Лабораторная работа 3  
Документация

# Тема:

Создание приложения для анализа и обработки изображений.

# Инструменты разработки:

Python (Tkinter, OpenCV, PIL) — для создания графического интерфейса и работы с изображениями.

PyInstaller — для создания исполняемого .exe-файла.

Git — для контроля версий проекта.

PyCharm — среда разработки.

# Использованные библиотеки и технологии:

* **Tkinter** — библиотека для создания GUI:
* **Frame** — для организации интерфейса.
* **Button, Label, OptionMenu** — для взаимодействия с пользователем и отображения изображений.
* **MessageBox** — для отображения сообщений об ошибках.
* **OpenCV** — библиотека компьютерного зрения, используемая для:
* Применения морфологических операций и фильтров к изображениям (например, медианный, максимальный, минимальный фильтры).
* Чтения изображений и выполнения операций с ядрами (структурными элементами).
* **Pillow (PIL)** — библиотека для работы с изображениями:
* Позволяет загружать, изменять размер и конвертировать изображения для отображения в Tkinter.
* **PyInstaller** — для создания исполняемого файла:
* Создание .exe, работающего с папкой изображений, используя --add-data для добавления тестовых файлов рядом с исполняемым файлом.

# Функционал приложения:

* **Загрузка изображений:**
  + - Приложение позволяет пользователю загружать изображения для обработки. Поддерживаемые форматы: JPEG, PNG, BMP, TIFF.
  + - Также можно выбрать фильтр или морфологическую операцию для обработки.
* **Обработка изображений:**
  + - Доступны методы фильтрации: медианный, максимальный, минимальный и нелинейный (по статистическим характеристикам).
  + - Доступны морфологические операции: дилатация (расширение), эрозия (сужение), открытие, закрытие. Пользователь выбирает вид структурного элемента (прямоугольник, эллипс, крест) и размер ядра.
* **Вывод обработанных изображений:**
  + - Обработанные изображения отображаются в интерфейсе, позволяя пользователю сравнить оригинал и результат обработки.

# Теоретические аспекты:

* **Фильтры:**
  + **Медианный фильтр:** устраняет шум, заменяя каждый пиксель медианным значением его окрестности, полезен при удалении солевых шумов.
  + **Максимальный и минимальный фильтры:** заменяют значение пикселя максимальным или минимальным значением в окрестности, используются для выделения контуров или уплотнения объектов.
* **Морфологические операции:**
  + **Дилатация (расширение):** увеличивает светлые области, заполняет пробелы.
  + **Эрозия (сужение):** уменьшает светлые области, удаляет мелкие объекты.
  + **Открытие:** сочетание эрозии и дилатации для удаления мелких шумов.
  + **Закрытие:** сочетание дилатации и эрозии для заполнения мелких пробелов.
* **Структурные элементы и размер ядра:**
  + **Форма структурного элемента** влияет на результат обработки — прямоугольник, крест и эллипс позволяют по-разному формировать области на изображении.
  + **Размер ядра** определяет размер обрабатываемой области вокруг каждого пикселя, чем больше ядро, тем сильнее влияние на изображение.

# Подбор изображений для тестирования

Для тестирования каждого метода использованы изображения с различными характеристиками, чтобы проверить эффективность и видимость изменений:

1. **Медианный фильтр:** изображения с солевыми шумами, где применение медианного фильтра улучшит качество путем удаления ярких пятен.
2. **Максимальный фильтр:** изображения с высококонтрастными объектами, на которых проверяется эффект на светлых участках.
3. **Минимальный фильтр:** изображения с темными объектами, где уменьшение яркости объектов важно для проверки границ.
4. **Морфологические операции:**
   * **Дилатация** — проверка на изображения с разреженными объектами, где требуется заполнить пробелы.
   * **Эрозия** — изображения с мелкими объектами для их удаления.
   * **Открытие и закрытие** — изображения с шумом и пробелами, где применение операций помогает очищать фон или соединять объекты.

# Структура файлов:

**Python скрипт:** основной код, обрабатывающий изображения и работающий с интерфейсом.

**Папка с тестовыми изображениями:** организована по методам.

**Исполняемый файл (.exe):** создан с помощью PyInstaller, содержит все необходимые для работы компоненты и тестовые изображения.

# Шаги по упаковке в EXE

**Подготовка скрипта к работе с относительными путями** для доступа к папке Тесты.

**Создание EXE с PyInstaller:** в командной строке выполнена команда:

pyinstaller --onefile --noconsole --add-data "Тесты;Тесты" image\_processor.py

**Проверка работы EXE-файла:** запущен EXE для проверки правильного отображения изображений и интерфейса.

# Вывод:

Приложение демонстрирует возможность обработки изображений с использованием различных фильтров и морфологических операций, позволяя пользователю получать визуальные результаты в интерактивном интерфейсе.