Оглавление

[ВВЕДНИЕ 3](#_Toc187506806)

[1. Общая информация о приложении 4](#_Toc187506807)

[1.1. Цель и задачи приложения 4](#_Toc187506808)

[1.2. Библиотеки приложения 5](#_Toc187506809)

[1.3. Области применения приложения 6](#_Toc187506810)

[2. Разработка 8](#_Toc187506811)

[2.1 Реализация функционала приложения 8](#_Toc187506812)

[2.2 Пользовательский сценарий 14](#_Toc187506813)

[ЗАКЛЮЧЕНИЕ 18](#_Toc187506814)

[Список использованных источников. 19](#_Toc187506815)

**ВВЕДНИЕ**

Современные технологии обработки изображений и компьютерного зрения находят широкое применение в различных областях, таких как безопасность, транспорт, розничная торговля и многие другие. Одним из наиболее перспективных направлений является использование алгоритмов для распознавания объектов в изображениях и видео, что позволяет автоматизировать процессы анализа и принятия решений.

В рамках данной работы представлен инструмент, предназначенный для обработки изображений с целью распознавания автомобилей, а также извлечения доминирующих цветов, присутствующих на изображениях. Основной задачей является использование модели YOLO (You Only Look Once) для обнаружения автомобилей (машин, грузовиков и автобусов) на изображениях. Дополнительно, для анализа цветовых характеристик объектов применяется метод кластеризации K-Means в цветовом пространстве HSV, что повышает стабильность извлечения цвета при различных условиях освещения.

1. **Общая информация о приложении**
   1. **Цель и задачи приложения**

Целью данного приложения является разработка инструмента для автоматизированного анализа изображений с использованием технологий компьютерного зрения. Программа предназначена для обнаружения автомобилей (включая легковые машины, грузовики и автобусы) на изображениях, извлечения доминирующих цветов из областей с автомобилями, а также предоставления пользователю удобного интерфейса для работы с результатами анализа.

Для достижения поставленной цели решаются следующие задачи:

1. Обнаружение объектов на изображениях

Использование модели YOLOv5 для точного распознавания автомобилей на загруженных изображениях.

Определение координат объектов на изображении (рамки объектов) с учетом классов "машина", "грузовик" и "автобус".

1. Извлечение доминирующих цветов

Разработка алгоритма для анализа цветовых характеристик объектов, выделенных на изображении.

Применение метода кластеризации K-Means для извлечения доминирующих цветов в цветовой модели HSV, устойчивой к различным условиям освещения.

1. Графический интерфейс пользователя (GUI)

Создание удобного интерфейса с помощью библиотеки PyQt6 для взаимодействия с пользователем.

Реализация возможности выбора директории с изображениями, отображения детализированной информации о каждом изображении, включая разрешение, размер файла и обнаруженные объекты.

Визуализация доминирующих цветов в виде цветных квадратов, что улучшает восприятие результатов анализа.

1. Обработка и сохранение данных

Организация сохранения обработанных изображений с автомобилями в указанной директории.

Реализация CSV-отчета с информацией об обнаруженных объектах и результатах анализа.

1. Обеспечение производительности и совместимости

Оптимизация работы приложения с использованием GPU (если доступно) для ускорения анализа изображений.

Поддержка различных форматов изображений и обеспечение совместимости с большинством систем.

1. Устранение ошибок и уведомления пользователя

Обработка исключений для предотвращения сбоев в работе приложения.

Реализация системы уведомлений для информирования пользователя о ходе обработки, успешных операциях и возникших ошибках.

* 1. **Библиотеки приложения**

PyQt - это библиотека, которая помогает делать кросс-платформенные приложения с графическим интерфейсом. Данная библиотека имеет обширный функционал, множество методов и форм.

PyTorch - это фреймворк глубокого обучения с открытым исходным кодом, разработанный группой исследователей из Facebook's AI Research lab (FAIR). PyTorch предоставляет инструменты для создания и обучения нейронных сетей.

YOLOv5 — это последняя версия семейства алгоритмов объектного обнаружения, разработанных для эффективного и быстрого обнаружения объектов на изображениях и в видеопотоках. YOLOv5 представляет собой значительное улучшение по сравнению с предыдущими версиями, в частности, по скорости и точности.

OpenCV — это open source библиотека компьютерного зрения, которая предназначена для работы с изображениями. С её помощью можно анализировать, считывать, обрабатывать изображения.

Scikit-learn — это библиотека для языка программирования Python, предоставляющая простой и эффективный инструментарий для анализа данных и машинного обучения. Она стремится предоставить простой и единый интерфейс для различных алгоритмов машинного обучения, что делает ее полезной для быстрого прототипирования и разработки в области анализа данных и машинного обучения.

Python Imaging Library — библиотека языка Python, предназначенная для работы с растровой графикой. PIL — это оригинальная библиотека Python для работы с изображениями.

Numpy - мощная библиотека для работы с многомерными массивами и выполнения численных вычислений. Используется для обработки и анализа данных изображений.

Csv – данная библиотека помогает с работой над файлами с расширением .csv необходима, чтобы записывать данные с результатом.

* 1. **Области применения приложения**

Анализ изображений в транспортной сфере

Приложение может быть использовано для автоматического обнаружения автомобилей, грузовиков и автобусов на изображениях. Это полезно для:

1. Мониторинга дорожной ситуации.
2. Контроля парковочных зон.
3. Анализа трафика и плотности движения.

Обработка данных для исследований

Инструмент позволяет классифицировать изображения, извлекать данные об их размерах, доминирующих цветах и сохранять их в удобном формате. Это может быть полезно для анализа данных в исследованиях, связанных с транспортом, урбанистикой.

1. **Разработка**
   1. **Реализация функционала приложения**

Данное приложение состоит из двух окон, которых имеется своя реализация. Test2Window – главное окно данного приложения, в нем выводятся списки с изображениями автомобилей, а также расположены кнопки для анализа изображения с машиной. ModalWindow – второе окно приложения. В данном окне выводится информация о выбранном автомобиле, а именно изображение, отметка автомобиля на изображении и вывод доминирующих цветов на данном изображении.

Для реализации работы окна Test2Window были реализованы следующие методы.

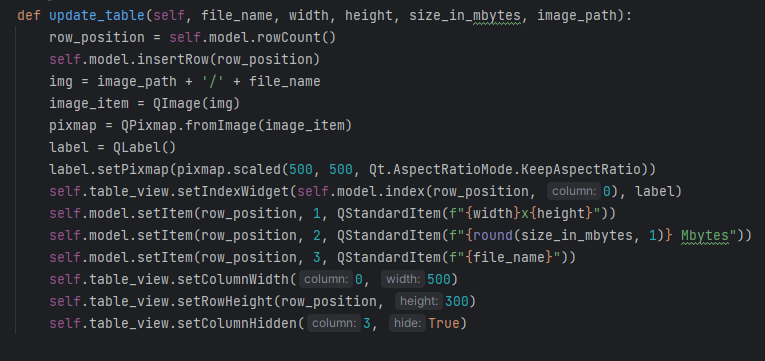


Рисунок 1. Реализация метода update\_table.

Метод update\_table() добавляет новую строку в таблицу, заполняет её изображением, его размером и разрешением, а также скрывает столбец с именем файла. Изображение он получает с помощью пути к изображению, добавляя к этому пути название самого изображения. Затем с помощью класса QImage() строка с путем и названием изображения преобразовывается уже в само изображение, а чтобы вывести изображение в таблицу используется класс QPixmap с методом fromImage(image\_item) с указанием в нем изображения полученного с пути к этому изображению. Это позволяет пользователю просматривать изображения и их характеристики в удобном формате в графическом интерфейсе.

file\_name — имя изображения (файла), которое будет отображено в таблице.

width — ширина изображения (в пикселях).

height — высота изображения (в пикселях).

size\_in\_mbytes — размер изображения в мегабайтах.

image\_path — путь к директории, в которой хранится изображение.

Следующий метод - это метод find\_car().

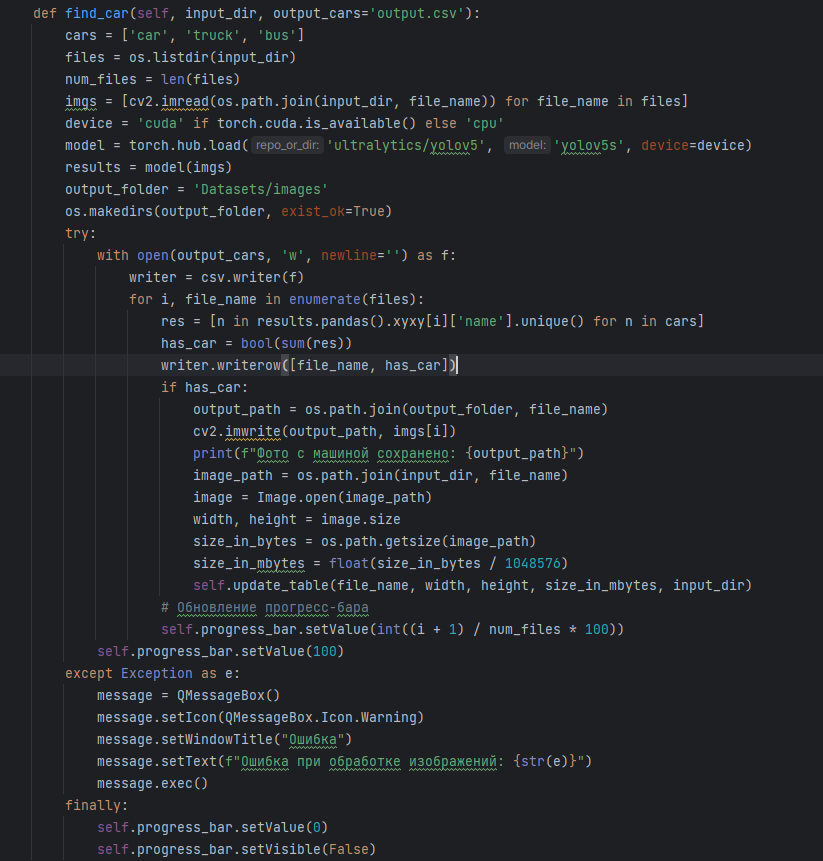


Рисунок 2. Метод find\_car().

Метод find\_car выполняет обработку изображений из указанной директории, применяя модель детекции объектов (YOLOv5) для нахождения автомобилей (машин), грузовиков и автобусов в изображениях. Далее метод сохраняет изображения, содержащие автомобили, в отдельную папку, а также записывает информацию о наличии автомобилей в CSV-файл. Кроме того, он обновляет графический интерфейс с отображением данных о файлах в таблице. Метод содержит в себе массив автомобилей, которые необходимо найти на изображении, данный массив записан в переменную cars. А также список файлов, которые необходимо прочитать с выбранной директории, выбранную директорию передают с помощью переменной input\_dir затем с помощью цикла for в строке:

imgs = [cv2.imread(os.path.join(input\_dir, file\_name)) for file\_name in files].

проходятся по всему списку файлов, полученных из директории. И с помощью подключенной модели YOLOv5 обрабатывает эти изображения на поиск автомобилей, затем открывается папка Datasets/images, в которую записывают файлы изображений, на которых был найден автомобиль. После записи и сохранения файла в папку images, необходимо получить данные об этом изображении, а именно название, разрешение, вес в мегабайтах и папку, в котором хранится изображение. Реализовано это с помощью класса Image библиотеки PIL. После получения всех необходимых данных вызывается метод update\_table(), который добавляет запись об изображение с машиной в таблицу. В данном методе также использовалась конструкция try, except, finally, которая используется для обработок исключений и ошибок, и чтобы программа, продолжила работу, просто выводя об ошибке информацию. Сообщение выводится с помощью элемента интерфейса QMessageBox(), в блоке конструкции except, элемент QMessageBox() является простым модальным окном для вывода информации об ошибке или исключении.

Следующий метод – это метод view\_result().

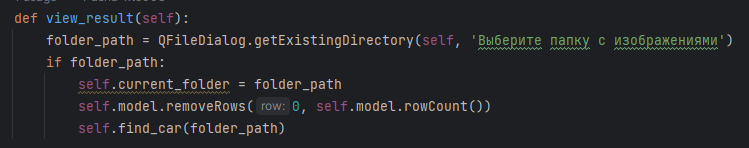


Рисунок 3. Метод view\_result().

Данный метод открывает диалоговое окно в папке, где находится главный класс приложения. Затем после того как пользователь выберет папку, в которой хранятся изображении для проверки, начнет выполняться метод find\_car(), который будет искать на каждом изображении в папке автомобиль.

Для того чтобы выделять цвет автомобиля на изображении реализован метод extract\_colors().

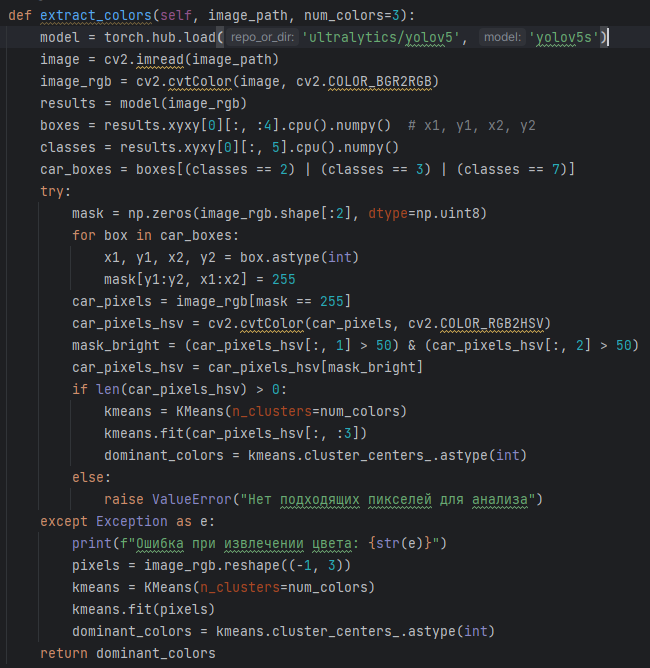


Рисунок 4. Реализация метода extract\_colors().

Метод extract\_colors извлекает доминирующие цвета автомобилей на изображении с использованием детекции объектов и кластеризации цветовых данных. В случае отсутствия подходящих объектов на изображении, метод выполняет кластеризацию всех пикселей изображения. В самом начале в метод подгружается модель YOLOv5. Затем из пути к изображению с автомобилем формируется само изображение с помощью OpenCV и метода cv2.imread(). После преобразуется цветовое пространство из BGR в RGB. Затем модель обнаруживает объекты на изображении и ограничивает эти объекты рамками boxes и классами данных объектов. Затем в переменную car\_boxes извлекаются рамки объектов с указанием классов 2 – машины, 3 – грузовики, 7 – автобусы в модели YOLOv5. Затем создается маска размером, где пиксели, попадающие в рамки объектов-автомобилей, отмечаются как белые (255). С помощью маски выделяются только те пиксели изображения, которые относятся к обнаруженным автомобилям. Затем выполняется кластеризация пикселей. Кластеризация пикселей — это процесс группировки пикселей изображения на основе схожести их характеристик, таких как цвет, интенсивность или текстура. В контексте анализа изображения кластеризация помогает выделить основные группы цветов (или других характеристик), которые представляют доминирующие свойства изображения. Во время выполнения кластеризации определяются центры кластеров, которые предоставляют доминирующие цвета на изображении в HSV (цветовая модель). В случае если обнаружение объектов или фильтрация пикселей не удается (например, на изображении нет автомобилей), то метод выполняет кластеризацию всех пикселей изображения, извлекая доминирующие цвета из него в целом.

Следующий метод занимается выборкой изображения автомобиля из всего списка изображений с машинами по нажатию пользователем кнопки.

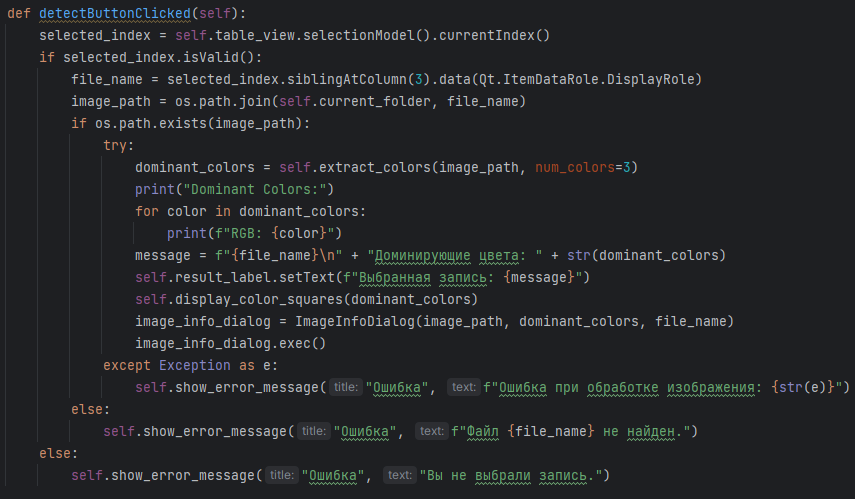


Рисунок 5. Реализация метода detectButtonClicked().

Метод выполняет задачи связанные с обработкой изображений, а именно с помощью этого метода срабатывает метод extract\_colors(), который выбирает доминирующие цвета машины. Работа метода начинается с проверки текущего выбранного элемента в таблице с помощью: selected\_index = self.table\_view.selectionModel().currentIndex().

Если выбранный элемент существует, метод продолжает выполнение. Из выбранной строки извлекает имя файла изображения, а затем строит путь к данному изображению, чтобы проверить существует ли такое изображение в системе, если оно существует, то метод продолжит работать, в обратном случае, пользователю выведется модальное окно с выводом данной ошибки. Затем в переменную dominant\_colors записывается результат метода extract\_colors(). Затем результаты данных цветов выводятся на экран в виде массива RGB цветов. После этого открывается диалоговое окно ModalWindow в котором выводится изображение, на котором в рамке отмечена найденная машина, и доминирующие цвета в отмеченной рамке. В случае же ошибки, пользователю выведется модальное окно с ошибкой в программе, а приложение продолжит работу без завершения.

* 1. **Пользовательский сценарий**

Перед тем как пользователь сможет запустить и полноценно работать в данном приложении, нужно загрузить несколько обязательных библиотек с помощью командной строки.

1. Библиотека PyQt, для установки данной библиотеки необходимо ввести команду $ pip install pyqt6.
2. Библиотека OpenCV, чтобы установить данную библиотеку, необходимо ввести команду $ pip install opencv-python.
3. Библиотека scikit-learn, для загрузки данной библиотеки нужно ввести команду $ pip install scikit-learn.
4. Библиотека PyTorch, чтобы установить данную библиотеку, необходимо ввести команду $ pip install tourch.
5. Библиотека YOLOv5, для установки данной библиотеки нужно ввести команду: $ pip install yolov5.
6. Библиотека ultralytics, для загрузки данной библиотеки необходимо ввести команду $ pip install ultralystics.
7. Библиотека PIL, для установки этой библиотеки нужно ввести команду: $ pip install Pillow.
8. Библиотека numpy, чтобы установить данную библиотеку, необходимо ввести команду $ pip install numpy.

Для удобства навигации по программе для пользователя приведен следующий сценарий:

Извлечение доминирующего цвета изображения с машиной:

Первым что откроется пользователю при запуске приложения – это главное окно, состоящая из таблицы, куда выводятся изображения с машинами, найденные в папке, выбранной пользователем, а также из секции с кнопками «Выбрать директорию», «Посмотреть», «Выход». При запуске приложения изначально таблица будет пустой, чтобы заполнить ее списком из файлов с изображениями, необходимо нажать на кнопку «Выбрать директорию», после чего пользователю откроется диалоговое окно с выбором директорией. После того как пользователь выбрал директорию, где есть файлы с изображением машин, таблица обновится и заполнится данными файлами.

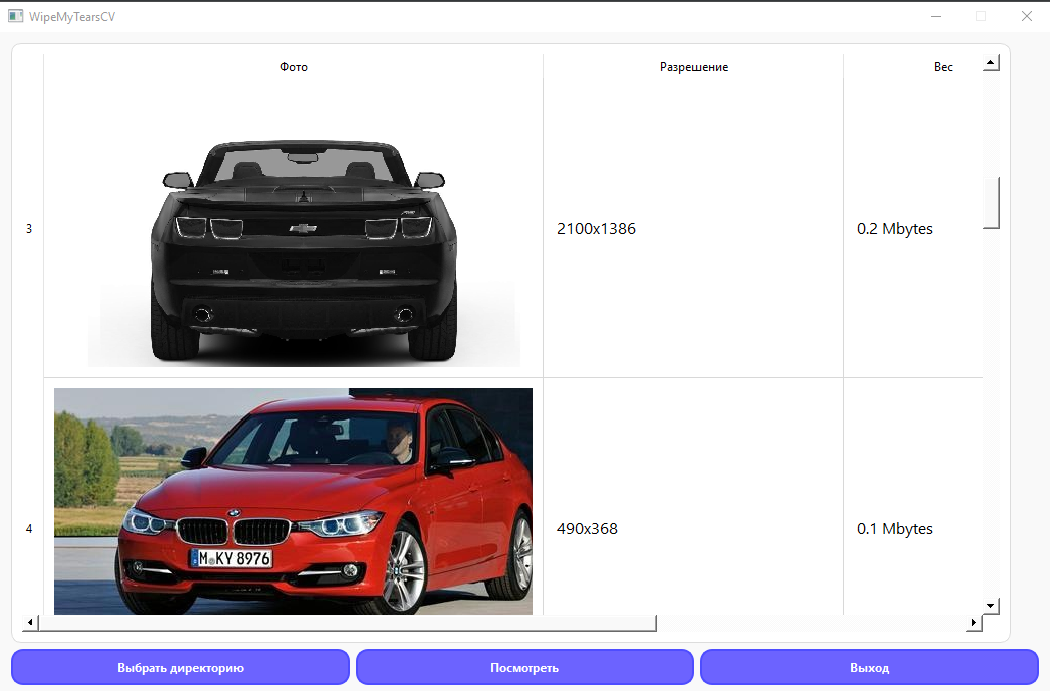


Рисунок 6. Главное окно приложения.

После подгрузки всех файлов в таблицу из выбранной директории, пользователь может посмотреть каждую такую машину отдельно, чтобы сделать это, пользователю необходимо выбрать строку с нужной машиной, нажать на данную строку, а затем нажать на кнопку «Посмотреть». В случае если пользователь не выбирает строку из таблицы, а просто нажимает кнопку «Посмотреть», то на экране появится модальное окно с предупреждением об ошибке пользователя.

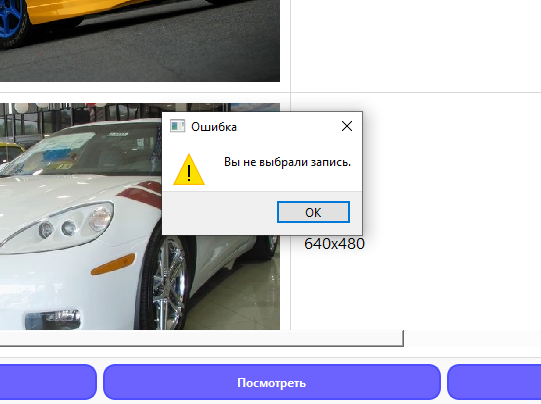


Рисунок 7. Модальное окно с ошибкой.

Если же пользователь все сделал правильно, а именно выбрал нужную запись и затем нажал на кнопку «Посмотреть», то на экране появится следующее окно ModalWindow, в котором выводится выбранное пользователем изображение автомобиля, а также доминирующие цвета на данном изображении.

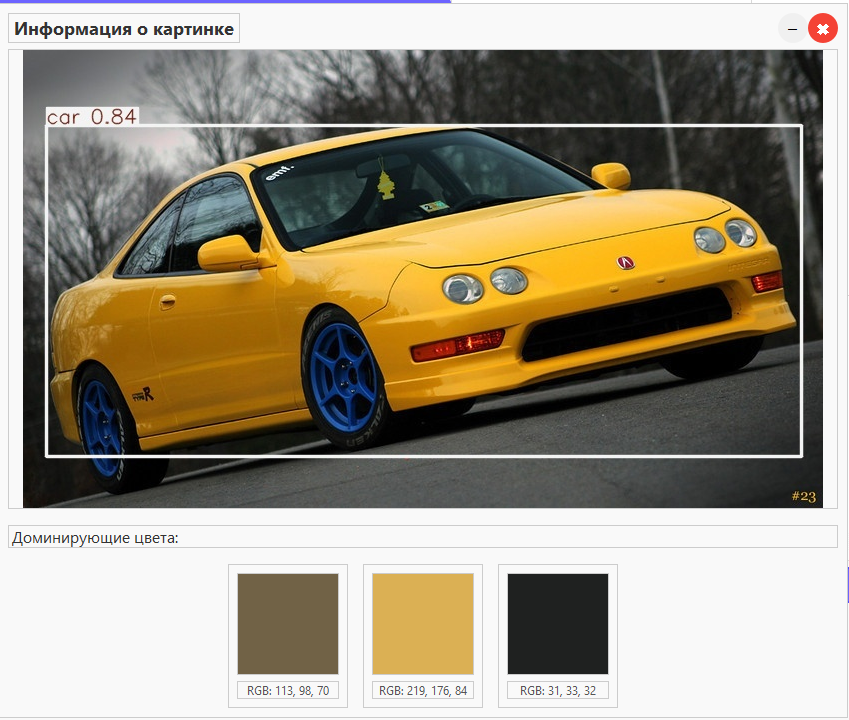


Рисунок 8. Модальное окно с выбранным изображением.

Часть изображения, которое пользователь выбрал, обводится в рамку, так отмечается автомобиль, найденный на изображении. После чего в данной рамке определяются главные цвета, данные цвета выводятся в виде квадратов с выводом данных в RGB формате в отдельной секции. После этого, пользователь может закрыть данное окно и снова выбрать запись с изображением автомобиля.

**ЗАКЛЮЧЕНИЕ**

Получилось реализовать программу с графическим интерфейсом для обнаружения машин на фотографиях.

Программа может определять машины и выделять их на фотографии. Получилось это осуществить с помощью модели искусственного зрения YOLOv5. Данные об обрабатываемых фотографиях выводятся в таблицу с результатом. В качестве результата записывается: наименование фотографии, разрешение и размер в мегабайтах.

Также имеется функция для определения цвета машины. Для того чтобы написать эту функцию был использован метод k-средних. Приложение выдает фотографию с обнаруженной машиной и открывает дополнительное окно с цветом.

В качестве дальнейшей работы над программой можно выделить оптимизацию, чтобы процесс обнаружения затрачивал не так много ресурсов, потому что, когда фотографий поступает слишком много, то потребляется много памяти. Также можно добавить функцию определения машины на видео, то есть её обнаружение в реальном времени.

**Список использованных источников.**

* 1. Модель для обнаружения объектов в реальном времени и сегментации изображений. https://github.com/ultralytics/yolov5
  2. Сайт с коллекциями датасетов для компьютерного зрения. https://universe.roboflow.com
  3. Книга “Python 3 и PyQt6” Николай Прохоренок.
  4. Документация по библиотеке PyQt6 https://doc.qt.io/qtforpython-6/
  5. Метод KMeans https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
  6. Использование элементов в библиотеке PyQt6 https://www.pythontutorial.net/pyqt/
  7. Настройка, использование yolov5 https://github.com/ultralytics/yolov5/issues/36
  8. Документация по использованию yolov5 https://docs.ultralytics.com/yolov5/
  9. Использование yolov5 на кастомном датасете https://www.youtube.com/watch?v=rZyY2pNzypQ
  10. Использование OpenCV в python https://www.tutorialkart.com/opencv/python/opencv-python-read-display-image/#gsc.tab=0