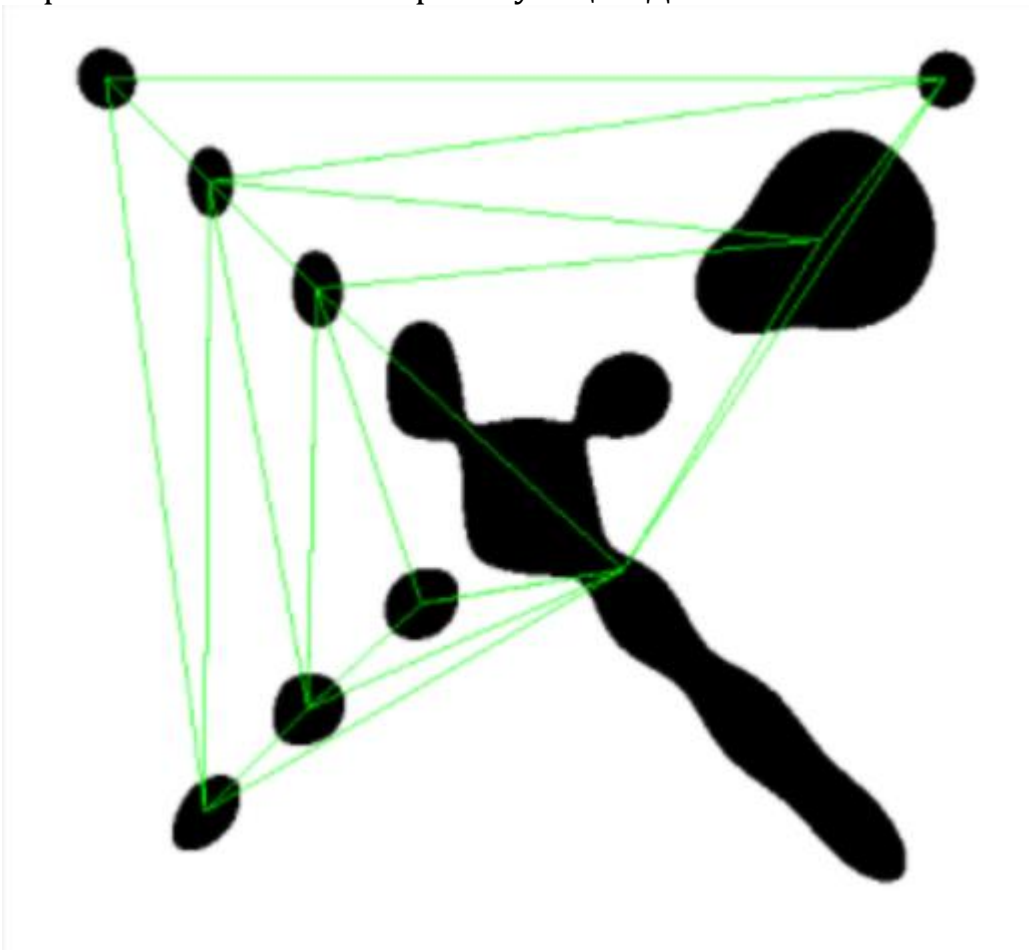
Что делает:
Строит триангуляцию Делоне для заданного набора точек на плоскости.

Как работает:
а) Создание супер-треугольника, содержащего все точки
б) Последовательное добавление точек:
- Поиск треугольников, чьи описанные окружности содержат добавляемую точку

- Удаление этих "плохих" треугольников
- Создание новых треугольников из оставшихся ребер и новой точки
в) Удаление треугольников, связанных с вершинами супер-треугольника

Зачем нужен:
- Для построения сети соединений между центрами компонент
- Создания основы для поиска пути
- Визуализации пространственных отношений

Скриншот 4.3: Готовая триангуляция Делоне



4.4 Алгоритм Дейкстры (Dijkstra's Algorithm)

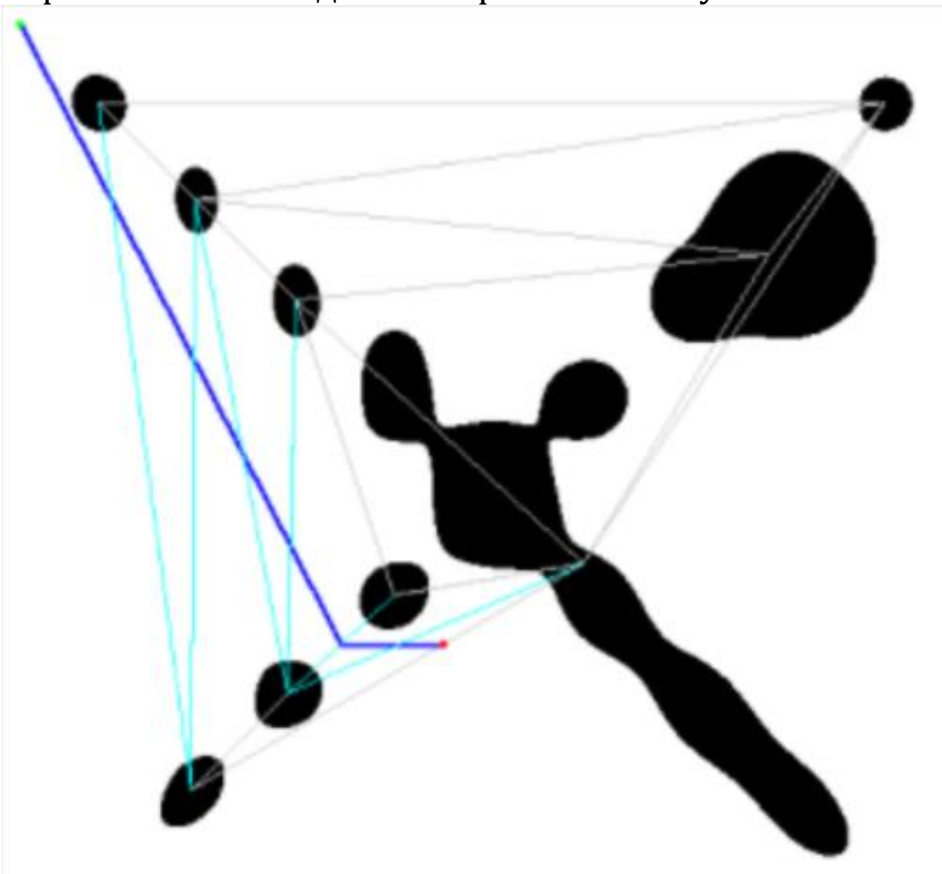
Что делает:
Находит кратчайший путь между двумя точками в графе с неотрицательными весами ребер.

Как работает:
а) Инициализация:
- Присвоение всем вершинам бесконечного расстояния
- Установка расстояния 0 для стартовой вершины
б) Основной цикл:
- Выбор вершины с минимальным расстоянием
- Обновление расстояний до соседей
- Помещение вершины в множество посещенных
в) Восстановление пути от конечной вершины к начальной

Особенности реализации:
- Учет радиуса машины при проверке препятствий
- Штраф за острые углы на поворотах
- Оптимизация через приоритетную очередь

Зачем нужен:
- Для поиска оптимального пути между точками
- Обхода препятствий
- Планирования маршрутов

Скриншот 4.4: Найденный кратчайший путь



4.5 Бинаризация (Thresholding)

Что делает:
Преобразует полутоновое изображение в бинарное по заданному порогу.

Как работает:
а) Для каждого пикселя исходного изображения:
- Если значение ≥ порога -> 255 (белый)
- Если значение < порога -> 0 (черный)
б) В программе реализован простой пороговый метод

Зачем нужен:
- Упрощение изображения для дальнейшей обработки
- Выделение значимых областей
- Подготовка к морфологическим операциям

Скриншот 4.5: Результат бинаризации

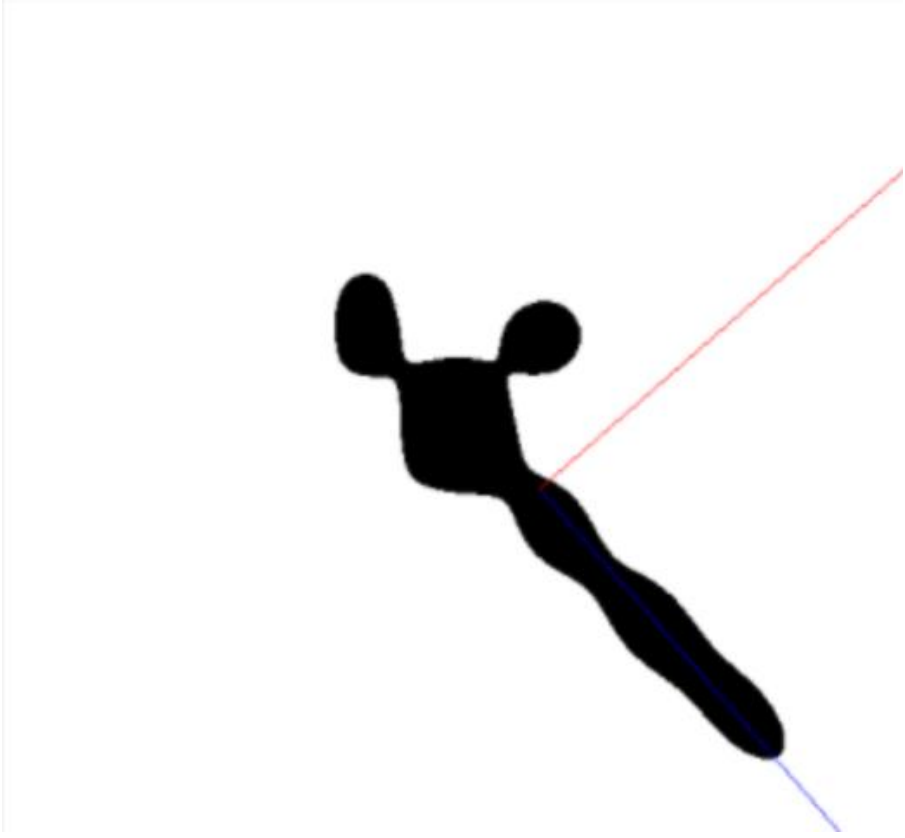
4.6 Поиск компонент связности (Connected Components)

Что делает:
Находит и анализирует связные области на бинарном изображении.

Как работает:
а) Применение алгоритма волны для нахождения компонент
б) Расчет характеристик для каждой компоненты:
- Границы (min/max x,y)
- Центр масс
- Собственные векторы и значения (для анализа формы)
в) Фильтрация по размеру (удаление шумов)

Зачем нужен:
- Анализ формы и расположения объектов
- Подготовка к триангуляции
- Удаление артефактов и шума

Скриншот 4.6: Визуализация собственных векторов одной из компонент



Каждый алгоритм играет важную роль в конвейере обработки данных программы, обеспечивая поэтапное преобразование исходных данных в конечный результат - оптимальный путь, учитывающий все особенности входного изображения или сгенерированного ландшафта.

5. Этапы работы программы

5.1 Этап 1: Ввод конфигурационного файла

Варианты исхода:
- Конфигурационный файл введен корректно
- Конфигурационный файл не введен или содержит ошибки

Скриншот 1: Пример корректного конфигурационного файла (config.txt):

```
Log_Server = ON
Log_Server_filename = server_log.txt
Log_Client = ON
Log_Client_filename = client_log.txt
Radius = 5
Batch_pause = OFF
Batch_file = ON
Batch_filename = batch.txt
Noize = 10
Gauss_DEF = 1.0 50 50 10 10
Field_DEF = 100 100
```

Скриншот 2: Ошибка при открытии конфигурационного файла (вывод в консоль):

```
Failed to open file: config.txt
```

5.2 Этап 2: Инициализация поля

Варианты исхода:
- Поле инициализировано с параметрами из конфига
- Поле инициализировано с параметрами из batch-файла
- Ошибка инициализации (неверные параметры)

Скриншот 3: Успешная инициализация поля 100x100 (вывод в консоль):

```
Field created with size 100x100
```

5.3 Этап 3: Добавление гауссовых распределений

Варианты исхода:
- Гауссы добавлены из конфига
- Гауссы добавлены из batch-файла
- Использованы гауссы по умолчанию

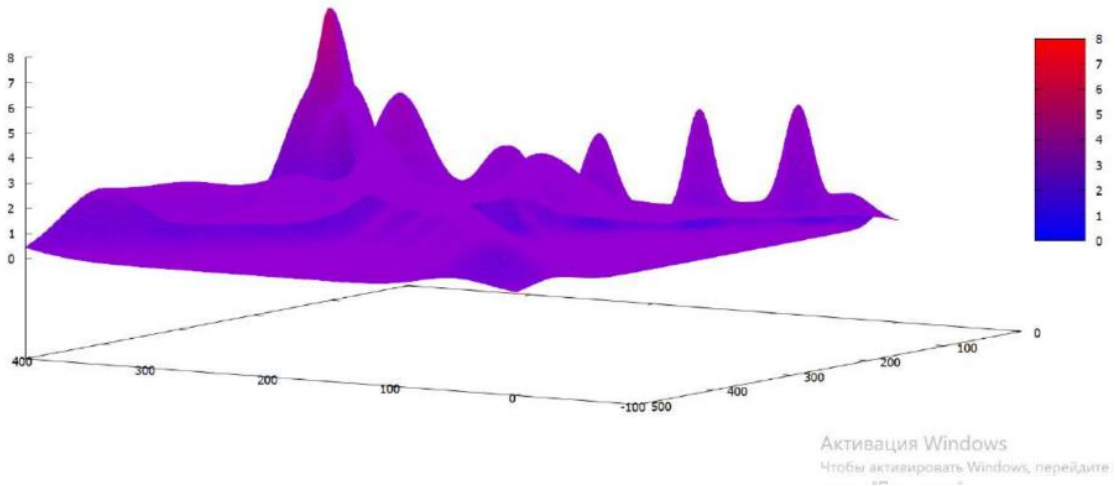
Скриншот 4: Пример добавления гаусса (вывод в консоль):

```
Added Gauss with h=1.0, x0=50, y0=50, sigma_x=10, sigma_y=10
```

5.4 Этап 4: Генерация поля

Варианты исхода:
- Успешная генерация поля
- Ошибка генерации (нет гауссов)

Скриншот 5: Визуализация сгенерированного поля в gnuplot:

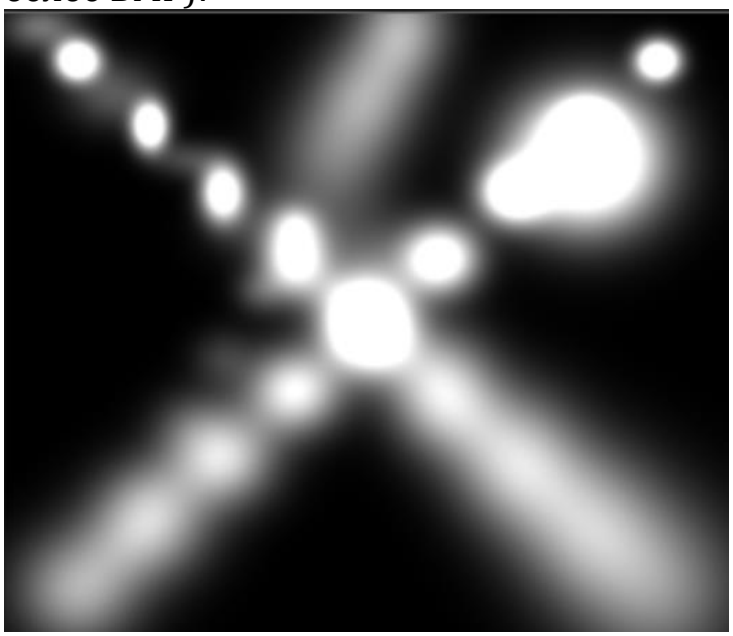


(Изображение поверхности с гауссовыми "холмами")

5.5 Этап 5: Создание разреза (bin)

Варианты исхода:
- Успешное создание разреза
- Ошибка (не инициализировано поле)

Скриншот 6: Бинарное изображение после операции bin (черно-белое BMP):

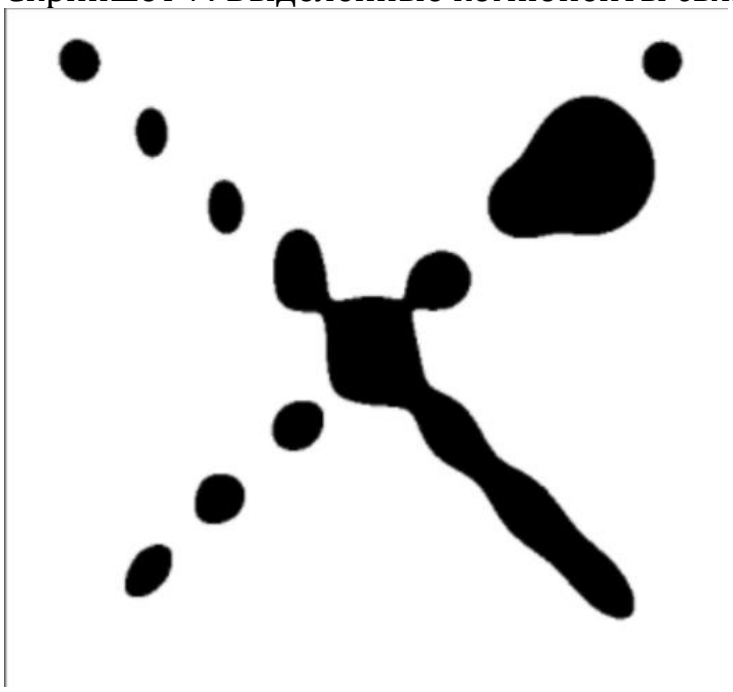


(Изображение с белыми и черными областями)

5.6 Этап 6: Алгоритм волны (wave)

Варианты исхода:
- Успешное выделение компонент
- Компоненты не найдены

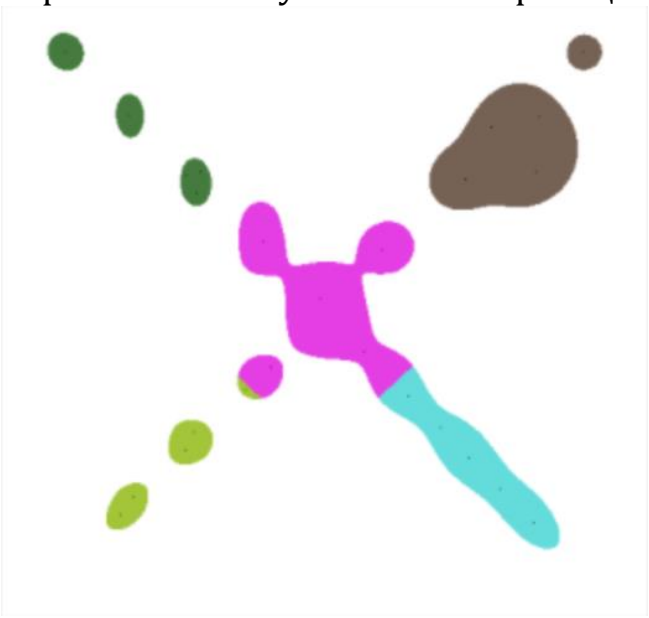
Скриншот 7: Выделенные компоненты связности:



5.7 Этап 7: Кластеризация (k-means)

Варианты исхода:
- Успешная кластеризация
- Ошибка (недостаточно точек)

Скриншот 8: Результат кластеризации:

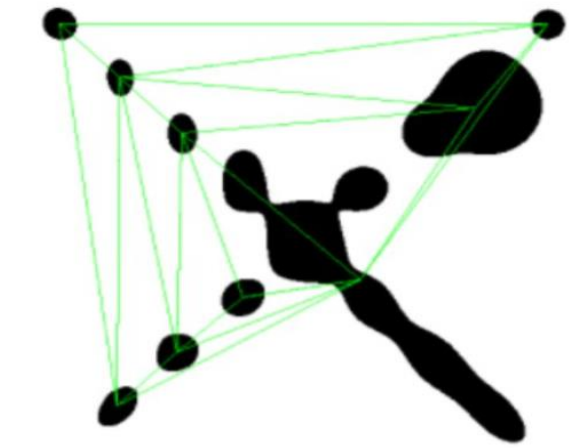


(Разноцветные кластеры с отмеченными центрами)

5.8 Этап 8: Триангуляция Делоне

Варианты исхода:
- Успешная триангуляция
- Ошибка (недостаточно точек)

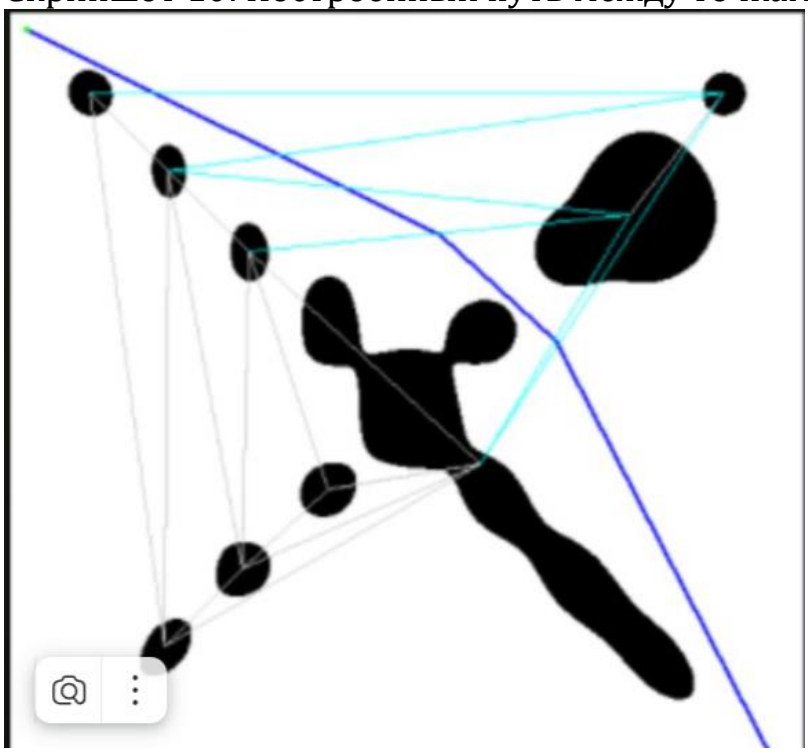
Скриншот 9: Триангуляция центров компонент:



5.9 Этап 9: Построение пути (road)

Варианты исхода:
- Успешное построение пути
- Путь не найден
- Ошибка (точки внутри препятствий)

Скриншот 10: Построенный путь между точками:



6. Команды программы

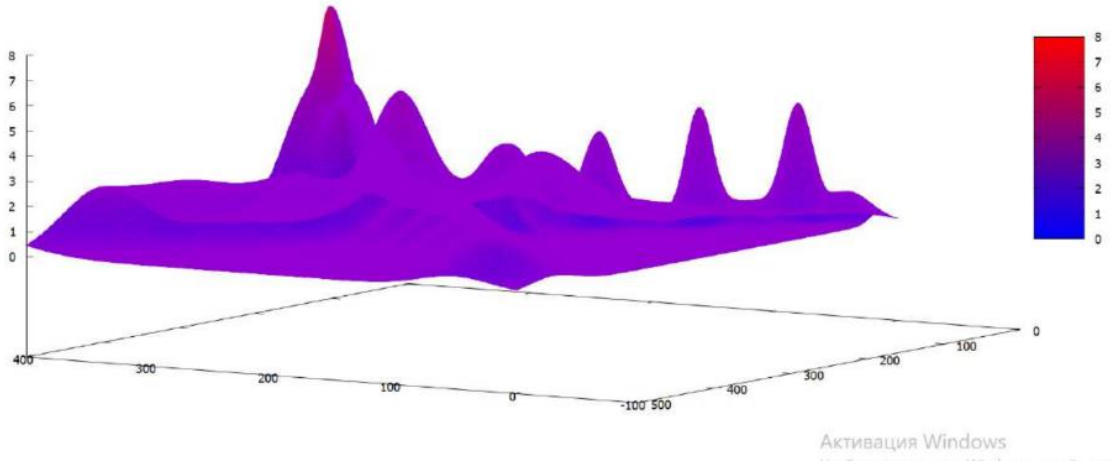
Основные команды, которые можно использовать в batch-файле:
- INI <length> <width> - инициализация поля
- G <h> <x0> <y0> <sigma_x> <sigma_y> - добавление гаусса
- GEN - генерация поля
- GNU <filename> - сохранение данных для gnuplot
- BMP <filename> - сохранение BMP
- RBMP <filename> - чтение BMP
- BIN <h> - создание разреза
- WAVE - алгоритм волны
- KMEANS <k> <p> - кластеризация
- TRIANG <filename> - триангуляция
- ROAD <x1>,<y1>,<x2>,<y2> - построение пути

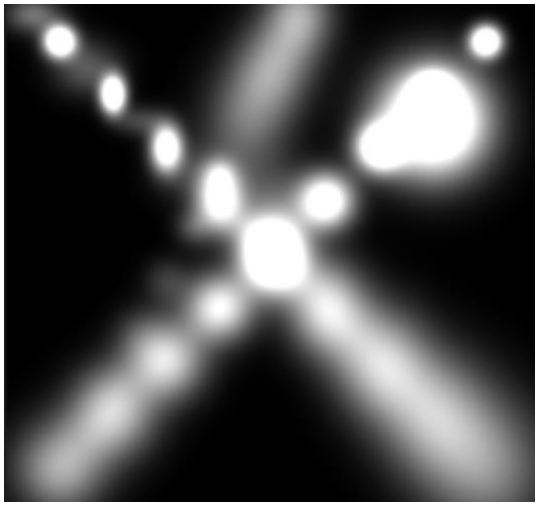
7. Пример использования.

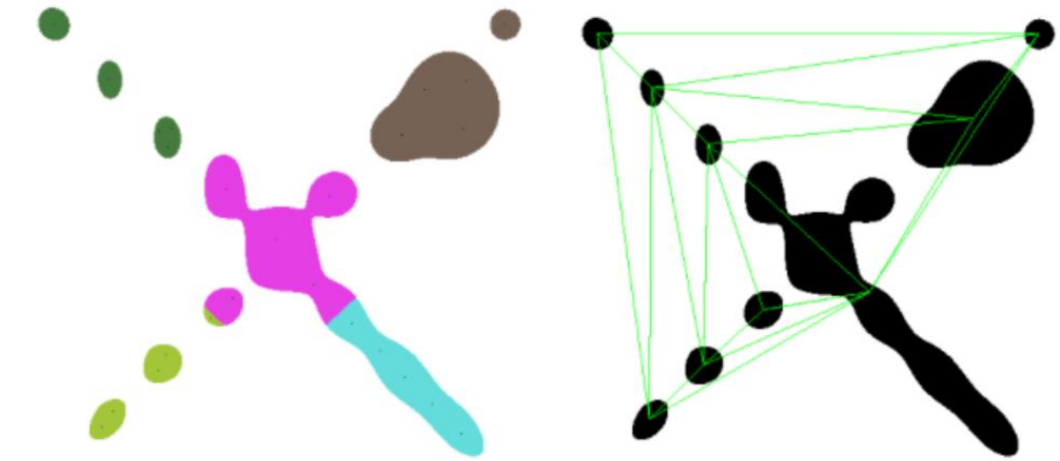
Пример: Полный цикл обработки

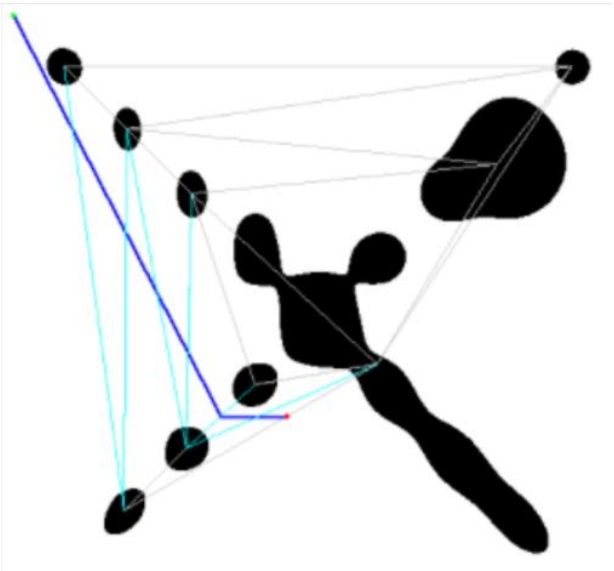
```
INI 200 200
G 1.0 50 50 20 20
G 0.8 150 150 15 15
GEN
BIN 0.5
WAVE
KMEANS 3 2
TRIANG triang.bmp
ROAD 10,10,190,190
```

Скриншот 12: Все промежуточные результаты









8. Заключение.

Данная программа предоставляет мощный инструмент для обработки и анализа двумерных данных. Она может быть использована в различных областях, таких как:
- Обработка изображений
- Геоинформационные системы
- Навигационные системы
- Научная визуализация

Программа имеет модульную структуру, что позволяет легко расширять её функциональность. Логирование всех операций помогает в отладке и анализе работы программы.