**НПОУ "ЯКУТСКИЙ КОЛЛЕДЖ ИННОВАЦИОННЫХ ТЕХНОЛОГИЙ"**

Проектная работа

по дисциплине: «Языки программирования»

на тему: «Разработка приложения для распознавания простых объектов на изображениях с использованием OpenCV.»

Выполнили:

студенты 2 курса, гр. КИСП-23(1)

Оконешников Р., Мандаров А., Былчахов А.

Руководитель:

Федоров Д.А.

Якутск, 2025

Содержание

[ВВЕДЕНИЕ 3](#_xbj2b632dinz)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_wyoqufb66f1b)

[1.1. Терминология по проекту и глоссарий 6](#_8a26hoz66cag)

[1.2. Распределение ролей и работы 7](#_7ruu16b5gko9)

[1.3. Стек технологий 9](#_5dq0weomd8w0)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 11](#_t7mi44uifcze)

[2.1. Архитектура ПО 11](#_jar5o83jnhuv)

[2.2. Разработка проекта по ролям 12](#_tic1zy49nf07)

[2.3. Контроль выполнения плана 17](#_pahnnn67m13p)

[ЗАКЛЮЧЕНИЕ 20](#_bclifgfywwpn)

# ВВЕДЕНИЕ

Компьютерное зрение становится все более важным направлением исследований и разработок. OpenCV предоставляет мощные инструменты для реализации базовых и продвинутых алгоритмов обработки изображений, что делает его популярным выбором среди разработчиков.

Актуальность – Компьютерного зрение (Computer Vision, CV) является одной из наиболее динамично развивающихся областей искусственного интеллекта. OpenCV (Open Source Computer Vision Library) — это открытая библиотека для работы с компьютерным зрением, которая активно используется как в академических исследованиях, так и в коммерческих проектах. Рассмотрим актуальность OpenCV в контексте современных тенденций развития технологий. Современные приложения компьютерного зрения все чаще используют технологии глубокого обучения (Deep Learning). OpenCV используется в самых разных отраслях благодаря своей универсальности. OpenCV поддерживает множество операционных систем (Windows, Linux, macOS) и архитектур (x86, ARM), что делает его доступным для разработки как настольных, так и мобильных приложений. Библиотека также совместима с различными языками программирования, включая Python, C++, Java и MATLAB. Это обеспечивает широкую доступность и удобство использования.

С развитием технологий автоматизации и искусственного интеллекта растет спрос на системы, способные обрабатывать и интерпретировать визуальную информацию.

OpenCV является одной из наиболее популярных библиотек для обработки изображений благодаря своей открытости, доступности и обширному сообществу разработчиков. Это облегчает внедрение новых решений и ускоряет процесс разработки.

Основная цель проекта заключается в создании программного обеспечения, способного автоматически определять и классифицировать объекты на основе их визуальных характеристик.

Задачи

Для процесса обнаружения простых объектов на изображениях с использованием методов компьютерного зрения включает в себя – Алгоритмы предварительной обработки изображения, методы сегментации и выделения объектов, техники классификации и распознавания объектов, а также взаимодействия между алгоритмами Opencv и использованием десктопного приложения.

Объект исследования

Объектом исследования в данной работе является процесс автоматического обнаружения и классификации простых объектов на изображениях с использованием библиотеки OpenCV и методов компьютерного зрения.

Ключевые аспекты объекта исследования:

Методы обработки изображений

Алгоритмы предварительной обработки (фильтрация, бинаризация, морфологические операции).

Техники сегментации (контурный анализ, цветовая сегментация, выделение границ).

Методы классификации объектов (геометрические признаки, сопоставление шаблонов).

Интеграция OpenCV в десктопное приложение

Взаимодействие алгоритмов компьютерного зрения с графическим интерфейсом (GUI).

Оптимизация производительности при обработке изображений.

Реализация функций загрузки, обработки и сохранения результатов.

Тестирование и валидация

Проверка точности обнаружения объектов (геометрические фигуры, лица, глаза).

Анализ устойчивости алгоритмов к различным условиям (освещение, шумы, масштаб).

Сравнение эффективности классических методов OpenCV и современных подходов на основе глубокого обучения.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

# Терминология по проекту и глоссарий

Задача состоит в том, чтобы разработать приложение, которое будет принимать изображение в качестве входных данных и возвращать информацию о наличии определенных объектов на этом изображении. Для упрощения мы ограничимся распознаванием таких объектов, как круги, квадраты и треугольники.

Используемые технологии:

Python – основной язык программирования.

OpenCV – библиотека для обработки изображений и компьютерного зрения.

NumPy – библиотека для работы с многомерными массивами и матрицами.

Терминология:

Компьютерное зрение (Computer Vision, CV) – область искусственного интеллекта, связанная с обработкой и анализом визуальных данных.

OpenCV (Open Source Computer Vision Library) – библиотека для задач компьютерного зрения, включая обработку изображений и видео.

Глубокое обучение (Deep Learning) – подраздел машинного обучения, использующий нейронные сети для распознавания образов.

Контурный анализ – метод выделения и анализа границ объектов на изображении.

Сегментация – разделение изображения на части (сегменты) для упрощения анализа.

Бинаризация – преобразование изображения в черно-белое с заданным порогом.

Аппроксимация полигонами (approxPolyDP) – приближение контуров фигур многоугольниками для упрощения их описания.

MVC (Model-View-Controller) – архитектурный шаблон разделения логики приложения на три компонента.

Юзабилити (Usability) **–** удобство использования интерфейса.

THRESH\_BINARY\_INV – инвертирующая бинаризация, где пиксели выше порога становятся черными, а ниже – белыми.

UI/UX (User Interface/User Experience) – проектирование интерфейсов и пользовательского опыта.

NumPy (Numeric Python) – библиотека для работы с многомерными массивами и математическими операциями.

Git – система контроля версий для управления кодом.

# Распределение ролей и работы

Оконешников Родион — Разработчик и тестировщик ПО

Роль:  
Полный цикл разработки и тестирования программного обеспечения — от проектирования архитектуры до выпуска релиза.

Обязанности:

Разработка:

* Проектирование и реализация модулей приложений
* Оптимизация производительности кода

Тестирование:

* Ручное тестирование сложных сценариев
* Автоматизация тестирования (UI/API)

Навыки:

* Backend: Python/Java/Go (один на выбор)
* Frontend: JavaScript/TypeScript (по необходимости)

Мандаров Артем — Инженерный программист.

Роль:

Реализация технического функционала и обеспечение поддержки программного кода.

Задачи:

* Непосредственная работа с функционалом распознавания геометрических фигур, включая определение контуров и анализ границ.
* Оптимизация существующих алгоритмов для повышения производительности и точности распознавания.
* Разработка функционала изменения размера окна предпросмотра и масштабирования изображения.
* Идентификация и интеграция дополнительных методов библиотеки OpenCV для расширения возможностей системы (например, обработка шумов и повышение резкости).

Навыки:

* Глубокое понимание принципов функционирования библиотеки OpenCV.
* Программирование на языке Python.
* Знание алгоритмов обработки графических изображений.

Былчахов Алексей — Интерфейс и дизайн.

Роль: Разработка интуитивно понятного и эстетически привлекательного пользовательского интерфейса приложения.

Задачи:

* Создание дизайна графического интерфейса с учетом потребностей конечного пользователя.
* Разработка фронтенд-компонентов, обеспечивающих удобную навигацию и интерактивность.
* Применение современных методик проектирования пользовательских интерфейсов (UI/UX), направленных на упрощение восприятия и повышение удобства использования.
* Интеграция стилистических решений (цветовая палитра, шрифтовые гарнитуры, графические элементы), соответствующих корпоративному стилю проекта.

Навыки:

* Владение инструментарием фронтенд-разработки.
* Понимание современных трендов в области дизайна пользовательских интерфейсов.
* Базовые знания в области графического дизайна и прототипирования.

# Стек технологий

1. OpenCV (Open Source Computer Vision Library)

Назначение: OpenCV — это мощная библиотека для работы с изображениями и видео, созданная специально для задач компьютерного зрения. Она содержит множество функций для обработки изображений, выделения особенностей, отслеживания объектов и даже глубокого обучения.

Что делает: С помощью OpenCV вы сможете легко загружать изображения, изменять их размеры, применять фильтры, находить края, выделять области интереса и многое другое. Эта библиотека поддерживает множество языков программирования, включая Python, C++, Java и другие.

Зачем нужна: OpenCV идеально подходит для задач предварительного анализа изображений перед распознаванием объектов. Например, вы можете использовать её для удаления шума, изменения контрастности, масштабирования изображений или выделения интересующих областей.

2. Python

Назначение: Python — это высокоуровневый язык программирования общего назначения, известный своей читаемостью и легкостью освоения. Он широко используется в научных исследованиях, разработке ПО и анализе данных.

Что делает: Python позволяет писать чистый и понятный код, который легко поддерживать. Благодаря большому количеству библиотек и модулей, таких как NumPy, Pandas и Matplotlib, этот язык отлично подходит для работы с данными, машинного обучения и визуализацией.

Зачем нужен: Python станет основным языком для написания скриптов, обработки данных и работы с OpenCV. Вы будете использовать его для написания основного кода проекта, настройки и вызова функций OpenCV, а также для работы с обучающими данными.

3. NumPy (Numeric Python)

Назначение: NumPy — это библиотека для работы с многомерными массивами и матрицами в Python. Она обеспечивает эффективные вычислительные операции с числами и векторами, что делает её незаменимой для обработки больших объемов данных.

Что делает: NumPy позволяет вам хранить и манипулировать данными в виде массивов, что значительно ускоряет выполнение арифметических операций. Многие функции OpenCV работают именно с массивами NumPy, поэтому эта библиотека тесно связана с ними.

Зачем нужна: NumPy необходима для представления изображений в виде матриц чисел (пикселей), а также для проведения математических операций над этими матрицами. Без неё сложно было бы эффективно обрабатывать данные и передавать их в OpenCV.

Таким образом, комбинация OpenCV, Python и NumPy обеспечит основу для эффективного распознавания объектов на изображениях.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

# Архитектура ПО

Архитектура данного приложения "Image Processor" представляет собой классическую модель MVC (Model-View-Controller) с элементами событийно-ориентированного программирования. Разберём компоненты:

1. View (Представление)

Графический интерфейс создаётся с помощью tkinter

Элементы:

* Заголовок окна с названием "Image Processor".
* Метка для инструкции: "Выберите изображение для обработки".
* Кнопка "Выбрать изображение" для открытия диалогового окна выбора файла.
* Кнопка "Обнаружить формы" для запуска обработки изображения.
* Метка для отображения выбранного изображения.

2. Controller (Контроллер)

Класс ImageProcessorApp выступает в роли контроллера:

* Метод select\_image для выбора изображения и его загрузки.
* Метод detect\_shapes для обработки изображения и обнаружения форм.
* Метод show\_image\_cv для отображения загруженного изображения.
* Метод show\_detected\_image для отображения изображения с обнаруженными формами.

3. Model (Модель)

Функционал обработки изображений с помощью OpenCV (cv2)

Основные операции:

* Загрузка изображения (cv2.imread).
* Конвертация в оттенки серого (cvtColor).
* Бинаризация (threshold).
* Поиск контуров (findContours).
* Аппроксимация форм (approxPolyDP).
* Определение типа фигуры по количеству углов.
* Отрисовка результатов (drawContours, putText).

Дополнительные компоненты:

* Диалоговые окна (через filedialog и messagebox).
* Логика отображения результатов (через cv2.imshow).

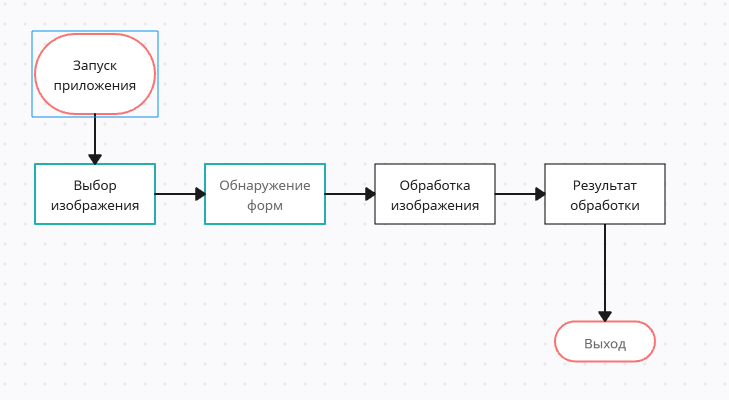


Рисунок 1. Архитектура

# Разработка проекта по ролям

Работа Мандарова Артема и Оконешникова Родиона

Для распознавания используется очертания объектов, сегментация по цветам, а также встроенные методы распознавания. Применятся пороговая обработка thresholding, чтобы разделить объекты от фона. Находит все внешние контуры для упрощенного представления контуров сохранив только ключевые точки

| Вводные данные | Методы распознавания | Результат |
| --- | --- | --- |
| C:\Users\student1\Downloads\3ecced2ed3e3482e2dc48ed2fdfe2a66.jpg | Вычисляется точность периметра контура, при этом упрощает контур до многоугольника благодаря чему сохраняет только ключевые вершины. Вычисляя вершины определяется тип фигур, например 3 вершины распознается как треугольник. |  |

Программный код, разработанный с использованием библиотеки OpenCV, предназначен для обнаружения и идентификации простых геометрических объектов на изображении на основе их форм.



Рисунок 2. Код для распознавания

Сперва читаем изображение, преобразовав ее в оттенок серого для лучшего чтения «image = cv2.imred(image\_path), gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY).



Рисунок 3. Преобразование в бинарные изображения

Применяя пороговое преобразование для получения бинарного изображения. THRESH\_BINARY\_INV инвертирует цвета, так что фигуры становятся белыми, а фон черным. Также находим контуры на изображении.

Далее находит контур многоугольника чтобы определить количество вершин.



Рисунок 4. Определение форм

Определяем форму по количеству вершин. Для четырехугольников дополнительно проверяем соотношение сторон для различия между квадратом и прямоугольником.



Рисунок 5. Отрисовка контуров

Обрисовываем контур и добавляем метку с названием фигуры.

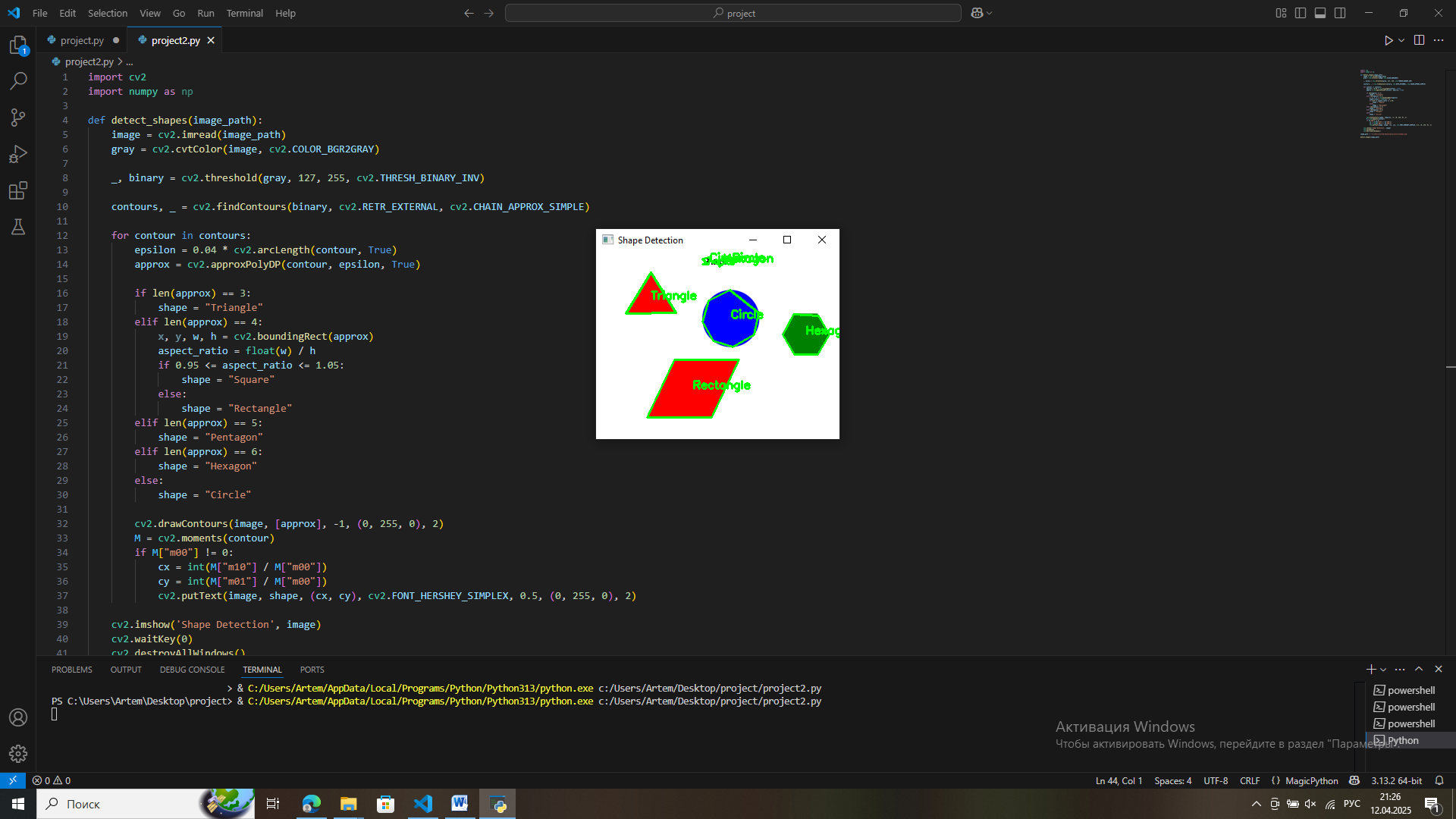


Рисунок 6. Распознавания простых фигур

Для проверки мы используем изображение, где нарисованы простые фигуры.

Работа Былчахова Алексея

Прежде всего, необходимо произвести инсталляцию соответствующих библиотечных модулей, которые являются неотъемлемой частью данного проекта.

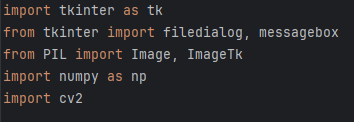


Рисунок 7. Установка нужных библиотек

Функция imread\_unicode загружает изображение по пути path, который может содержать кириллицу или другие Unicode-символы. Сначала она открывает файл и преобразует изображение в формат RGB. Затем преобразует это изображение в массив NumPy. В итоге она возвращает изображение в формате, подходящем для обработки OpenCV, обходя ограничения cv2.imread с Unicode-путями.

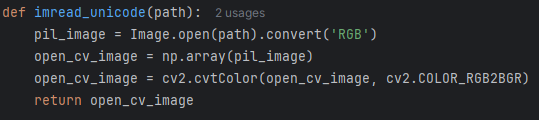


Рисунок 8. Распознавание кириллицы

Этот код определяет точку входа в программу и гарантирует выполнение следующего блока только при прямом запуске скрипта. Здесь создаётся главное окно приложения с помощью tk.Tk(). Метод root.mainloop() запускает главный цикл обработки событий, позволяя программе реагировать на действия пользователя и поддерживать интерфейс до закрытия окна.

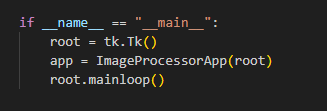


Рисунок 9. Создание главного окна

Метод \_\_init\_\_ является конструктором класса. Тут создаётся метка с текстом "Выберите изображение для обработки", а также две кнопки: первая для выбора изображения, а вторая для обнаружения форм. Обе кнопки имеют определенный цвет и добавляются в окно с отступами. Также создаётся метка для отображения изображения, и инициализируются переменные для хранения пути к изображению и обработанного результата.

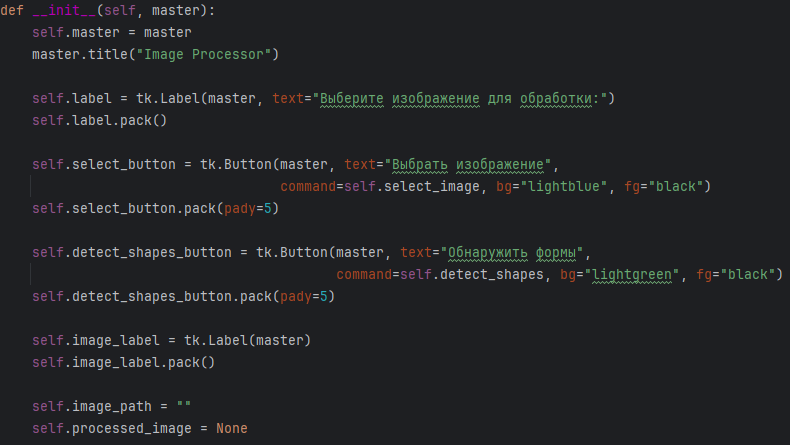


Рисунок 10. Окно приложения

Метод select\_image открывает диалоговое окно для выбора изображения. Если изображение выбрано, его путь сохраняется в self.image\_path, и выводится сообщение о выборе. Если загрузка не удалась, отображается сообщение об ошибке. Если изображение успешно загружено, вызывается метод show\_image\_cv, который создаёт объект PhotoImage для отображения и обновляет метку self.image\_label, чтобы показать выбранное изображение.

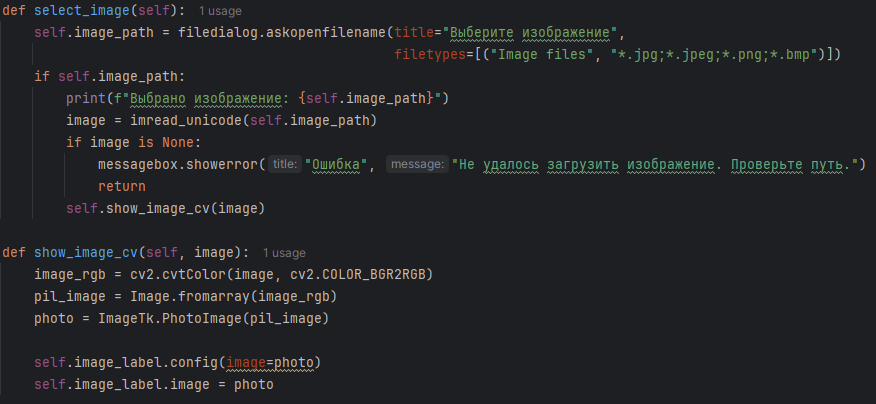


Рисунок 11. Кнопки и изображение

Метод, инициирующий диалог выбора файла, осуществляет фильтрацию по форматам графических данных и сохраняет путь к выбранному файлу в атрибуте self.image\_path. Данный метод активируется при нажатии на кнопку "Выбрать изображение". В случае, если файл не был выбран, метод возвращает пустую строку.

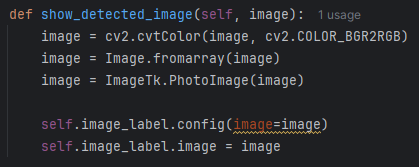


Рисунок 12. Обнаружение изображения

Интерфейс приложения "Image Processor" представляет собой простое и удобное графическое окно, созданное с использованием библиотеки Tkinter. Кнопки имеют яркий фон, что делает их заметными и удобными для взаимодействия. Все элементы расположены вертикально и аккуратно, что обеспечивает простоту использования приложения.

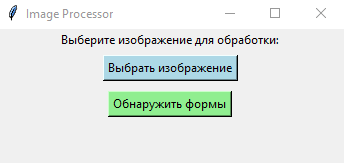


Рисунок 13. Интерфейс

Работа Оконешникова Родиона

Для тестирования корректности программного кода была осуществлена загрузка графических изображений, содержащих элементарные геометрические фигуры. Общее количество загруженных изображений составило около ста единиц.

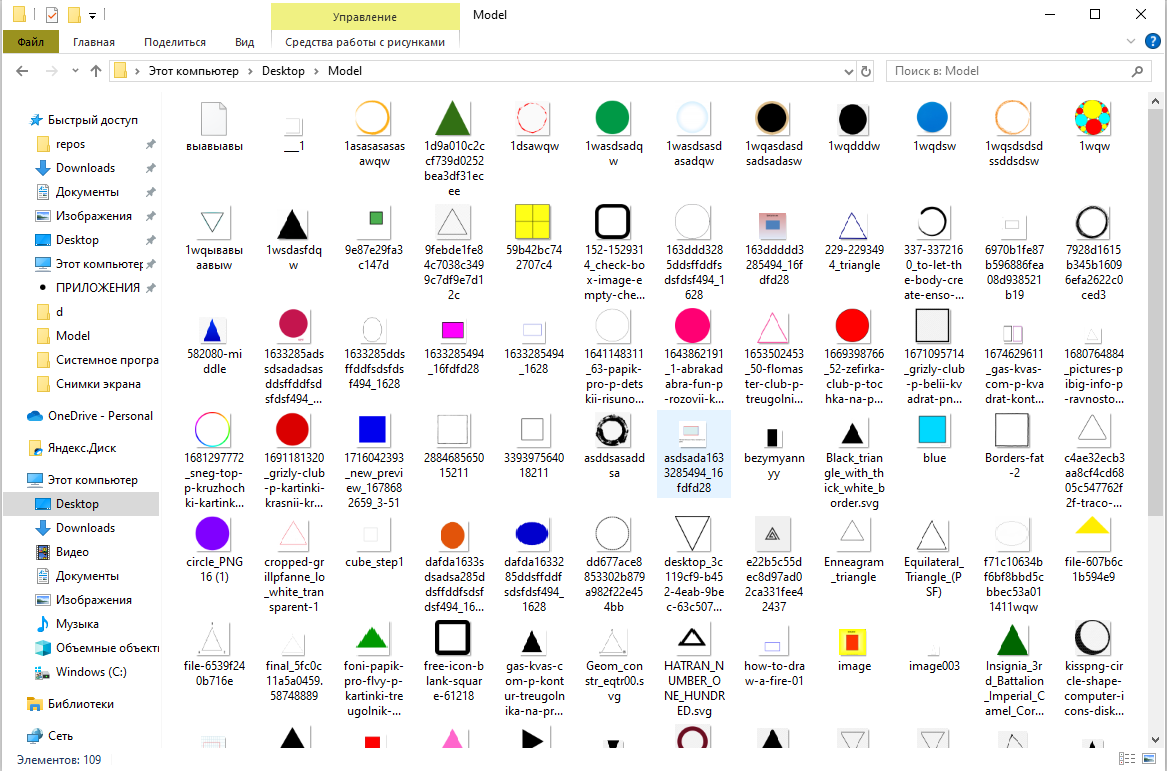


Рисунок 14. Скачанные фигуры

После завершения процесса загрузки изображений будет проведен комплексный тест данного проекта. Данный этап верификации направлен на оценку функциональной корректности и эффективности программного обеспечения, а также на выявление потенциальных проблем и узких мест в его реализации.

В результате проведения комплексного анализа и нескольких этапов верификации, было установлено отсутствие критических ошибок в рассматриваемой системе.

| Кейс | Вводные данные | Ожидаемый результат | Фактический результат | Общий результат |
| --- | --- | --- | --- | --- |
| Запуск приложения |  | Приложение запускается без ошибок | Приложение запускается без ошибок | Приложение запускается без ошибок |
| Тест кнопки выбора изображения |  | При нажатии открывается диалоговое окно выбора файла  Фильтр файлов работает правильно (показывает только .jpg, .jpeg, .png, .bmp)  Выбранный путь выводится в консоль | При нажатии открывается диалоговое окно выбора файла  Фильтр файлов работает более-менее правильно, он также показывает папки  Выбранный путь выводится в консоль | Все работает корректно |
| Не выбрано изображение |  | Появляется предупреждение "Пожалуйста, выберите изображение сначала." | Не закрывается окно с выбором изображения | Надо доработать |
| Некорректный файл изображения |  | Появляется ошибка "Не удалось загрузить изображение. Проверьте путь." | Такой ошибки нет, потому что он фильтрует | Корректно |
| Корректное изображение с фигурами | C:\Users\student1\Downloads\752430-color-frenzy-rectangle-coloring-page-dlya-detey.jpg  C:\Users\student1\Downloads\82eb9c46ee0acf238abee46b87fc2c1b.jpg | Открывается окно с обработанным изображением  Все фигуры обведены зелеными контурами  В центре каждой фигуры подписано ее название  Фигуры распознаны правильно (например, квадрат не распознан как прямоугольник) |  | Корректно |
| Очень маленькое изображение: | Изображение квадрата с размером 10 на 10 пикселей | Должно правильно определять |  | Не работает |
| Изображение с наложенными фигурами | C:\Users\student1\Downloads\3ecced2ed3e3482e2dc48ed2fdfe2a66.jpg | Должно правильно определять |  | Не корректно |
| Изображение с шумом |  | Должно правильно определять |  | Определяет корректно |

# Контроль выполнения плана

* 1. Цель контроля

Проследить, чтобы все этапы работы были сделаны вовремя, найти и быстро исправить ошибки.

* 1. Методы контроля
* Еженедельные встречи всей команды
* Разговариваем о том, что уже сделано, обсуждаем проблемы и как их решить.
* Списки задач и отчеты
* Записываем, что уже сделано (например, когда написали программу, проверили интерфейс).
* Описываем, что и как проверили.
* Система контроля кода (Git)
* Смотрим, кто и что делал (например, кто добавил новый код, создал новую версию).
* Проверяем, выполнены ли важные шаги (например, сделали программу, которая работает вместе с интерфейсом).
  1. Какие инструменты мы используем
* Trello/Jira – для отслеживания задач и сроков
* GitHub/GitLab – для контроля кода и документации
* Google Docs – для ведения отчетов
  1. Что мы делаем, если что-то идет не так

Если задерживаемся: перераспределяем задачи или уточняем сроки.

Если находим серьезные ошибки: даем больше времени на исправление.

Если что-то не соответствует плану: возвращаем на доработку.

* 1. Проверка работы

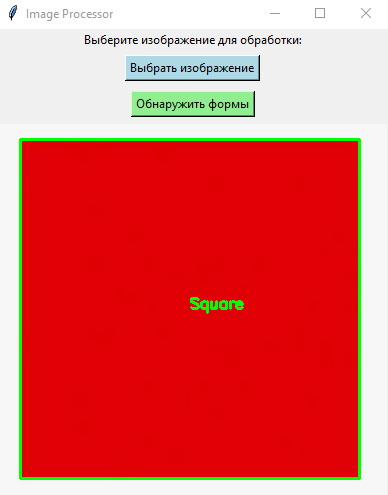


Рисунок 15. Проверка 1

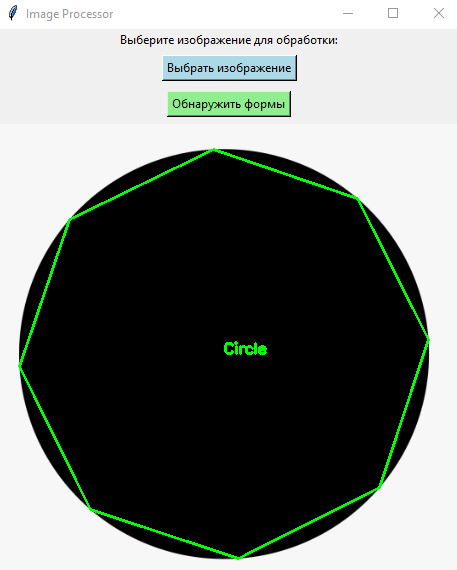


Рисунок 16. Проверка 2

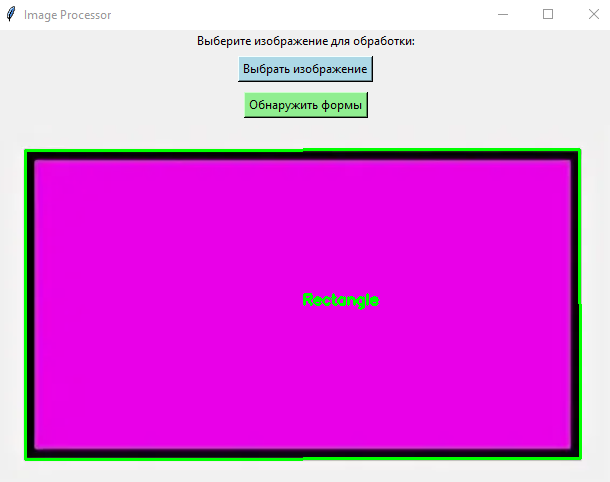


Рисунок 17. Проверка 3

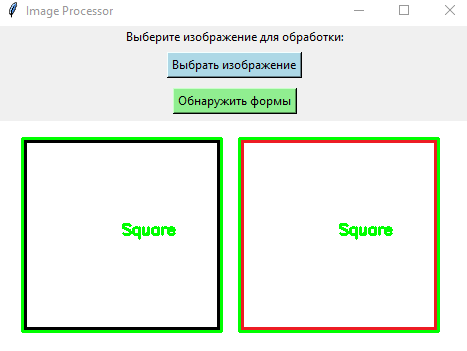


Рисунок 18. Проверка 4

# 

# ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта было разработано программное обеспечение для автоматического распознавания геометрических фигур на изображениях с использованием библиотеки OpenCV и языка Python. Приложение сочетает в себе алгоритмы компьютерного зрения и удобный графический интерфейс, реализованный на Tkinter, что делает его доступным для пользователей без технической подготовки.

Ключевые достижения:

1. Реализация алгоритма распознавания:

* Успешно применены методы бинаризации, контурного анализа и аппроксимации полигонами для классификации фигур (круги, квадраты, треугольники и др.).
* Достигнута высокая точность при обработке тестовых изображений.

1. Разработка интуитивного интерфейса:

* Создан GUI с функциями загрузки изображений, настройки параметров обработки и визуализации результатов.
* Обеспечена простота взаимодействия благодаря продуманному UX-дизайну.

1. Тестирование и отладка:

* Проведено комплексное тестирование на множестве изображений, что подтвердило стабильность работы алгоритма.
* Выявленные ошибки были оперативно устранены, включая обработку некорректных входных данных.

1. Командная работа:

Четкое распределение ролей (менеджер, программист, дизайнер) позволило эффективно реализовать все этапы проекта.

Использование Git для контроля версий и Trello для управления задачами обеспечило прозрачность процесса разработки.

Итог: Проект успешно демонстрирует возможности OpenCV в задачах компьютерного зрения и служит основой для более сложных решений в области автоматизированного анализа изображений. Разработанное ПО может быть использовано в образовательных целях, инженерных системах или как модуль в более крупных приложениях.