Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Кафедра «Прикладные информационные технологии»

**Фильтрация изображений. Работа с контурами**

**Отчет по курсу**

**«Методы обработки сигналов и изображений**

Выполнил:

Студент группы м2-ИФСТ-11

Маркелов Александр Сергеевич

Проверил:

ст. преподаватель каф. ПИТ

Пиминов Дмитрий Алексеевич

Саратов 2025

# **Теоретическая часть**

Бинаризация изображения — это процесс преобразования исходного изображения (обычно в градациях серого или цветного) в двоичное изображение, где каждый пиксель принимает одно из двух значений: 0 (черный) или 1 (белый). Этот процесс является ключевым шагом в задачах компьютерного зрения, таких как определение контуров, сегментация объектов и распознавание форм. Определение контуров, в свою очередь, заключается в выделении границ объектов на изображении, что позволяет анализировать их форму, структуру и взаимное расположение.

Бинаризация необходима, поскольку алгоритмы обнаружения контуров (например, оператор Кэнни или алгоритмы на основе градиентов) лучше работают с изображениями, где границы четко разделены. Существует множество методов бинаризации, каждый из которых подходит для определенных типов изображений и условий освещения. Далее я подробно опишу основные способы бинаризации, их принципы работы, а затем перейду к иерархии контуров и родительско-дочерним отношениям.

## **1.1. Способы бинаризации изображений**

### **1. Фиксированный порог (Thresholding with Fixed Threshold)**

#### Принцип работы

Фиксированный порог — это простейший метод бинаризации. Задается единое значение порога *T* (например, 128 для изображения в градациях серого, где интенсивность пикселей варьируется от 0 до 255). Каждый пиксель изображения сравнивается с этим порогом:

* Если интенсивность пикселя *I(x,y)≥T*, он становится белым (1).
* Если *I(x,y)<T*, он становится черным (0).

#### Формула

#### Преимущества

* Простота реализации.
* Быстрота выполнения.

#### Недостатки

* Не учитывает вариации освещения или контраста в разных частях изображения.
* Работает плохо на изображениях с неоднородным фоном.

#### Пример

Предположим, у нас есть изображение с объектом на светлом фоне. Если фон имеет интенсивность 200, а объект — 50, то порог *T=100* успешно разделит их. Однако, если освещение меняется (например, часть фона становится темнее), фиксированный порог может ошибочно классифицировать пиксели.

#### Применение для контуров

После бинаризации изображение передается в алгоритм обнаружения контуров, например, оператор Кэнни, который выделяет границы между черными и белыми областями.

#### **Адаптивный порог (Adaptive Thresholding)**

#### Принцип работы

Адаптивный порог решает проблему неоднородного освещения. Вместо единого значения *T* порог вычисляется локально для каждого пикселя на основе его окрестности (обычно квадратного окна размером *N×N*). Значение порога зависит от средней интенсивности или медианы в этой области, скорректированной на константу *C* .

#### Формула

или

где:

* — интенсивности пикселей в окрестности пикселя *(x,y)*,
* *C* — константа смещения (обычно небольшое положительное число).

Затем применяется стандартное правило бинаризации:

#### Преимущества

* Хорошо работает на изображениях с переменным освещением.
* Адаптируется к локальным особенностям.

#### Недостатки

* Требует выбора размера окна и параметра *C*, что может быть неочевидно.
* Более высокая вычислительная сложность.

#### Пример

Рассмотрим изображение документа, где часть текста освещена ярче, а часть — темнее. Фиксированный порог может потерять часть текста, тогда как адаптивный порог (например, с окном 11x11 и *C*=2) выделит текст везде, учитывая локальные изменения яркости.

#### Применение для контуров

Адаптивная бинаризация создает более точные границы объектов, что улучшает качество работы алгоритмов контурного анализа.

### **3. Метод Отсу (Otsu’s Method)**

#### Принцип работы

Метод Отсу — это автоматический способ выбора порога, основанный на гистограмме интенсивностей изображения. Он стремится минимизировать внутриклассовую дисперсию (или максимизировать межклассовую дисперсию) между двумя классами пикселей: фоном и объектом.

1. Строится гистограмма интенсивностей изображения.
2. Для каждого возможного значения порога *T* (от 0 до 255) вычисляется:
   * Доля пикселей в классе фона (ω0),
   * Доля пикселей в классе объекта (ω1​),
   * Средние интенсивности классов *(μ0* и *μ1*),
   * Межклассовая дисперсия: σ2= ω0ω1 (*μ0* - *μ1*)2.
3. Выбирается *T*, при котором σ2 максимальна.

#### Формула

#### Преимущества

* Полностью автоматический метод.
* Хорошо работает на изображениях с бимодальной гистограммой (два четких пика).

#### Недостатки

* Плохо справляется с изображениями, где гистограмма не бимодальна.
* Не учитывает пространственную информацию.

#### Пример

На изображении монеты на однородном фоне гистограмма покажет два пика: один для фона (яркий), другой для монеты (темный). Метод Отсу автоматически найдет порог между ними, например, *T=150*, что позволит четко выделить контур монеты.

#### Применение для контуров

Метод Отсу часто используется как предварительный шаг перед алгоритмами вроде findContours в OpenCV, обеспечивая оптимальное разделение объектов и фона.

### **4. Бинаризация на основе кластеризации (K-Means Clustering)**

#### Принцип работы

Этот метод использует алгоритм кластеризации K-Means для разделения пикселей изображения на два кластера (фон и объект). Интенсивности пикселей рассматриваются как точки в одномерном пространстве, и алгоритм итеративно:

1. Случайно выбирает два начальных центра кластеров.
2. Присваивает каждый пиксель ближайшему центру.
3. Пересчитывает центры как среднее значение интенсивностей в кластерах.
4. Повторяет шаги, пока центры не стабилизируются.

Порог *T* устанавливается как среднее между финальными центрами кластеров.

#### Преимущества

* Может учитывать сложные распределения интенсивностей.
* Адаптируется к данным.

#### Недостатки

* Зависит от начальной инициализации.
* Вычислительно сложен.

#### Пример

На изображении с шумным фоном и объектом K-Means может выделить два кластера: один с интенсивностью около 50 (объект), другой около 200 (фон). Порог *T=125* разделит их.

#### Применение для контуров

После кластеризации бинаризованное изображение позволяет точно выделить контуры сложных объектов.

### **5. Бинаризация с использованием морфологических операций**

#### Принцип работы

Этот метод сочетает стандартную бинаризацию (например, фиксированный порог) с морфологическими операциями (эрозия, дилатация, открытие, закрытие), чтобы улучшить качество результата. Например:

* Эрозия убирает мелкие шумы на фоне.
* Дилатация заполняет разрывы в объектах.

#### Пример

На зашумленном изображении текста фиксированный порог может оставить мелкие точки шума. Применение эрозии удалит их, а последующая дилатация восстановит контуры текста.

#### Применение для контуров

Морфологические операции делают контуры более гладкими и непрерывными, что упрощает их дальнейший анализ.

## **1.2. Иерархия контуров**

### Что такое иерархия контуров?

После бинаризации и обнаружения контуров (например, с помощью функции findContours в OpenCV) возникает вопрос: как организованы найденные контуры? Иерархия контуров — это структура, описывающая взаимное расположение контуров, включая вложенность (контуры внутри других контуров). Она представлена в виде дерева, где каждый контур имеет:

* Уникальный идентификатор.
* Связь с родительским и дочерними контурами.
* Указание уровня вложенности.

В OpenCV иерархия возвращается как массив, где для каждого контура указаны:

1. Индекс следующего контура на том же уровне.
2. Индекс предыдущего контура на том же уровне.
3. Индекс первого дочернего контура.
4. Индекс родительского контура.

Если какого-то элемента нет, используется значение -1.

#### Пример структуры иерархии

Контур 0 (внешний)

├── Контур 1 (дочерний)

│ ├── Контур 2 (внутренний)

│ └── Контур 3 (внутренний)

└── Контур 4 (дочерний)

### Родительско-дочерние отношения контуров

#### Определение

* **Родительский контур** — это внешний контур, внутри которого находится другой контур.
* **Дочерний контур** — это внутренний контур, который полностью содержится внутри родительского.

Эти отношения возникают из-за вложенности объектов на изображении. Например, если на изображении есть кольцо (внешний круг с отверстием внутри), то:

* Внешний контур кольца — родитель.
* Контур отверстия — дочерний.

#### Как это работает?

Алгоритм findContours анализирует бинаризованное изображение и определяет вложенность на основе топологии. Например:

* Если белая область (1) окружена черной (0), это внешний контур.
* Если внутри белой области есть черная (0), окруженная белой (1), это дочерний контур.

#### Уровни вложенности

* Уровень 0: внешние контуры (нет родителя).
* Уровень 1: дочерние контуры уровня 0.
* Уровень 2: дочерние контуры уровня 1 и так далее.

### Примеры иерархии контуров

#### Пример 1: Кольцо

* Бинаризованное изображение: белое кольцо на черном фоне.
* Контуры:
  + Контур 0: внешняя окружность кольца (уровень 0, родитель).
  + Контур 1: внутренняя окружность (уровень 1, дочерний).
* Иерархия: [Контур 0: дочерний=1, родитель=-1], [Контур 1: дочерний=-1, родитель=0].

#### Пример 2: Матрешка

* Изображение: три вложенных круга (как матрешка).
* Контуры:
  + Контур 0: самый большой круг (уровень 0).
  + Контур 1: средний круг (уровень 1, дочерний от 0).
  + Контур 2: маленький круг (уровень 2, дочерний от 1).
* Иерархия: [Контур 0: дочерний=1, родитель=-1], [Контур 1: дочерний=2, родитель=0], [Контур 2: дочерний=-1, родитель=1].

#### Пример 3: Текст

* Изображение: буква "О" на белом фоне.
* Контуры:
  + Контур 0: внешняя граница буквы (уровень 0).
  + Контур 1: внутренняя граница (уровень 1).
* Иерархия: аналогична кольцу.

## Заключение

Бинаризация изображений — это фундаментальный процесс для выделения контуров, и выбор метода зависит от характеристик изображения:

* Фиксированный порог прост, но ограничен.
* Адаптивный порог гибок для сложных условий.
* Метод Отсу автоматизирует выбор порога.
* K-Means подходит для сложных распределений.
* Морфология улучшает качество результата.

Иерархия контуров добавляет структурную информацию, позволяя анализировать вложенные объекты. Родительско-дочерние отношения помогают понять топологию сцены, что критично для задач вроде распознавания объектов, анализа текста или 3D-реконструкции.

# **Практическая часть**

**Назначение программы**

"ImageProcessingApp" предназначено для выполнения следующих задач:

Загрузка и отображение изображений: Пользователь может загрузить изображение в популярных форматах (PNG, JPG, JPEG, BMP) и увидеть его в исходном виде.

Обработка изображений: Приложение предоставляет набор фильтров и алгоритмов для изменения визуальных характеристик изображения, включая повышение резкости, размытие в движении, эффект тиснения, медианную фильтрацию и выделение границ.

Комбинирование эффектов: В отличие от многих простых инструментов обработки, программа позволяет последовательно применять несколько преобразований к изображению, сохраняя результат предыдущих операций.

Сохранение результатов: Обработанное изображение можно сохранить на диск в выбранном пользователем формате.

Логирование действий: Все операции фиксируются в журнале с указанием времени и типа действия, что обеспечивает прозрачность и возможность анализа процесса обработки.

Сброс изменений: Пользователь может в любой момент вернуть изображение к исходному состоянию.

**Функциональные возможности**

**Загрузка изображения**:

* Поддерживаемые форматы: PNG, JPG, JPEG, BMP, GIF
* Использование стандартного JFileChooser с фильтром расширений
* Автоматическое отображение в двухпанельном интерфейсе (оригинал/результат)
* Сохранение пропорций при масштабировании для отображения

**Обработка изображения**:

**Повышение резкости**: Ядро 3x3: [[0,-1,0], [-1,5,-1], [0,-1,0]] с усилением центрального пикселя.

**Размытие в движении**: Большое ядро 9x9 с линейным распределением для эффекта горизонтального движения.

**Тиснение**: Ядро 3x3: [[-2,-1,0], [-1,1,1], [0,1,2]] с последующей градацией серого.

**Медианная фильтрация**: Окно 3x3 (радиус 1) для эффективного удаления шума с сохранением границ.

**Детектор Canny**: Упрощенная реализация с:

* + Гауссовым размытием (3x3)
  + Оператором Собеля
  + Пороговой обработкой (фиксированный порог 50)

**Оператор Робертса**: Диагональные разности 2x2 с пороговой обработкой (порог 30).

**Сброс изменений**:

* Мгновенный возврат к исходному изображению
* Реализация через хранение оригинального BufferedImage

**Архитектура программы**

**Класс ImageProcessorApp (JFrame):**

* Главное окно приложения
* Инициализация компонентов:
  + ImagePanel - для отображения
  + ControlPanel - панель управления
  + ImageProcessor - обработчик изображений
* Настройка layout (BorderLayout)

**Класс ImagePanel (JPanel):**

* Двухпанельное отображение (оригинал/результат)
* Автомасштабирование с сохранением пропорций
* Переопределенный paintComponent для рендеринга

**Класс ControlPanel (JPanel):**

* Кнопки управления:
  + Загрузка/сброс
  + 6 фильтров обработки
* Обработка событий через ActionListener
* Валидация наличия загруженного изображения

**Класс ImageProcessor:**

* Хранение оригинального изображения
* Реализация всех алгоритмов обработки:
  + Сверточные операции (ConvolveOp)
  + Медианная фильтрация
  + Детекторы границ
* Вспомогательные методы:
  + Преобразование в grayscale
  + Применение ядер свертки

**Примеры использования**

1. **Улучшение качества фото:**
   * Загрузить размытое изображение
   * Применить "Повышение резкости"
   * Удалить шум "Медианным фильтром"

Изображение выглядит как млекопитающее, примат, обезьяна, Мартышковые

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как черно-белый, зарисовка, млекопитающее, рисунок

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 - использование дектора canny и медианной фильтрации.

1. **Творческий проект**:
   * Загрузить портрет
   * Применить "Тиснение" для эффекта гравюры
   * Добавить "Размытие в движении" для динамики
   * Экспортировать в PNG

Изображение выглядит как млекопитающее, примат, обезьяна, Мартышковые

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 - использование Тиснения и размытия в движении.

1. **Исследовательская работа**:

Исследователь загружает медицинское изображение (например, рентген), повышает резкость для выделения деталей, затем применяет "Оператор Робертса" для анализа границ. Комбинирование фильтров помогает подготовить данные для анализа (пример представлен на рисунке 3).

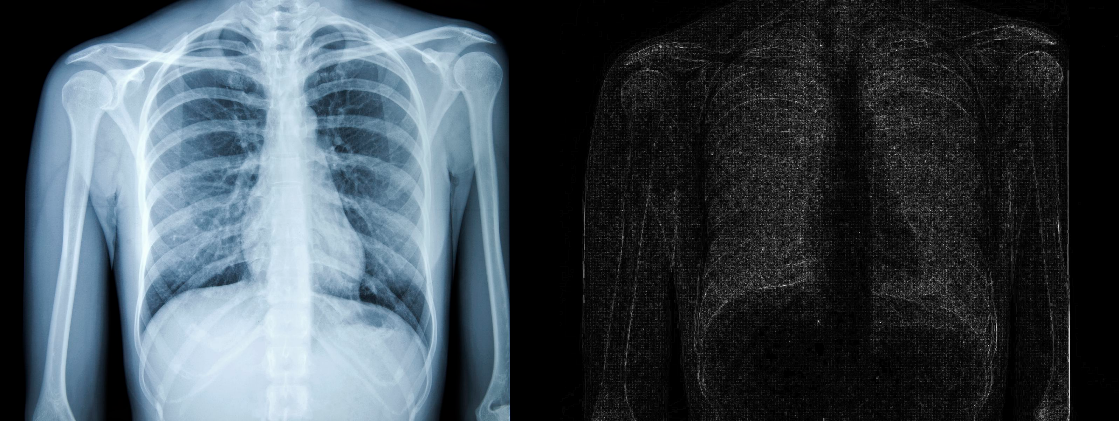


Рисунок 3 - использование повышения резкости и применения Оператора Робертса.

**Преимущества программы**

1. **Чистая архитектура:** Разделение логики, отображения и управления, а также легкость добавления новых фильтров.
2. **Производительность**: Оптимизированные алгоритмы на Java2D. Работа с памятью через BufferedImage.
3. **Надежность**: Обработка ошибок и поддержка кириллических путей обеспечивают стабильность.
4. **Кроссплатформенность**: Работает на Windows, macOS и Linux благодаря Python и Tkinter.

**Заключение**

Разработанное приложение успешно демонстрирует возможности обработки изображений с использованием стандартных Java-технологий. Применение Swing для GUI и Java2D для обработки изображений обеспечило кроссплатформенность и стабильность работы без внешних зависимостей. Программа реализует все базовые операции обработки изображений — от повышения резкости до детектирования границ — через четко структурированные алгоритмы на Java.