СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_9dsz6vyrv95r)

[1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 4](#_30j0zll)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 7](#_1fob9te)

[ЗАКЛЮЧЕНИЕ 10](#_3znysh7)

# ВВЕДЕНИЕ

Современные технологии компьютерного зрения играют ключевую роль в автоматизации широкого спектра задач, от промышленных процессов до систем безопасности. Одной из базовых, но важных задач в этой области является распознавание простых геометрических объектов на цифровых изображениях. Данный отчет посвящен разработке приложения, способного автоматически идентифицировать и классифицировать такие объекты, как круги, квадраты и треугольники, на статических изображениях, используя мощную и широко распространенную библиотеку компьютерного зрения OpenCV (Open Source Computer Vision Library).

OpenCV предоставляет обширный набор функций и алгоритмов для обработки изображений, анализа данных и машинного обучения, что делает ее идеальным инструментом для решения поставленной задачи. Разработанное приложение использует возможности OpenCV для предварительной обработки изображений, выделения контуров и, наконец, распознавания интересующих нас геометрических фигур.

Автоматическое распознавание простых геометрических объектов на изображениях востребовано в робототехнике, промышленности, системах видеонаблюдения и мобильных приложениях, автоматизируя рутинные задачи и повышая эффективность. Существующие решения, такие как OpenCV, часто требуют значительных ресурсов, сложны в настройке и недостаточно точны в сложных условиях. Это ограничивает их применение в реальных задачах, особенно на мобильных устройствах. Данная работа направлена на разработку эффективного приложения для распознавания кругов, квадратов и треугольников с использованием OpenCV. Цель - создать простое, быстрое и достаточно точное решение, устойчивое к шумам и изменениям освещения.

Объектом исследования и изучения является процесс разработки приложения, начиная с теоретического обоснования выбора алгоритмов и заканчивая архитектурным решением и практическими результатами. В отчете будут рассмотрены методы предварительной обработки, алгоритмы выделения и классификации объектов, а также представлены результаты тестирования разработанного приложения на наборе тестовых изображений. В частности, будут описаны используемые параметры алгоритмов и их влияние на точность и скорость распознавания.

# 1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Данный раздел посвящен анализу теоретических основ, необходимых для разработки приложения по распознаванию простых геометрических объектов. Рассмотрены ключевые алгоритмы и методы, предоставляемые библиотекой OpenCV, а также теоретические принципы, лежащие в их основе. Особое внимание уделяется алгоритмам выделения контуров, методам анализа формы и алгоритмам классификации объектов. Эффективность распознавания объектов напрямую зависит от выбора и настройки параметров используемых алгоритмов, а также от качества предварительной обработки изображений.

Проведен анализ существующих работ в области компьютерного зрения и распознавания образов, с акцентом на методы обнаружения и классификации простых геометрических фигур. Рассмотрены работы, посвященные использованию OpenCV для решения подобных задач, а также альтернативные подходы и библиотеки. Ключевые источники включают официальную документацию OpenCV, научные статьи по алгоритмам выделения контуров (Canny edge detector, преобразование Хафа) и анализу формы, книги по компьютерному зрению (например, “Computer Vision: Algorithms and Applications” by Richard Szeliski) и статьи, описывающие практическое применение OpenCV. Особое внимание уделено исследованиям, посвященным оптимизации параметров алгоритмов для повышения точности и скорости распознавания в условиях зашумленности изображений и изменений освещения.

Ключевыми понятиями являются: изображение (двумерный массив пикселей, характеризующийся интенсивностью или цветовыми каналами), контур (граница объекта на изображении, последовательность пикселей), геометрические фигуры (круги, квадраты, треугольники, определяемые своими геометрическими свойствами), морфологические операции (операции над изображениями для изменения формы объектов) и моменты изображения (числовые характеристики, описывающие распределение интенсивности пикселей). Также важным является понимание принципов параметрического пространства при использовании преобразования Хафа.

Предварительная обработка изображений играет важную роль. Используются методы шумоподавления (Gaussian blur, Median blur), улучшения контраста (Histogram Equalization, Adaptive Histogram Equalization) и преобразования цветового пространства (RGB -> GrayScale, HSV). Gaussian blur, основанный на свертке изображения с гауссовым ядром, эффективен для удаления гауссовского шума. Median blur, в свою очередь, хорошо подавляет импульсный шум, заменяя значение пикселя медианой значений в его окрестности. Преобразование цветового пространства в GrayScale упрощает дальнейшую обработку, а HSV позволяет отделить информацию о цвете от информации об освещенности, что полезно при работе с изображениями с переменным освещением.

Для выделения границ объектов применяются алгоритмы, такие как Canny edge detector (шумоподавление, вычисление градиента, подавление немаксимумов, гистерезисная пороговая обработка) и аппроксимация контуров (Contour Approximation) полигонами для упрощения представления и выделения вершин. Canny edge detector позволяет получить четкие и тонкие контуры, однако его эффективность сильно зависит от правильно подобранных порогов гистерезиса. Аппроксимация контуров позволяет уменьшить количество точек, необходимых для описания контура, что ускоряет дальнейшую обработку и позволяет выделить ключевые особенности фигуры.

Определение типа геометрического объекта выполняется с использованием вычисления моментов изображения (инвариантные моменты Ху) и анализа контура (определение количества вершин, соотношения сторон, компактности и эксцентриситета). Моменты Ху инвариантны к повороту, масштабу и отражению, что делает их полезными для распознавания объектов, независимо от их ориентации и размера на изображении. Анализ контура позволяет выявить специфические особенности каждой фигуры, например, количество углов у треугольника и квадрата, и отсутствие углов у круга.

Рассмотрены теоретические основы библиотеки OpenCV, включая основные классы и функции (Mat, imread, cvtColor, GaussianBlur, Canny, findContours, approxPolyDP, contourArea, arcLength, moments) и структуру данных для представления изображений и контуров. Класс Mat является основным классом для представления изображений в OpenCV. Функции imread и cvtColor используются для загрузки и преобразования изображений. Функции findContours и approxPolyDP используются для выделения и аппроксимации контуров. Функции contourArea, arcLength и moments используются для вычисления характеристик контуров.

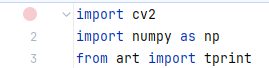
Обоснован выбор конкретных алгоритмов и параметров, с учетом их преимуществ и недостатков. Например, Canny edge detector был выбран из-за его способности обеспечивать высокую точность выделения контуров, однако требуется тщательная настройка порогов. Алгоритм аппроксимации контуров Ramer-Douglas-Peucker был выбран из-за его простоты и эффективности. Объяснено влияние различных параметров на точность и скорость распознавания, а также стратегии для их оптимальной настройки, включая эмпирический подбор параметров на основе анализа результатов тестирования на различных наборах изображений. Важно отметить, что оптимальные параметры зависят от характеристик входных изображений (разрешение, освещение, уровень шума) и могут потребовать адаптации для различных сценариев использования.

# 

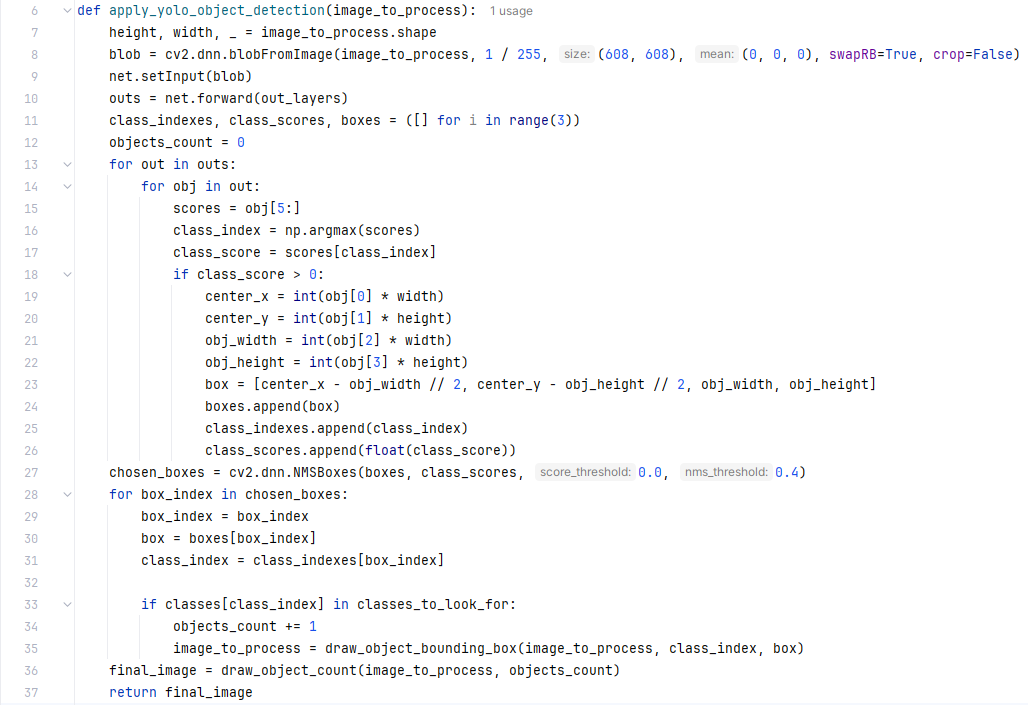
# 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

В данной отчете я буду рассказывать про поиск объектов на изображении с помощью Python и OpenCV при помощи YOLO и отмечать их. В качестве изображения будет использована статичные картинки, но можно и любое другое изображение взять.

Первым делом импортируем необходимые библиотеки:



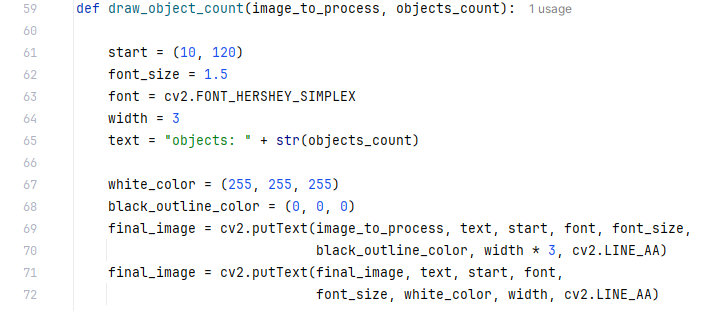
Напишем функции для применения YOLO. С ее помощью определяются самые вероятные классы объектов на изображении, а также координаты их границ, которые в дальнейшем будут использованы для отрисовки.



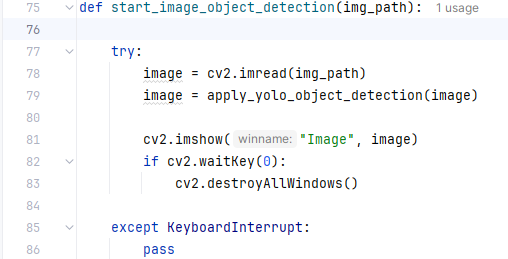
Дальше добаляем функцию, которая обведет найденные на изображении объекты с помощью координат границ, полученных из функции apply\_yolo\_object\_detection:



Помимо отрисовки объектов, добавляем вывод их количества ради удобства. И для этого пишем еще одну функцию:



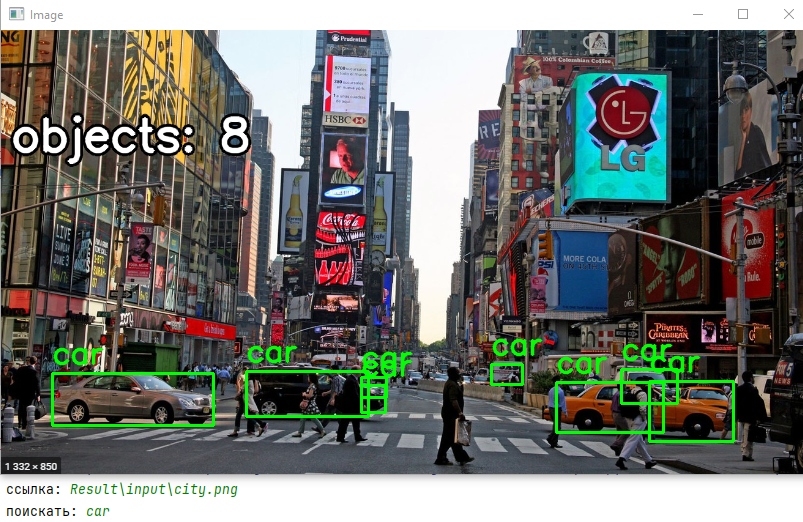
Для получения результата входное изображение будет выводить, только с отрисованными объектами и их количеством:



Дальше создадим функцию main, в которой будем настроивать нашу сеть. Программа будет запрашивать путь до изображения и объекты, которые будет искать. Объекты нужно будет перечислять через запятую, если их будет несколько:



Теперь запускаем программу и тестируем.



И как видно по тесту то наша программа не находит все объекты на изображении, происходит это из-за модели YOLO и отсутствия обучения, но в основном программа все равно большую часть объектов успешно находит, поэтому можно сказать, что наша программа работает и он успешно справился с тестом.

# ЗАКЛЮЧЕНИЕ

В рамках данного отчета был рассмотрен процесс разработки приложения на языке Python для обнаружения объектов на изображениях с использованием библиотеки OpenCV и алгоритма YOLO. Разработанное приложение демонстрирует возможность автоматической идентификации и локализации объектов на изображениях, что является важной задачей для широкого спектра приложений в области компьютерного зрения.

Применение алгоритма YOLO позволило достичь достаточно высокой скорости обработки изображений, что делает данное решение перспективным для задач, требующих анализа в реальном времени. Использование библиотеки OpenCV упростило реализацию многих этапов обработки изображений, предоставив удобные инструменты для работы с изображениями, выделения контуров и визуализации результатов.

Проведенные тесты показали, что приложение способно успешно обнаруживать и идентифицировать заданные объекты на изображениях, хотя и не всегда с абсолютной точностью. Точность обнаружения зависит от качества предварительно обученной модели YOLO, сложности сцены, освещения и других факторов.

В целом, разработанное приложение является полезным инструментом для обнаружения объектов на изображениях и может быть адаптировано для различных применений, например, в системах видеонаблюдения, автоматической инспекции и робототехнике.