1. **ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

Рассмотрим подробно функционирование программы. Для этого проведем анализ основных блоков программы и рассмотрим их зависимости. А также проанализируем все функциональные компоненты, которые входят в состав кода программы, и рассмотрим назначение всех методов и переменных классов этих блоков.

В разрабатываемом приложении можно выделить следующие блоки:

* блок пользовательского интерфейса;
* блок разделения видеозаписей на кадры;
* блок обработки изображения;
* блок детектирования;
* блок стандартизации изображений;
* блок классификации;
* блок приведения полученных данных к одному формату;
* блок экспортирования результатов в базу данных.

Изначально пользователь попадает на главный экран, логика которого находится в классе Form1. Здесь расположен весь основной графический интерфейс. В данном классе находятся все элементы, через которые пользователь взаимодействует с приложением. Здесь отображаются все информационные сообщения и выводятся управляющие элементы. Пользователь управляет программой в основном через нажатие на клавиши, расположенные в окне приложения. При нажатии происходит вызов необходимых функций, происходят определенные операции и пользователь получает какое-либо уведомление о завершении произошедшей операции.

Главный экран приложения разделен на составные блоки, в соответствии с этапом обработки входных данных.

На главном экране пользователь может указать на папку, в которой находятся видеозаписи, на которых необходимо произвести распознавание. После этого производится выбор места расположения для полученных изображений, которые будут сохранены во вложенные папки в соответствии с названиями исходных видеозаписей. Также при обработке исходных видеоматериалов есть возможность настройки частоты создания изображений. После этого пользователь выбирает набор необходимых преобразований и фильтров для улучшения качества распознавания. Затем он может выполнить операцию детекции областей изображений, на которых есть дорожные знаки. После этого производится запуск классификатора обнаруженных знаков. И в конце происходит запись результатов в базу данных с заранее определенными полями.

* 1. **Классы разрабатываемого программного средства**
     1. **Класс ffmpegConverter**

Этот класс является функциональной оберткой консольного вызова программы ffmpeg из одноименного набора библиотек для разбиения с определенной частотой видеозаписи на изображения.

Поле sourcePath хранит в себе путь к набору видеозаписей для распознавания.

В поле fps хранится частота преобразования видеозаписи, то есть количество кадров за секунду видеозаписи.

Метод convertVidToImages() создает новый процесс, в котором через консоль Windows вызывает программу ffmpeg c определенным набором аргументов. Через аргументы передается расположение полный путь к видеозаписи и папки для сохранения полученных изображений, а также количество получаемых кадров за секунду.

Аргумент -qscale указывается для того, чтобы использовать фиксированную шкалу качества. Аргумент –r устанавливает частоту кадров. Пример команды вызова программы ffmpeg из консоли ОС Windows приведен на рисунке 3.1.



Рисунок 3.1 – Пример команды ffmpeg для раскадрирования видеозаписи

Метод convertVidToSubs() создает новый процесс, в котором через консоль Windows вызывает программу ffmpeg c определенным набором аргументов. Через аргументы передается расположение видеозаписи, некоторые дополнительные параметры и путь к новому файлу для сохранения полученных изображений.

Аргументы -vn и -an используются, чтобы пропустить включение потоков видео и аудио соответственно, независимо от того, отображены ли они вручную или автоматически, за исключением тех потоков, которые являются выходами сложных фильтров.

Аргумент -map нужна для ручного управления выбором потока в каждом выходном файле.

Пример команды вызова программы ffmpeg из консоли ОС Windows приведен на рисунке 3.2.

Метод convertAll() вызывает два предыдущих метода для полного преобразования набора видеозаписей в соответствующее число папок, содержащих изображения в формате .jpg и файлы субтитров в формате .txt. Он в зависимости от успешности выполнения включенных методов возвращает либо *true,* либо *false.*



Рисунок 3.2 – Пример команды ffmpeg для получения субтитров из видеозаписи

* + 1. **Класс ImgOps**

Данный класс содержит методы для преобразования изображений с целью увеличения количественно-качественных характеристик распознавания дорожных знаков. Он использует методы библиотеки EmguCV.

Используя метод *RGBtoGrey,* происходит преобразование изображения из цветного в градации серого. Это позволяет уменьшить количество цветовых каналов с 3 до 1, что позволяет использовать меньше вычислительных мощностей и уменьшает время распознавания.

Метод *RGBtoHSV* преобразует изображение из цветового пространства RGB в цветовое пространство HSV.

Для того чтобы получить бинарное изображение используется метод *toBinary*. Он принимает изображение, а также порог бинаризации. На выходе получается изображение, в котором каждый пиксель принимает значение “1” либо “0”. Если яркость пикселя исходного изображения меньше порога бинаризации, то в преобразованном изображении его значение будет равно нулю, если же больше – единице.

В методе *InterpolationResize* производится масштабирование изображения с использованием бинарной интерполяции. Данный метод на входе получает исходное изображение и необходимые размеры результата и возвращает полученное изображение.

Суть интерполяции заключается в использовании имеющихся данных для получения ожидаемых значений в неизвестных точках. Интерполяция изображений работает в двух измерениях и пытается достичь наилучшего приближения в цвете и яркости пикселя, основываясь на значениях окружающих пикселей. Бикубическая интерполяция рассматривает массив из 4x4 окружающих пикселей – всего 16. Поскольку они находятся на разных расстояниях от неизвестного пикселя, ближайшие пиксели получают при расчете больший вес. Бикубическая интерполяция производит значительно более резкие изображения, чем другие методы, и возможно, является оптимальной по соотношению времени обработки и качества на выходе. Результат увеличения с интерполяцией можно увидеть на рисунке 3.2(а).





а – исходное изображение; б – после увеличения с интерполяцией

Рисунок 3.2 Результат увеличения с интерполяцией



Рисунок 3.3 Результат увеличения без интерполяции

Для улучшения различимости элементов классифицируемого фрагмента изображения содержащего дорожный знак необходимо применить метод *ContrastAlignment*. Метод принимает исходное изображение и возвращает масштабированное изображение. В данном методе используется контрастное выравнивание CLAHE (Contrast-limited adaptive histogram equalization).

Контрастное выравнивание CLAHE используется для изображений, имеющих неоднородное геометрическое распределения яркостей. Оно анализирует небольшие участки изображения и позволяет усилить локальный контраст. Для каждого пикселя рассматривается небольшая окрестность изображения, по которой строится функция преобразования, при этом все изображение, как таковое, не используется. Оно позволяет уменьшить неоднородность освещения дорожных знаков. Результат контрастного выравнивания можно увидеть на рисунке 3.4.



а – исходное изображение; б – после контрастного выравнивания

Рисунок 3.4 Результат контрастного выравнивания CLAHE

В методе Filter происходит фильтрация изображения, после которой на готовом изображении будут находиться только те области изображения, цвет которых находится в заданном HSV интервале. Метод принимает нижний и верхний порог фильтрации и возвращает полученное изображение.

* + 1. **Класс SignsHaarCascade**

Этот класс выполняет распознавание знаков на основе выбранных каскадов Хаара. Класс содержит экземпляр класса CascadeClassifier из библиотеки EmguCV.

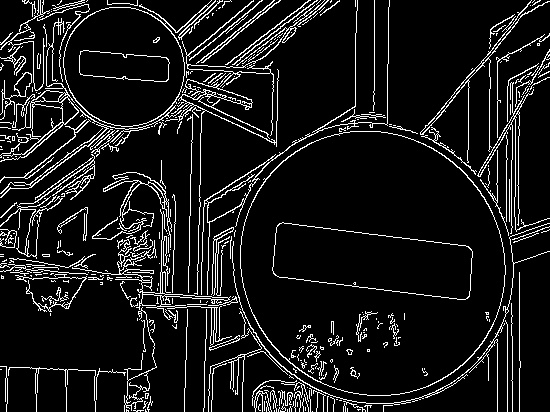
При создании объекта этого класса выбирается существующий на данном компьютере заранее обученный каскад, который будет использоваться для детекции знаков в базе фотографий.

Метод *detectAll* принимает изображение, на котором необходимо найти знаки. Здесь вызывается функция *DetectMultiScale* класса *CascadeClassifier* из библиотеки EmguCV, который находит прямоугольные области в изображении, которые, вероятно, содержат объекты, для которых обучен каскад, и возвращает эти области в виде последовательности прямоугольников. Функция сканирует изображение несколько раз и в разных масштабах. Каждый раз он учитывает перекрывающиеся области на изображении. Также может быть применена некоторая эвристика для уменьшения количества анализируемых областей, например, алгоритм Кэнни.

Детектор границ Кэнни - оператор [обнаружения границ](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%86) изображения. Границы здесь отмечаются там, где градиент изображения приобретает максимальное значение. Они могут иметь различное направление, поэтому алгоритм Кэнни использует четыре фильтра для обнаружения горизонтальных, вертикальных и диагональных ребер в предварительно размытом для удаления шумов изображении. Результат работы данного детектора можно увидеть на рисунке 3.5.

Далее вызывается метод *getListOfROI,* который принимает коллекцию *регионов интереса* – областей изображения, в которых были найдены нужные знаки. Метод возвращает коллекцию, содержащую части исходного изображения в ранее обнаруженных регионах интереса.





а – исходное изображение; б – после применения детектора

Рисунок 3.4 – Результат применения детектора границ Кэнни

* + 1. **Класс OpenHaarCascadeFileDialog**

Данный класс предназначен для создания окна, в котором пользователь выбирает файл формата XML, в котором хранится результат обучения каскада Хаара.

Главным методом этого класса является *openCascade*. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к нужному каскаду Хаара. Метод возвращает строку, в которой находится полный путь к выбранному каскаду.

* + 1. **Класс OpenVideoFolderFileDialog**

Этот класс нужен для вывода на экран окна для выбора папки с видеозаписями, которые нужно использовать для пополнения базы данных новыми данными.

Главным методом этого класса является *openFolder*. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к папке. Из нее программа получает доступ к видеозаписям, которые необходимо обработать и выдать нужный результат. Метод возвращает массив строк, в котором находится полный путь к видеозаписям из выбранной папки.

* + 1. **Класс OpenPictureFolderFileDialog**

Этот класс используется для вывода на экран окна, которое используется для выбора папки с изображениями, которые нужно использовать для нахождения знаков.

Главным методом этого класса является *openFolder*. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к папке, из которой программа может получить доступ к изображениям. Метод возвращает массив строк, в котором находится полный путь к изображениям из выбранной папки.

* + 1. **Класс ImageFolder**

Этот класс используется для создания объектов, которые содержат путь к папке с фотографиями. Также здесь хранятся и сами фотографии, и их количество в определенной папке.

Поле path хранит путь к набору изображений.

Поле count показывает количество изображений, которые находятся в пути path.

* + 1. **Класс VideoFolder**

Этот класс используется для создания объектов, которые содержат путь к папке с видеозаписями. Также здесь хранятся и сами видеозаписи, и их количество в определенной папке.

Поле path хранит путь к набору видеозаписей.

Поле count показывает количество видеозаписей, которые находятся в пути path.

* + 1. **Класс CNN**

В классе CNN будет находиться программная реализация свёрточной нейронной сети.

Поле trainPath хранит путь к директории, содержащей тренировочную выборку, разделенную на классы.

Поле dataPath хранит путь к папкам с изображениями для классификации.

Поле weights[] хранит веса ядра свёртки.

Поле bias отображает значение смещения. Смещение — это статическая величина, на которую следует “сместить” выходные значения. По своей сути это обычная операция сложения каждого элемента выходной матрицы с величиной смещения.

В поле stride хранится Шаг перемещения фильтра. Страйд определяет, на какое количество пикселов должен сместиться фильтр за один проход.

Поле trainLoss отображает качество обучения данной модели нейронной сети.

В методе PrepareData будет происходить подготовка исходных данных к формату, который пригоден для обучения нейронной сети.

В методе Learn будет производиться обучение нейронной сети на основе обучающей выборки, разделенной на классы.

В методе Classify заранее обученная сеть будет классифицировать найденный заранее знак на участке изображения и с процентной вероятностью делать выбор о принадлежности знака к тому или иному классу.

Метод CalculateLoss служит для отображения качества обнаружения обученной сети.

* 1. **Организация Properties.Settings**

Организация Properties.Settings — это xml файл, который можно найти в папке пользователя. Данный файл позволяет хранить и получать доступ к значениям, которые сохраняются между сеансами выполнения приложения. Эти значения называются *параметры*. Используя параметры, могут быть записаны пользовательские настройки или ценные сведения, которые приложению необходимо использовать.

Планируется хранить следующие настройки приложения:

* номер версии приложения;
* последний выбранный путь к папке с видеозаписями;
* последний выбранный путь к папке с изображениями;
* последний выбранный путь к обученному каскаду;
* флаг первого открытия приложения для отображения сообщения с инструкциями по пользованию.
  1. **Хранение полученных результатов в базе данных Oracle**

Для сохранения результатов в базе данных Oracle будет использоваться таблица с заранее определенными полями. Их можно увидеть в таблице 3.1.

Таблица 3.1 Структура таблицы в базе данных

|  |  |
| --- | --- |
| 1 Id | Уникальный номер знака |
| 2 Class | Вид знака |
| 3 Lattitude | Широта |
| 4 Longitude | Долгота |
| 5 Date | Дата обнаружения знака на видеозаписи |
| 6 RoadId | Id дороги, на которой расположен знак |
| 7 RoadKm | Километр от начала дороги, на которой расположен знак |