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 MainForm. Здесь расположен весь основной графический интерфейс. В данном классе находятся все элементы, через которые пользователь взаимодействует с приложением. Главный экран приложения разбит на части, на каждой из которых находятся управляющие элементы, с помощью которых производится тот или иной этап обработки исходных данных. 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.

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. Результаты предыдущих этапов обработки приводятся к единому формату для стандартизации и упрощения экспортирования данных. Результаты этого этапа обработки сохраняются по выбранному пользователем пути. И в конце происходит запись результатов в базу данных с заранее определенными полями. Так же все предыдущие этапы можно выполнить в полуавтоматическом режиме. Для этого необходимо заполнить все важные поля и использовать соответствующие элементы управления на главном экране, в котором находятся все необходимые пути, настройки и другие возможные опции. Диаграмма классов разрабатываемого средства изображена на чертеже ГУИР.400201.067 РР.1.

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

Этот класс является функциональной оберткой программы ffmpeg из одноименного набора библиотек для разбиения с определенной частотой видеозаписи на изображения. Здесь реализованы все необходимые методы для полной обработки каждой видеозаписи и получения полного набора важных для работы приложения данных.

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

Поле savePath формата *string* хранит путь, по которому необходимо сохранить полученный результат обработки.

Переменная timeStepMilliseconds отображает количество изображений, которое необходимо получить за каждую секунду исходной видеозаписи.

Коллекция pointsList объектов класса movementPoint служит для создания файла соответствия названия изображения полученного при обработке видеозаписи и его координат.

Метод ConvertAll является главным методом, с помощью которого происходит непосредственно само преобразование. В нем производится создание директории, куда будут сохранятся полученные изображения и их координаты, а также производятся вызовы методов для преобразования видеозаписи. Он принимает логическое значение rewrite, которое соответствует необходимости перезаписи уже созданных изображений по данному пути при совпадении их имен. Кроме того, здесь происходит перестановка элементов коллекции изображений в связи с неправильным порядком считывания их из папки, так как для работы приложения соответствие изображений и других данных является критичным в связи со спецификой работы программы.

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. Количество получаемых кадров за секунду видеозаписи высчитывается из отношения секунды времени к величине шага отсчета. Так же для сохранения полученных изображений создается новая папка с именем, соответствующим названию исходной видеозаписи.

Аргумент командной строки -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 использования программы ffmpeg для разбиения видеозаписи на кадры

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* нужен для ручного управления выбором потока в каждом выходном файле. Отсчет потоков начинается с нуля, и мы выбираем именно его, так как в данном случае использования программы у нас нет других потоков.

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



Рисунок 3.2 – Пример команды ffmpeg для получения субтитров из видеозаписи

Метод ParseSubtitleFile производит разбиение полученного текстового файла с субтитрами с помощью регулярного выражения. Каждое найденное соответствие приводится к объекту класса MovementPoint с помощью метода ParseMovementPoint. После обработки получается коллекция координат, которая используется в дальнейшей обработке.

В методе ParseMovementPoint происходит создание объекта класса MovementPoint путем разбиения текстовой строки по полям объекта класса.

Метод ConvertDegreesAndDecimalMinutesStringToDecimal необходим для приведения географических координат к пригодному для чтения десятичному виду.

* + 1. **Класс ImgOps**

Данный класс содержит методы для преобразования и фильтрации 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. Он принимает изображение в цветовом пространстве RGB и возвращает изображение в градациях серого цвета.

Метод RGBtoHSVпреобразует изображение из цветового пространства RGB в цветовое пространство HSV. Он принимает изображение в цветовом пространстве RGB и возвращает изображение в цветовом пространстве 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 бинарной 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 обычно 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(б).





а – исходное изображение; б – после увеличения с интерполяцией

Рисунок 3.2 Результат увеличения с интерполяцией



Рисунок 3.3 – Результат увеличения без интерполяции

В методе RGBFilter выполнятся выделение определенных цветов изображения в цветовом пространстве RGB. Данный метод принимает минимальные и максимальные значения для каждого цветового канала, которые необходимо выделить. На выходе получается черно-белое изображение, где пиксели белого цвета находятся в тех местах, где цветовые значения находились в введенном цветовом диапазоне.

В методе makeSmooth производится сглаживание изображения. Это происходит с помощью метода GaussianBlur из библиотеки EmguCV. В качестве аргументов он принимает размер исходного изображения по осям абсцисс и ординат, размер окна фильтра Гаусса.



Рисунок 3.4 – Результат использования функции makeSmooth

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 3.5.



Рисунок 3.5 – Результат применения 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** **MovementPoint**

В классе MovementPoint содержится сущность, описывающая точку координат, пригодном для создания объектов при экспортировании полученных данных.

Поле Lat отображает географическую широту.

Поле Lon отображает географическую долготу.

Поле Azimuth содержит азимут в интервале от 0 до 360.

Поле Date позволяет узнать, когда были получены данные о координатах той либо иной точки.

* + 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 в файле формата XML, 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.

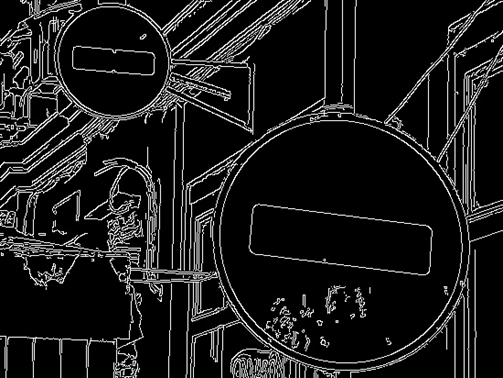


Рисунок 3.4 – Результат применения детектора границ Кэнни

* + 1. **Класс DetectFolder**

Класс DetectFolder выполняет поиск дорожных знаков для каждой фотографии из выбранной папки. Также здесь имеется возможность производить обработку изображения несколькими каскадами, что увеличивает количество видов найденных знаков.

В методе DetectAll() производится проверка всех фотографий в выбранной директории на предмет присутствия на них дорожных знаков с помощью файлов каскадов Хаара и производит запись на жесткий диск найденных областей интереса (обрезанных частей изображения, на котором были обнаружены дорожные знаки) в соответствии с названиями исходных фотографий.

* + 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 вызывается 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 ImageFolder**

Используя методы этого класса можно получить путь к директории, содержащей изображения в соответствующих форматах (jpg, png, bmp).

Метод Load() производит открытие папки по выбранному пути и заполнения массива путей к изображениям из этой папки и возвращает количество найденных изображений.

Метод GetCount() возвращает количество изображений в определенной папке.

Метод GetPath() возвращает путь к определенной папке.

Метод SetPath() производит замену пути к папке, после которой заново получает доступ к изображениям в папке по новому пути.

Метод GetAll() возвращает массив данных типа string в которых содержатся пути к изображениям из определенной папки.

Метод GetImg(*int*) возвращает путь к изображению с полученным номером.

Метод Sort() необходим для упорядочивания изображений по алфавиту для корректного соответствия имени файла с изображением и его координат.

* + 1. **Класс VideoFolder**

Используя методы этого класса можно получить путь к директории, содержащей видеозаписи в формате MP4.

Метод Load() производит открытие папки по выбранному пути и заполнения массива путей к видеозаписям из этой папки и возвращает количество найденных видеозаписей.

Метод GetCount() возвращает количество видеозаписей в определенной папке.

Метод GetPath() возвращает путь к определенной папке.

Метод SetPath() производит замену пути к папке, после которой заново получает доступ к видеозаписям в папке по новому пути.

Метод GetAll() возвращает массив данных типа *string* в которых содержатся пути к видеозаписям из определенной папки.

Метод GetImg(*int*) возвращает путь к видеозаписи с полученным номером.

* + 1. **Класс XmlFolder**

Используя методы этого класса можно получить путь к директории, содержащей файлы формата xml.

Метод Load() производит открытие папки по выбранному пути и заполнения массива путей к изображениям из этой папки и возвращает количество найденных изображений.

Метод GetCount() возвращает количество изображений в определенной папке.

Метод GetPath() возвращает путь к определенной папке.

Метод SetPath() производит замену пути к папке, после которой заново получает доступ к изображениям в папке по новому пути.

Метод GetAll() возвращает массив данных типа *string* в которых содержатся пути к изображениям из определенной папки.

Метод GetXml(*int*) возвращает путь к изображению с полученным номером.

Метод GetFolderPath возвращает путь к корневой директории с файлами формата xml.

* + 1. **Класс OpenVideoFolderFileDialog**

Этот класс используется 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 в форматах jpg, png, bmp из выбранной папки.

* + 1. **Класс OpenXmlFolderFileDialog**

Этот класс используется для вывода на экран окна, которое используется для выбора папки с файлами формата xml.

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 файлам. Метод возвращает массив строк, в котором находится полный путь к файлам.

* + 1. **Класс** **OpenPictureFileDialog**

Этот класс используется для вывода на экран окна, которое используется для выбора изображения.

Главным методом этого класса является openFile. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к изображению. Метод возвращает строку, в котором находится полный путь к изображению.

* + 1. **Класс** **OpenTextFileDialog**

Этот класс используется для вывода на экран окна, которое используется для выбора текстового файла формата txt.

Главным методом этого класса является openFile. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к текстовому файлу. Метод возвращает строку, в котором находится полный путь к текстовому файлу.

* + 1. **Класс** **OpenXmlFileDialog**

Этот класс используется для вывода на экран окна, которое используется для выбора файла формата xml.

Главным методом этого класса является openFile. При его вызове создается стандартное окно указания пути ОС Windows, с помощью которого пользователь указывает путь к файлу. Метод возвращает строку, в котором находится полный путь к файлу.

* + 1. **Класс PhCoord\_Connect**

Данный класс служит для создания коллекции вида PhotoData, с помощью которой можно соотнести изображение с его географическими координатами.

Главный метод Connect выполняет создание коллекции элементов класса PhotoData с последующим её возвратом. Поиск координат производится по времени с первого кадра. Далее к нулевому отсчёту добавляется частота раскадровки видеозаписи умноженная на номер изображения по порядку с последующим созданием объекта и добавлением его в коллекцию PhotoData.

* + 1. **Класс ShapeDetection**

В этом классе находятся функции поиска геометрических фигур и контуров на изображении.

Метод detectShapesпринимает изображение, производит поиск геометрических фигур (треугольники, квадраты, прямоугольники, круги, шестиугольники), выделяет их и возвращает полученное изображение. Поиск производится с помощью метода контурной аппроксимации.

В методе detectShape производится поиск и выделение областей определенных фигур. При вызове функции указывается, какие фигуры нужно обнаружить, очертить и вернуть полученное изображение. Работа методов класса основана на функциях библиотеки EmguCV.

* + 1. **Класс ShapeComparation**

В этом классе находятся функции поиска одинаковых областей на двух изображениях.

Метод FindMatch принимает изображения, производит поиск похожих областей изображений выделяет их и возвращает полученное изображение. Поиск производится с помощью метода k ближайших.

В методе Draw производится выделение похожих областей на результирующем изображении.

* + 1. **Класс Photo**

Этот класс содержит описание изображения с помощью его номера в папке и его имени.

Поле Number формата *int* содержит номер фотографии по порядку нахождения в папке.

Поле FileName формата *string* хранит путь к изображению на компьютере пользователя.

Каждое поле имеет открытые методы для их изменения либо получения значения.

* + 1. **Класс CNN**

Переменная net типа Network<double> хранит в себе слои нейронной сети (НС) и веса нейронов, находящихся на них.

В переменной stepCount находится счетчик итерации обучения сети.

Для проведения загрузки данных заранее обученной сети используется переменная loadedJson, в которой находится информация о НС в строковом формате, которая потом преобразуется к типу нейронной сети.

Поле JsonToSave хранит данные о сети в формате string для дальнейшего сохранения её данных.

Переменная Aim отражает минимальную точность, которую необходимо достичь при обучении НС.

Переменная Acc хранит достигнутую точность обученной НС.

В переменной classes типа *int* хранится количество слоев созданной НС.

Флаг isNetLearned отражает статус НС (обучена ли на данный момент НС).

Объект trainer класса SgdTrainer является необходимой для проведения обучения НС. В ней хранятся: скорость обучения НС, размер выборки для каждой стадии обучения, момент НС.

В переменной path хранится путь для сохранения либо загрузки данных сети.

Коллекция testAccWindow класса CircularBuffer<double> содержит в себе промежуточные данные о точности обучения сети на каждой стадии на основе тестирования тестовой выборки.

Коллекция trainAccWindow класса CircularBuffer<double> содержит в себе промежуточные данные о точности обучения сети на каждой стадии на основе тестирования тренировочной выборки.

Метод CreateCNN служит для создания нейронной сети, добавления необходимых слоев и инициализации классов распознаваемых дорожных знаков. Возвращает количество слоев созданной НС.

В методе TeachCNN происходит обучение нейронной сети методом градиентного спуска. Здесь создается обучающая и проверочная выборки и происходит корректировка весов каждого слоя до тех пор, пока точность распознавания сети не достигнет необходимого значения.

Метод Train вызывается на каждую итерацию обучения. Он производит изменение весов каждого слоя для улучшения качества распознавания.

Метод Test служит для проверки качества распознавания НС для тех весовых коэффициентов, которые на данный момент записаны в НС.

В методе SaveCNN происходит запись в файл по заранее определенному пути данных об обученной сети. Это выполняется с помощью приведения данных к стандартному виду, пригодному для дальнейшего использования. В имени полученного файла находятся наиболее важные данные о сети для выбора необходимой сети при загрузке.

При вызове метода LoadCNN производится загрузка и инициализация данных о сети, параметры которой находятся в текстовом файле по заданному пути.

Метод Recognize служит для непосредственной классификации дорожного знака на изображении. Для этого информацию об изображении в виде байтового массива пропускают через обученную сеть и на основе полученных результатов на выходном слое, и выбирается наибольшее значение слоя. Номер нейрона выходного слоя говорит о принадлежности изображения к данному классу дорожных знаков. Далее вызывается метод GetClassNameFromNumber, который на основе номера класса возвращает его названия для удобного восприятия пользователем.

Метод GetLayersCount возвращает количество слоев в созданной нейронной сети.

Метод GetClassesCount возвращает количество распознаваемых классов дорожных знаков в созданной нейронной сети.

Метод GetAccuracy возвращает точность заранее обученной нейронной сети.

Метод IsLearned возвращает флаг, который говорит о состоянии нейронной сети (настроены ли весовые коэффициенты должным образом для распознавания с заданной точностью).

* + 1. **Класс CircularBuffer**

Данный класс служит для хранения промежуточных результатов обучения нейронной сети. Он является коллекцией шаблонов с возможностью добавления в нее новых элементов.

Переменная buffer является массивом шаблонов, в котором хранятся элементы.

Переменная nextFree хранит номер следующей свободной ячейки для добавления нового элемента.

Метод Add производит добавление нового элемента и изменение номера следующего возможного для добавления элемента.

* + 1. **Класс DataSet**

Этот класс производит создание набора данных для обучения нейронной сети. Он содержит в себе набор изображений в количестве заранее определенного размера пачки из обучающей выборки и номеров классов, к которым они принадлежат.

Метод NextBatch производит создание кортежа из набора обучающих изображений, набора тестовых изображений и массива соответствующих тестовой выборке номеров классов.

* + 1. **Класс Datasets**

Данный класс содержит в себе объекты Train и Test класса DataSet, которые являются хранилищами обучающих и тестовых выборок для обучения сети.

В методе Load происходит чтение выборок с жёсткого диска и создание объектов Train и Test. При удачном чтении возвращает значение *true*, а при возникновении ошибки – *false*.

* + 1. **Класс ImageEntry**

В классе ImageEntry содержится сущность описывающая изображение в виде, пригодном для создания тестовой выборки при обучении сети.

Переменная Image хранит непосредственно само изображение в виде массива байт.

Переменная Label содержит номер класса, к которому принадлежит изображение.

* + 1. **Класс ImageReader**

Данный класс используется для считывания с ПК пользователя изображений и приведения их к общему формату для создания обучающей и тренировочной выборок.

Метод Load производит инициализацию коллекций для последующего их заполнения. Он принимает путь в строковом формате к папке с изображениями.

Метод LoadImages принимает путь к директории с изображениями и выполняет операцию считывания для каждого найденного изображения в выбранной директории и его загрузки в коллекцию. Метод возвращает коллекцию изображений в виде массива байт.

Метод LoadLabels принимает путь к директории с изображениями, в которой находится и файл с субтитрами из видеозаписи и выполняет операцию считывания названий для каждого найденного изображения в выбранной директории, в которых находится номер класса дополнение коллекции номерами классов.

* 1. **Структура организации Properties.Settings**

Организация Properties.Settings – это файл формата xml, который можно найти в папке c приложением. Данный файл позволяет хранить и получать доступ к значениям, которые сохраняются между сеансами выполнения приложения. Эти значения называются параметрами. Используя параметры, могут быть записаны пользовательские настройки или ценные сведения, которые приложению необходимо использовать при следующем запуске приложения. Перечень всех сохраняемых параметров можно увидеть в таблице 3.1.

Таблица 3.1 Сохраняемые параметры приложения

|  |  |
| --- | --- |
| Название параметра | Описание параметра |
| 1 | 2 |
| 1 is\_opened\_first\_time | Флаг первого открытия пользователем приложения. Используется для отображения инструкции |
| 2 last\_path\_for\_detected\_images\_to\_save | Последний выбранный путь для сохранения частей фотографий, на которых были найдены дорожные знаки |
| 3 last\_path\_for\_images\_to\_detect | Последний выбранный путь для поиска изображений, на которых необходимо провести поиск знаков |
| 4 last\_path\_for\_images\_to\_save | Последний выбранный путь для сохранения изображений, полученных при разбиении видеозаписей на кадры |
| 5 last\_path\_to\_results\_save | Последний выбранный путь для сохранения полученных результатов работы приложения |

Продолжение таблицы 3.1

|  |  |
| --- | --- |
| 1 | 2 |
| 5 last\_path\_to\_cascade | Последний выбранный путь к файлу формата xml, в котором хранится каскад Хаара |
| 6 last\_path\_to\_learn\_pictures | Последний выбранный путь к директории, содержащей обучающие изображения для тренировки нейронной сети |
| 7 last\_path\_to\_network | Последний выбранный путь к текстовому файлу, в котором хранятся параметры нейронной сети |
| 8 last\_path\_to\_pictures | Последний выбранный путь для изображений, на которых необходимо провести классификацию знаков |
| 9 last\_path\_to\_test\_pictures | Последний выбранный путь к директории, содержащей тестовые изображения для тренировки нейронной сети |
| 10 last\_path\_to\_videos | Последний выбранный путь для директории с видеозаписями, которые необходимо обработать |
| 11 version | Номер текущей версии приложения |

* 1. **Экспорт результатов в базу данных**
     1. **Класс ResultExport**

В этом классе происходит запись результатов в локальное хранилище, связь с базой данных, записи новых результатов и обновления уже существующих.

Поле login содержит имя пользователя для подключения к базе данных.

Поле password содержит пароль для подключения к базе данных.

Поле dataSourceуказывает значение, соответствующее атрибуту источника данных.

Объект xmlFolder класса XmlFolder хранит пути к файлам формата xml.

Объект formatter класса XmlSerializer необходим для сохранения и чтения xml файлов.

Объект con класса OracleConnection служит для создания соединения с удаленной базой данных Oracle.

Массив results, в котором хранятся объекты класса Result, является промежуточным хранением для экспортирования полученных результатов работы.

Поле constr хранит параметры подключения к удаленно базе данных Oracle.

В методе Connect происходит подключение к базе данных с помощью связки логина и пароля и проверяется доступность базы данных. Метод возвращает значение true, если соединение установлено и база данных доступна, либо значение false, если произошла какая-либо ошибка.

В методе ExportToDB.выполняется считывание локального хранилища и последующее экспортирование каждого файла путем приведения файла формата xml к массиву results и последующей записи полей этого массива в удаленную базу данных.

Метод ExportToFile производит запись результатов обработки видеозаписей в локальное хранилище в виде файлов формата xml с соответствующими полями.

Метод ExportFolder служит для проведения записи всех файлов выбранной директории.

Метод UploadFromFile производит считывание файла формата xml на локальном хранилище, приведение его к виду массива объектов класса Result.

Метод CloseConnection выполняет закрытие соединения с удаленной базой данных.

* + 1. **Класс Result**

Этот класс служит для стандартизации результатов обработки и распознавания.

Поле id отображает уникальный номер знака.

Поле signClass содержит номер класса, к которому принадлежит знак.

Поле lattitude показывает географическую широту найденного дорожного знака.

Поле longitude показывает географическую долготу найденного дорожного знака.

Поле date содержит время на видеозаписи, когда найденный дорожный знак был заснят.

Поле roadId указывает номер дороги, на которой расположен найденный дорожный знак.

ПолеroadKm показывает километр от начала дороги, на которой расположен знак.

Метод getId возвращает уникальный номер знака в формате *int*.

Метод getSignClass возвращает номер класса знака в формате *int*.

Метод getLattitude возвращает географическую широту расположения знака в формате *string*.

Метод getLongitude возвращает географическую долготу расположения знака в формате *string*.

Метод getDate возвращает время на видеозаписи, когда дорожный знак был заснят, в формате DateTime.

Метод getRoadId возвращает уникальный номер дороги, на которой расположен знак в формате *int*.

Метод getRoadKm возвращает километр от начала дороги, на которой расположен знак, в формате double.

* 1. **Хранение полученных результатов в базе данных Oracle**

Для сохранения результатов в базе данных Oracle будет использоваться таблица определенными полями. Их можно увидеть в таблице 3.2.

Таблица 3.2 Структура таблицы в базе данных

|  |  |
| --- | --- |
| Название поля | Описание поля |
| 1 | 2 |
| 1 Id | Уникальный номер знака |
| 2 Class | Номер класса знака |
| 3 Lattitude | Географическая широта расположения знака |
| 4 Longitude | Географическая долгота расположения знака |
| 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 |