Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

К ЗАЩИТЕ ДОПУСТИТЬ
Зав. каф. ЭВМ
_____ Б.В. Никульшин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему
ПРОГРАММНОЕ СРЕДСТВО РАСПОЗНАВАНИЯ, АНАЛИЗА И УЧЕТА КОМПОНЕНТОВ ДОРОЖНОЙ ИНФРАСТРУКТУРЫ

БГУИР ДП 1–40 02 01 01 067 ПЗ

Студент	Д.О. Санкевич
Руководитель	И.И. Глецевич
Консультанты:	
от кафедры ЭВМ	И.И. Глецевич
по экономической части	Ф.М. Файзрахманов
Нормоконтролер	А.С. Сидорович
Рецензент	

МИНСК 2019
Реферат

Дипломный проект предоставлен следующим образом. Электронные носители: 1 компакт-диск. Чертежный материал: 6 листов формата А1. Пояснительная записка: 102 страницы, 16 рисунков, 4 таблицы, 12 литературных источников, 2 приложения.

Ключевые слова: обработка видеозаписей, обработка изображений, географические координаты, распознавание образов, каскады Хаара, .Net приложение, классификация, сверточные нейронные сети, базы данных Oracle.

Предметной областью данного проекта являются компоненты дорожной инфраструктуры. Объектом разработки является учет дорожных знаков, находящихся на дорогах Республики Беларусь.

Целью разработки является программное средство, способное с высокой точностью распознавать на полученных видеозаписях дорожные знаки и их координаты и производить запись результатов в базу данных.

В программном средстве используются каскады Хаара и сверточные нейронные сети, а также набор утилит FFMPEG.

Разработано приложение, с точностью распознавания около 94%.

Данная разработка будет применяться в государственной компании для добавления на ранее разработанную фотокарту компонентов дорожной инфраструктуры.

Принимая во внимание результаты оценки экономической эффективности проекта, было выявлено, что разработка программного средства является экономически целесообразной и выгодной.

Приложение готово к использованию, однако имеются возможности для улучшения харакстеристик.

Реферат

Дипломный проект предоставлен следующим образом. Электронные носители: 1 компакт-диск. Чертежный материал: 6 листов формата А1. Пояснительная записка: 102 страницы, 16 рисунков, 4 таблицы, 12 литературных источников, 2 приложения.

Ключевые слова: обработка видеозаписей, обработка изображений, географические координаты, распознавание образов, каскады Хаара, .Net приложение, классификация, сверточные нейронные сети, базы данных Oracle.

Предметной областью данного проекта являются компоненты дорожной инфраструктуры. Объектом разработки является учет дорожных знаков, находящихся на дорогах Республики Беларусь.

Целью разработки является программное средство, способное с высокой точностью распознавать на полученных видеозаписях дорожные знаки и их координаты и производить запись результатов в базу данных.

В программном средстве используются каскады Хаара и сверточные нейронные сети, а также набор утилит FFMPEG.

Разработано приложение, с точностью распознавания около 94%.

Данная разработка будет применяться в государственной компании для добавления на ранее разработанную фотокарту компонентов дорожной инфраструктуры.

Принимая во внимание результаты оценки экономической эффективности проекта, было выявлено, что разработка программного средства является экономически целесообразной и выгодной.

Приложение готово к использованию, однако имеются возможности для улучшения харакстеристик.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Общая протяженность сети автомобильных дорог общего пользования в Республике Беларусь, согласно [1], составляет около 94,797 тыс. км, из них:

– 15,476 тыс. км – республиканские дороги;

– 70,192 тыс. км – местные.

Для регулирования дорожного движения по всей длине дорог установлено огромное количество дорожных знаков. Они информируют водителей об опасных участках дороги, указывают направление движения, запрещают или дают право проезда, обязывают снизить скорость, а также выполняют множество других задач.

Для удобства построения маршрута в распоряжении водителей есть большой выбор средств навигации. Для максимально правдоподобного отображения картины предполагаемого пути необходимо в полной мере отразить все важные для пользователя компоненты дорожной инфраструктуры.

Целью данного дипломного проекта является разработка и реализация программного средства, способного на основе обновляемой базы видеозаписей полученных с помощью видеорегистраторов детектировать, классифицировать и заносить в базу данных необходимые дорожные знаки.

В соответствии с поставленной целью были определены следующие задачи:

– выбор доступных технологий, с помощью которых лучше всего происходит выполнение поставленной цели;

– разделение программы на составные блоки, каждый из которых выполняет часть поставленной цели;

– анализ доступных алгоритмов обнаружения и классификации дорожных знаков и выбор наиболее подходящих;

– программная реализация алгоритмов обнаружения и классификации обнаруженных дорожных знаков на изображении;

– выбор преобразований для увеличения количества найденных дорожных знаков;

– разработка модуля экспортирования в базу данных полученных результатов.

Разрабатываемое программное средство будет выполнять следующие функции:

– преобразование видеозаписей в коллекцию изображений и координат;

– подготовку изображений для улучшения качества распознавания;

– детектирование дорожных знаков на полученных изображениях;

– классификация найденных знаков;

– занесение названий найденных знаков, их координат и времени обнаружения на видеозаписи в базу данных.

1 ОБЗОР ЛИТЕРАТУРЫ

1.8 Обзор существующих аналогов

На этапе проектирования системы было проведено исследование существующих аналогов. Среди обнаруженных аналогов преобладают программные средства, которые распознают дорожные знаки с помощью видеорегистратора или смартфона. Они выводят на экран обнаруженные дорожные знаки и предупреждают пользователя о превышении скорости.

Наиболее приближенным среди найденных приложений является программное средство Roadly для ОС Android.

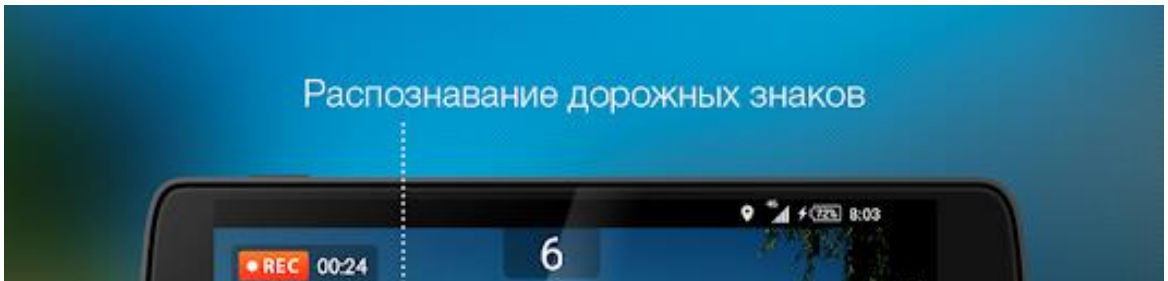


Рисунок 1.1 – Приложение Roadly [2]

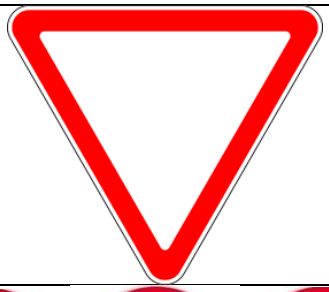
Данное приложение является видеорегистратором с возможностью обнаружения на видео дорожных знаков, а также отображения предупреждений пользователю об их присутствии на пути. Обнаруженные на устройстве знаки сохраняются в общую базу, откуда потом они загружаются к другим пользователей. Чем больше пользователей приложения, тем чаще и полнее обновляется база. Кроме того, оно автоматически подстраивает экспозицию, чтобы не допускать переосвещенных областей, а также умеет самостоятельно запускаться при обнаружении движения или при повороте устройства в горизонтальное положение. Приложение может работать в фоне как авторегистратор и выдавать предупреждения поверх любых навигационных приложений, например: Waze, Навител Навигатор (Navitel), Яндекс.Навигатор и других [2].

На данный момент приложение загрузило более 500 000 человек (Google play).

В текущей версии приложение способно распознавать следующие дорожные знаки (см. таблицу 1.1):

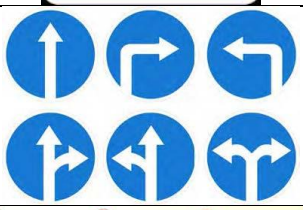
– уступи дорогу;
– ограничения скорости;
– обгон запрещен;
– остановка запрещена;
– стоянка запрещена;
– пешеходный переход;
– знаки, предписывающие проезд перекрестков;
– предупреждающие знаки.

Таблица 1.1 – Изображения знаков [3]

Название знака	Изображение знака
1	2
Уступи дорогу	
Ограничения скорости	

Обгон запрещен	
Остановка запрещена	

Продолжение таблицы 1.1

1	2
Стоянка запрещена	
Пешеходный переход	
Знаки, предписывающие проезд перекрестков	
Предупреждающие знаки	

К минусам данного аналога можно отнести высокое энергопотребление, в связи с высоким потреблением ресурсов мобильного устройства, нестабильную работу и низкое качество видеозаписи.

Также в ходе исследования было обнаружено, что многие известные автопроизводители разработали подобные приложения либо ведут работы в данном направлении. Как пример можно привести

Opel Eye, Speed Limit Assist (Mercedes-Benz), Road Sigh Information (Volvo). Применяемые на автомобилях системы распознавания дорожных знаков имеют типовую конструкцию, которая включает в себя видеокамеру, блок управления и средство вывода информации (экран). Камера снимает пространство перед автомобилем в зоне расположения дорожных знаков (сверху и справа по ходу движения) и передает изображение в электронный блок управления. Тот же в свою очередь по характерным признакам распознает знаки и выводит их на дисплей комбинации приборов или дисплей информационной системы и остается видимым, пока ограничение не закончится или не будет изменено.

Системы с похожим принципом работы встроены в самоуправляемые автомобили. Однако эти системы являются более сложными, так как во время поездки они должны сканировать окружающее машину пространство, распознавая сотни элементов: другие автомобили (едущие и припаркованные на периферии), пешеходов, дорожную разметку и знаки, плотность дорожного потока и свободное пространство на парковке.

1.9 Аналитический обзор

Разрабатываемое программное средство будет являться .NET Desktop приложением для ОС Windows, не требующим подключения к сети Интернет при наличии на ПК пользователя всех необходимых библиотек (ffmpeg, EmguCV).

Реализация в виде приложения Windows Forms позволит предоставить пользователю удобный интерфейс. Это графические приложения, которые легко развертывать и обновлять, которые могут работать, при подключении к сети Интернет или без него, и могут получать доступ к ресурсам на локальном компьютере более безопасным способом, чем традиционные приложения на основе Windows. В Windows Forms форма – это визуальный компонент, на которой пользователю выводится вся необходимая информация. Обычно приложение Windows Forms строится путем помещения элементов управления на форму и написания кода для ответной реакции на действия пользователя, такие как щелчки мыши или нажатия клавиш. Элемент управления – это отдельный элемент пользовательского интерфейса, предназначенный для хранения, отображения или ввода данных [4].

ffmpeg – бесплатный набор кроссплатформенных библиотек (лицензии GPL 2.0 и LGPL 2.1) который можно использовать в своих проектах, как в коммерческих, так и в свободно-распространяемых. С помощью программы ffmpeg, которую можно запускать из командной строки Windows с необходимыми аргументами, можно конвертировать видео из одного формата в другой, склеивать несколько видеороликов в один, разбирать видео на отдельные кадры (изображения) с определенной частотой, а потом склеивать обратно, ускорять-замедлять, масштабировать, добавлять субтитры и несколько звуковых дорожек [5].

Список доступных возможностей данного набора библиотек приведен на рисунке 1.2.

EmguCV – это кроссплатформенная оболочка для библиотеки обработки изображений OpenCV, которая позволяет использовать ее функции в фреймворке .Net и создавая приложения на языке c#.

OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом) – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Она реализована на C/C++, также разрабатывается для Python, Java, Ruby и других языков. Может свободно использоваться в академических и коммерческих целях – распространяется в условиях лицензии BSD.

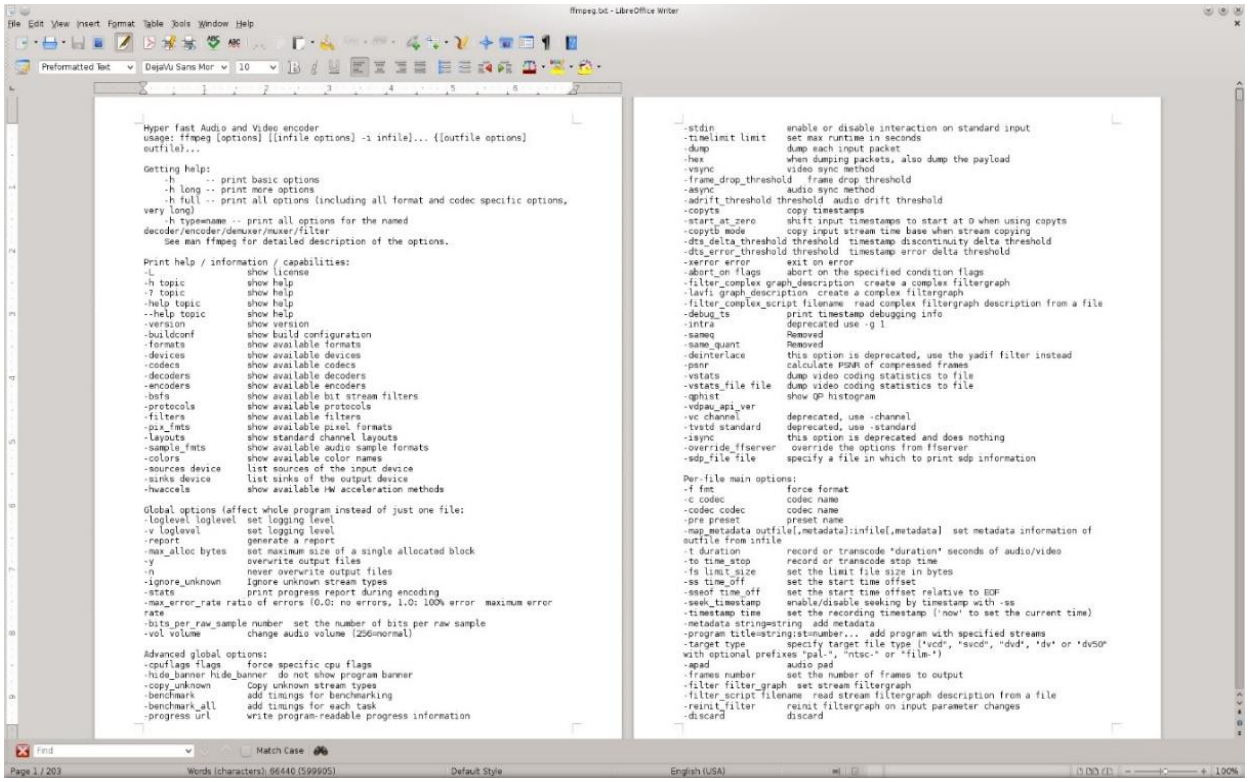


Рисунок. 1.2 - Список возможностей команды ffmpeg [3]

Фактически, OpenCV – это набор типов данных, функций и классов для обработки изображений алгоритмами компьютерного зрения. В библиотеке присутствует более 2500 оптимизированных алгоритмов, которые включают в себя полный набор как классических, так и современных алгоритмов компьютерного зрения и машинного обучения. Данная библиотека ориентируется в основном на приложения в режиме реального времени и использует преимущества команд MMX и SSE. Кроме того, в настоящее время активно развиваются полнофункциональные интерфейсы CUDA и OpenCL.

Основной идеей создания приложений на платформе .NET Framework является удобство и гибкость разработки за счет предоставления разработчику возможности создавать приложения различных типов, способных запускаться на различных типах устройств и в различных средах. Вторым принципом стала ориентация на семейство систем Microsoft Windows (XP, 7, 8, 10) [6].

1.3 Событийно-ориентированное программирование

Важным вопросом при разработке приложения является его интерфейс. В данном проекте при разработке пользовательского интерфейса был сделан выбор в сторону событийно-ориентированного программирования (СОП) как наиболее подходящего для поставленных задач. СОП можно также определить, как способ построения компьютерной программы, при котором в коде явным образом выделяется главный цикл приложения, тело которого состоит из двух частей: выборки события и обработки события. Событие в данном контексте можно определить, как изменение состояния определенного компонента.

В языке C# события реализованы как элемент языка и являются членами классов. Механизм событий здесь реализует шаблон проектирования Publisher-Subscriber. Данный шаблон является вариацией шаблона Наблюдатель. Каждый объект может публиковать сообщения без адресата, а получатели сами подписываются на определенные виды сообщений [5].

1.4 Базы данных Oracle

На данный момент Oracle является одной из наиболее развитых, мощных и производительных баз данных. База Oracle – реляционная база данных, со сложной структурой, но в то же время и с огромными возможностями. По своей функциональности, она не имеет аналогов. Одной из характеристик СУБД Oracle является функционирование системы на большинстве платформ. В том числе на больших ЭВМ, UNIX-серверах, персональных компьютерах и так далее. Одна из отличительных особенностей сервера Oracle – возможность хранения и обработки различных типов данных. Данная функциональность интегрирована в ядро СУБД и поддерживается модулем interMedia в составе Oracle Database. Он обеспечивает работу с текстовыми документами, включая различные виды поиска, в том числе контекстного; работу с графическими образами более двадцати форматов, а также работу с аудио и видеоинформацией. На сегодняшний день она стала обязательной частью практически любой серьезной информационной системы.

Механизмы масштабирования последней версии позволяют безгранично увеличивать мощность и скорость работы сервера Oracle и своих приложений, добавляя новые и новые узлы кластера. Это не требует остановки работающих приложений, не требует переписывания старых приложений, разработанных для обычной архитектуры, состоящей из одного ПК. Кроме того, выход из строя отдельных узлов кластера также не приводит к остановке приложения.

Последние версии стали значительно проще в установке и первоначальной настройке. Также возросли возможности по специализированной настройке работы СУБД под конкретную задачу. В результате, и при работе с OLTP-системой, и с хранилищем данных, используя эти возможности по настройке СУБД Oracle, можно достичь поистине впечатляющих результатов.

Также стоит рассказать о грамотной политике Oracle по отношению к обновлению СУБД. Понимая то, что переход с более старой версии СУБД на новую довольно трудоемкая процедура, которая обязательно будет включать в себя тестирование работы существующих приложений в новом окружении, Oracle, при выпуске новых продуктов уделяет большое внимание совместимости в новых версиях с более старыми, делая этот переход максимально удобным для разработчика.

Помимо этого, для перехода и переноса данных из СУБД других фирм (IBM DB2, MySQL, Microsoft SQL Server, Microsoft Access, Sybase, Teradata) в СУБД Oracle, Oracle бесплатно предлагает специальный инструментарий. Обладая удобным графическим интерфейсом, Oracle Migration Workbench в пошаговом режиме, полуавтоматически, поможет выполнить довольно непростую процедуру миграции [7].

1.5 Entity Framework

Entity Framework предоставляет разработчику специальную объектно-ориентированную технологию, основанную на базе фреймворка .NET, для работы с данными. Если традиционные средства .NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет функционал для более высокого уровня абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже можем работать с объектами.

Центральной концепцией здесь является понятие сущности или *entity*. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из баз данных, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является *Entity Data Model* – набор концепций, которые описывают структуру данных независимо от формы, в

которой они хранятся в базе данных. Эта модель сопоставляет классы сущностей с реальными таблицами в базе данных. Также она поддерживает набор примитивных типов данных, которые определяют свойства в концептуальной модели.

Ее архитектуру можно разделить на три уровня:

1. Концептуального уровня, где происходит определение классов сущностей, используемых в приложении;
2. Уровня хранилища, который определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных;
3. Уровня сопоставления (маппинга), который служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц.

Таким образом, мы можем через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

1.6 Постановка задачи

Разрабатываемое программное средство должно отвечать следующим требованиям:

1. Иметь удобный и интуитивно-понятный интерфейс, который в полной мере дает пользователю контроль над работой приложения;
2. В полной мере выполнять необходимый функционал, определенный в поставленных задачах;
3. Не допускать ошибок, приводящих к остановке работы приложения.

1.7 Анализ источников

Исчерпывающие инструкции по созданию приложения Windows Forms и существующие возможности подобных приложений описаны в [8]. Так же там показаны примеры использования доступных элементов управления и их возможности.

Существующие алгоритмы детектирования и распознавания дорожных знаков, такие как каскады Хаара и нейронные сети, их достигнутая точность и сравнительная характеристика были рассмотрены в [9]. Здесь же сказано, что признаки Хаара состоят из смежных прямоугольных областей. Они позиционируются на изображении, далее суммируются интенсивности пикселей в областях, после чего вычисляется разность между суммами. Эта разность и будет значением определенного признака, определенного размера, определенным образом расположенного на изображении.

Для использования ffmpeg изучена документация и инструкции по использованию, которые подробно расписаны в [10]. Были отобраны необходимые аргументы и их диапазон для лучшего и наиболее полного кадрирования для того, чтобы сохранить баланс между количеством кадров и занимаемым пространством, а также инструменты для экспорта в текстовый файл субтитров.

Для работы с библиотекой EmguCV был изучен [11]. Здесь была обнаружена вся необходимая информация для разработки программ с помощью библиотеки EmguCV. В данном источнике содержится описание всех классов, методов и полей присутствующих в данной библиотеке. Были получены необходимые знания для работы с этой библиотекой в рамках обработки изображений и компьютерного зрения.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Изучив теоретические аспекты разрабатываемого средства и выработав список требований необходимых для разработки, разбиваем программу на функциональные блоки (модули).

В разрабатываемом приложении можно выделить следующие блоки:

– блок пользовательского интерфейса;

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

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.067 С1.

Рассмотрим функциональные блоки приложения.

2.1 Блок пользовательского интерфейса

Блок пользовательского интерфейса представляет собой совокупность средств, при помощи которых пользователь взаимодействует с приложением.

Для построения интерфейса используется технология Windows Forms - интерфейс программирования приложений, отвечающий за графический интерфейс и представление данных, получение и обработку ввода пользователя и представляет собой событийно-ориентированное приложение, поддерживаемое Microsoft .NET Framework. Здесь пользователь взаимодействует с программой с помощью различных элементов управления. Так как обработка исходной информации происходит в несколько этапов, то интерфейс изменяется по мере прохождения определенных стадий.

2.2 Блок разделения на кадры

Блок разделения на кадры выполняет задачу кадрирования полученных видеозаписей для последующей работы с отдельными изображениями. Он является оберткой ffmpeg – набора программ с открытым исходным кодом, которые позволяют обрабатывать, конвертировать и передавать цифровые аудио- и видеозаписи в различных форматах. В данной программе из набора используется только одноименный модуль ffmpeg – утилита командной строки для конвертирования видеофайла из одного формата в другой. Он вызывается как отдельный процесс, в который передается набор аргументов (путь к папке с видеозаписями для обработки, путь к папке для сохранения результатов, частота преобразования). В результате получается набор папок с названиями соответствующими исходным видео в заданной директории.

2.3 Блок обработки изображения

Блок обработки изображения необходим для улучшения количественных и качественных характеристик модулей детектирования и классификации. Здесь происходит преобразования изображений в наиболее удобные для работы форматы и цветовые форматы. Функционал основан на использовании библиотеки EmguCV.

2.4 Блок детектирования

Блок детектирования работает на основе алгоритма каскадов Хаара, используя Виолы-Джонса для нахождения знаков на изображении. В методе Виолы-Джонса основу составляют примитивы Хаара, представляющие собой результаты разбития заданной прямоугольной области на наборы разнотипных прямоугольных подобластей.

Метод использует следующие ключевые концепции:
– простые прямоугольные функции, называемые функциями Хаара;
– метод машинного обучения AdaBoost;
– интегральное изображение для упрощения поиска;
– каскадный классификатор для эффективного совмещения множественных функций.

Каскады Хаара – наборы масок, прямоугольных окошек, каждое из которых представляет собой изображение с неким черно-белым узором

(комбинацией черных и белых частей). Таких масок может быть неограниченное множество, сложность узоров может также разниться (см. рис. 2.1).

AdaBoost – это алгоритм усиления классификаторов. Усилением классификаторов называют ансамблевый алгоритм обучения, который использует множество алгоритмов обучения, например, деревья решений, и объединяет их. Целью является взять набор или группу слабых классификаторов и объединить их в один сильный.

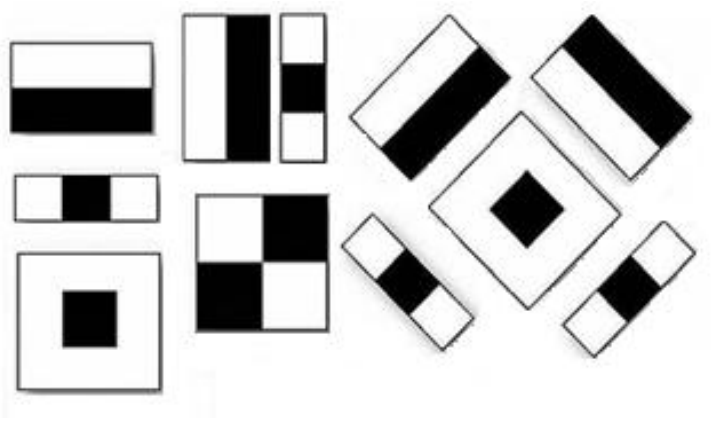


Рисунок 2.1 – Примитивы Хаара [11]

Интегральное представление изображения – это матрица, размерность которой совпадает с размерностью исходного изображения. Элементы этой матрицы рассчитываются по следующей формуле:

$$II(x, y) = Sum(I(i,j)), \quad (2.1)$$

где I(i,j) – яркость пикселя исходного изображения.

Из формулы видно, что каждый элемент интегрального изображения II[x, y] содержит в себе сумму пикселей изображения в прямоугольнике от (0,0) до (x, y). Расчет интегрального изображения занимает линейное время, пропорциональное числу пикселей исходного изображения.

В данном блоке используется класс *CascadeClassifier* библиотеки EmguCV. Для работы алгоритма необходимы файлы обученных каскадов. Для обучения необходимо не менее 1000 положительных и столько же отрицательных примеров и около суток для обучения. Результат обучения хранится в xml файле, который и используется данным блоком.

2.5. Блок стандартизации изображений

Блок стандартизации изображений необходим для преобразования изображений, полученных на этапе детектирования, к одному формату. Это необходимо для проведения последующей операции классификации. Необходимо привести изображения к одному размеру, выровнять яркость.

2.6 Блок классификации

Блок классификации представляет собой сверточную нейронную сеть (СНС), которая после обучения на подготовленной выборке производит классификации дорожных знаков на найденных на предыдущем этапе прямоугольных фрагментах изображения. СНС предназначены для эффективного распознавания образов на основе заранее подготовленной базы. Здесь СНС на основе обучающей выборки при подаче изображения будет определять, с какой вероятностью знак на изображении принадлежит к тому или иному классу и на основе этих значений будет выбираться наибольшее, что, скорее всего, означает принадлежность к этому классу. Каждая СНС состоит из нескольких слоев. Среди них могут быть:

1. Входной. На данный слой поступает каждый пиксель исходного изображения. Количество нейронов на данном слое должно быть равно количеству пикселей на классифицируемом изображении. Данный слой

выполняет только одну задачу – распределение входных сигналов остальным нейронам. Нейроны этого слоя не производят никаких вычислений;

2. Свертки. Этот слой является основным и обязательным для данного типа нейронной сети. Слой свертки включает в себя для каждого канала свой фильтр, ядро свертки которого обрабатывает предыдущий слой по фрагментам (суммируя результаты матричного произведения для каждого фрагмента). Весовые коэффициенты ядра свертки неизвестны и устанавливаются в процессе обучения;

3. Слой активации. Слой активации обычно логически объединяют со слоем свертки. Результат каждой свертки в скалярном виде попадает на функцию активации в данном слое. Функция активации же представляет собой некую нелинейную функцию. Функция нелинейности в свою очередь может быть любой, традиционно для этого используется функция типа гиперболического тангенса ($f(x) = \tanh(x), f(x) = |\tanh(x)|$);

4. Пулинг или слой субдискретизации. Данный слой представляет собой нелинейное уплотнение карты признаков, при этом группа пикселей (обычно размера 2×2) уплотняется до одного пикселя, проходя нелинейное преобразование. Чаще всего для этого используется функция максимума. Операция пулинга позволяет существенно уменьшить пространственный объем изображения. Смысл операции пулинга можно описать так: если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного;

5. Выходной. Количество нейронов в этом слое, как правило, равно количеству определяемых классов. При этом устанавливается соответствие между выходом нейронной сети и классом, который он представляет. Когда сети предъявляется некий образ, на одном из ее выходов должен появиться признак того, что образ принадлежит этому классу. В тоже время на выходах остальных нейронов должны быть признаки того, что образ данному классу не принадлежит.

К преимуществам данного типа нейронной сети можно отнести:

–лучшие показатели в распознавании и классификации изображений;

–гораздо меньшее количество настраиваемых весов (по сравнению с полносвязной нейронной сетью). Используется одно ядро весов для всего изображения. Это приводит к обобщению демонстрируемой информации, а не попиксельному запоминанию каждой показанной картинки, что улучшает качество распознавания;

– удобное распараллеливание вычислений, а, следовательно, возможность использования графических процессоров для ускорения работы;

– относительная устойчивость к повороту и сдвигу распознаваемого изображения;

– обучение при помощи классического метода обратного распространения ошибки.

Недостатком же является большое количество варьируемых параметров сети. Что приводит к сложности конфигурации сети для определенной поставленной задачи. Существует несколько выверенных и прекрасно работающих конфигураций сетей, но не хватает рекомендаций, по которым нужно строить сеть для новой задачи.

2.7 Блок получения координат

Блок получения координат содержит необходимые методы получения географических координат дорожных знаков, используя исходные видеозаписи. В исходных видеозаписях каждый кадр имеет соответствующую запись о координатах транспортного средства в момент записи. Эти данные хранятся в субтитрах и, проведя соответствующее преобразование, используя их можно определить координаты съемки определенного кадра. Используя утилиту ffmpeg и необходимые аргументы можно получить субтитры в текстовом формате txt, в котором каждая запись содержи текущее время записи, географические координаты и время, когда данных дорожный знак был обнаружен на видеозаписи.

2.8 Блок приведения полученных данных к одному формату

Блок приведения полученных данных к одному формату необходим для стандартизации данных, полученных при обработке видеозаписей в ходе работы программы, таких как название обнаруженного дорожного знака, его географических координат и времени его обнаружения к формату, который пригоден для экспортирования в базу данных. Каждый подготовленный элемент содержит название знака, его координаты и время создания видеозаписи, обработка которого была произведена, что позволит контролировать изменение дорожных знаков на определенных участках дорог.

2.9 Блок экспортирования результатов в базу данных

Блок экспортирования результатов в базу данных содержит программный код для сохранения полученных результатов в реляционную sql базу данных. Пользователь указывает место хранения данных результатов предыдущего блока. Для сохранения результатов работы программы будет использоваться СУБД Oracle. При попытке записи в базу данных дорожного знака, который был обнаружен повторно, будет произведено обновление времени его обнаружения без полной перезаписи.

3. 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.

3.1.Классы разрабатываемого программного средства

3.1.1. 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 с 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.

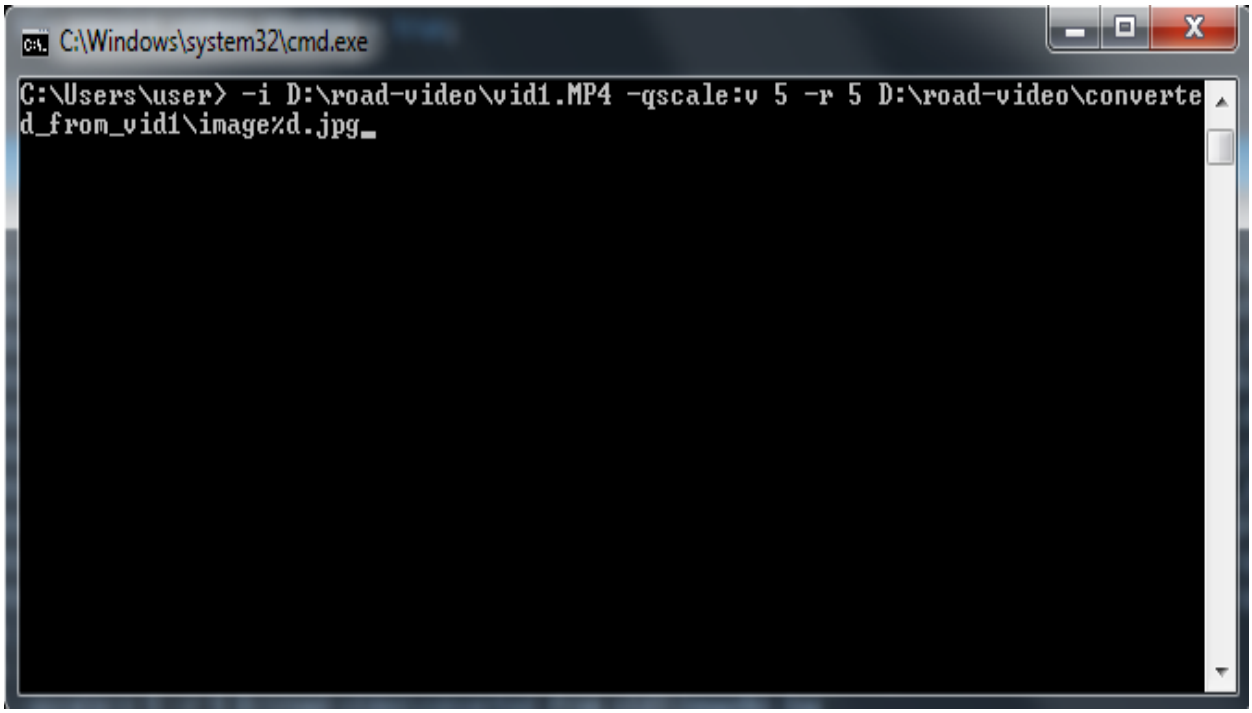


Рисунок 3.1 – Пример использования программы ffmpeg для разбиения видеозаписи на кадры

Метод `convertVidToSubs` создает новый процесс, в котором через консоль Windows вызывает программу ffmpeg с 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

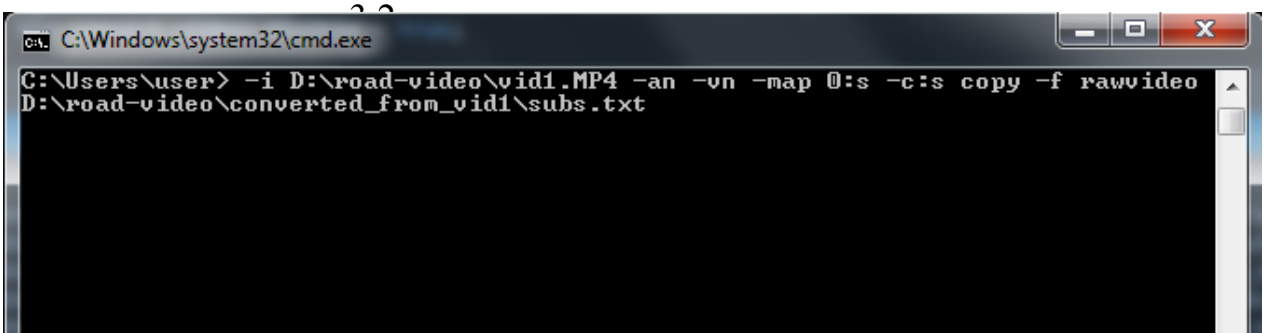


Рисунок 3.2 – Пример команды ffmpeg для получения субтитров из видеозаписи

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.

3.1.2. 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.

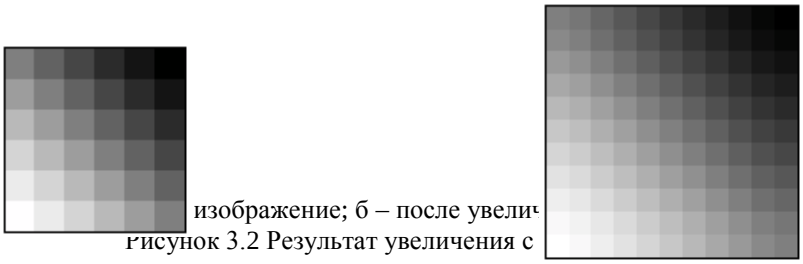
Метод `RGBtoHSV` преобразует изображение из цветового пространства RGB в цветовое пространство HSV. Он принимает изображение в цветовом пространстве RGB и возвращает изображение в цветовом пространстве HSV.

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).



изображение; б – после увели
Рисунок 3.2 Результат увеличения с

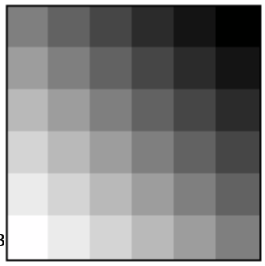


Рисунок 3.3 – Результат ув яции

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.

3.1.3. 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.

3.1.4. 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 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.

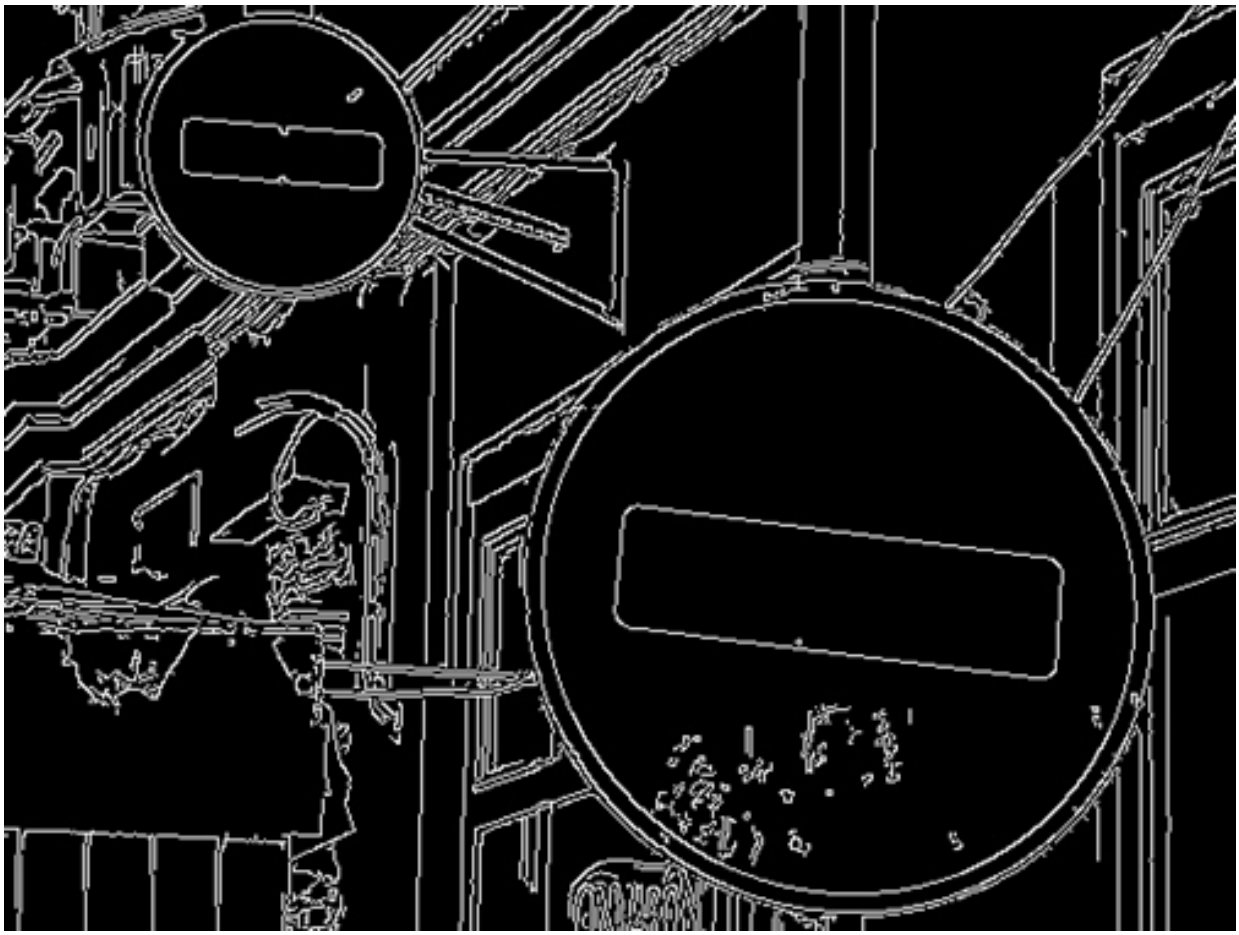


Рисунок 3.4 – Результат применения детектора границ Кэнни

3.1.5. 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.

3.1.6. 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.

3.1.7. 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.

3.1.8. 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.

3.1.9. 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.

3.1.10. 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.

3.1.11. 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.

3.1.12. 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.

3.1.13. 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.

3.1.14. 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.

3.1.15. 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.

3.1.16. 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`.

3.1.17. Класс ShapeDetection

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

Метод `detectShapes` принимает 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.

3.1.18. 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.

3.1.19. 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.

3.1.20. 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).

3.1.21. 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.

3.1.22. 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.

3.1.23. 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.

3.1.24. 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.

3.1.25. 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.

3.2. Структура организации Properties.Settings

Организация Properties.Settings – это файл формата xml, который можно найти в папке с 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

3.3. Экспорт результатов в базу данных

3.3.1. 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 `dataSource` указывает 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.

3.3.2. 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.

Поле `roadKm` 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.

3.4. 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

ЗАКЛЮЧЕНИЕ

Во время работы над преддипломной практикой была изучена предметная область, детально рассмотрены алгоритмы обнаружения и классификации объектов на изображениях. Кроме того, был получен опыт работы с прикладным программным обеспечением в области работы с разнообразными форматами и обработки изображений, машинного обучения и нейронных сетей. Для сохранения и учета результатов обработки исходных видеозаписей были изучены методы работы с базами данных Oracle.
Также в ходе преддипломной проектирования была разработана структурная схема, диаграмма классов, диаграмма последовательности и модель данных реализуемого программного средства. Благодаря системному подходу к проектированию возможно дальнейшее ее улучшение, исправление пока необнаруженных ошибок и расширение функциональности в целом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Автомобильные дороги в Беларуси [Электронный ресурс]. – Режим доступа: https://www.belta.by/infographica/view/dorogi-belarusi-7061 – Дата доступа: 12.04.2019.

[2] RoadAR Google Play [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=ru.roadar.android – Дата доступа: 13.04.2019.

[3] Знаки дорожного движения [Электронный ресурс]. – Режим доступа: https://pdd.am.ru/road-signs/ – Дата доступа: 13.04.2019.

[4] Общие сведения о Windows Forms [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/dotnet/framework/winforms/windows-forms-overview – Дата доступа: 14.04.2019.

[5] Обзор FFMPEG [Электронный ресурс]. – Режим доступа: https://andreyv.ru/ffmpeg.html – Дата доступа: 13.04.2019.

[6] Википедия свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org –Дата доступа: 10.04.2019.

[7] www.omega.ru [Электронный ресурс]. – Режим доступа: http://www.omega.ru/oracleinfo.html – Дата доступа: 16.04.2019.

[8] Руководство по программированию в Windows Forms [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/windowsforms – Дата доступа: 15.04.2019.

[9] Towards Data Science [Электронный ресурс]. – Режим доступа: https://towardsdatascience.com – Дата доступа: 14.04.2019.

[10] FFMPEG official website [Электронный ресурс]. – Режим доступа: https://ffmpeg.org – Дата доступа: 11.04.2019.

[11] EmguCV wiki [Электронный ресурс]. – Режим доступа: http://www.emgu.com/wiki/index.php/Main_Page – Дата доступа: 10.04.2019.

[12] Habrahabr [Электронный ресурс]. – Режим доступа: https://habr.com/ru – Дата доступа: 16.04.2019.

ПРИЛОЖЕНИЕ А

Вводный плакат

ПРИЛОЖЕНИЕ Б

Заключительный плакат

ПРИЛОЖЕНИЕ В

Диаграмма классов

ПРИЛОЖЕНИЕ Г

Диаграмма последовательности

ПРИЛОЖЕНИЕ Д

Модель данных

ПРИЛОЖЕНИЕ Е
Исходный текст программы

ПРИЛОЖЕНИЕ Ж

Ведомость документов

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Логика взаимодействия пользователя и интерфейса приложения

4.1.1 Нажатие кнопки открытия папки с видеозаписями

После того, как пользователь нажал на данный элемент графического интерфейса, происходят следующие действия:

1. Производится вызов диалогового окна, в котором пользователь производит выбор пути к директории, в которой находятся видеозаписи, которые необходимо обработать;
2. Происходит проверка пути на существование;
3. Если путь существует, производится создание объекта класса `VideoFolder`, иначе выводится соответствующее сообщение и переход к шагу номер 7;
4. Затем выполняется считывание всех путей к видеозаписям в выбранной директории;
5. Вычисляется количество найденных видеозаписей и их количество отображается на графической форме;
6. Выбранный путь сохраняется в настройки приложения;
7. Возврат из вызова функции.

4.1.2 Нажатие кнопки выбора директории для сохранения полученных после преобразования изображений

После того, как пользователь нажал на данный элемент графического интерфейса, происходят следующие действия:

1. Производится вызов диалогового окна, в котором пользователь производит выбор пути к директории, в которой будут сохранены полученные узображения;
2. Происходит проверка пути на существование;
3. Если путь существует, производится сохранение данного пути в настройках приложения и отображения его в соответствующем элементе графической формы, иначе выводится соответствующее сообщение;
4. Возврат из вызова функции.

4.1.3 Нажатие кнопки преобразования видеозаписей

После того, как пользователь нажал на данный элемент графического интерфейса, происходят следующие действия:

1. Проверяется выбранный путь к папке с видеозаписями;
2. Если путь пуст – выводится соответствующее сообщение и переход к шагу номер 11;

3. Если путь для сохранения полученных изображений пуст – выводится соответствующее сообщение и переход к шагу номер 11;

4. Пользователь оповещается о начале обработки и просьба подождать ее завершения;

5. Затем для каждого пути к видеозаписи из выбранной директории создается экземпляр класса `FFMPEGConverter` и производится вызов метода `ConvertAll` этого же класса;

6. После этого вычисляется количество папок для дальнейшего поиска дорожных знаков;

7. Затем это количество отображается на графической форме;

8. Возврат из вызова функции.

4.2 Преобразование видеозаписей в коллекцию изображений

Для преобразования базы видеозаписей к единому формату, удобному для дальнейшей обработки используется класс `FFMPEGConverter`. Здесь происходит разбиения видеозаписи на кадры, получения субтитров, содержащих записи координат и времени видеосъемки.

4.2.1 Разбиение видеозаписи на кадры

Для того чтобы преобразовать видеозапись в папку с изображениями используется функция `ConvertVidToImages` из класса `FFMPEGConverter`. Функция выполняет следующие шаги:

1. Создается переменная флага результата операции;

2. Если видеозапись по заданному существует выполняется шаг 3, если не существует – флаг результата устанавливается в нулевое значение;

3. Для каждой видеозаписи создается директория, в которую будут сохранены полученные изображения;

4. Высчитывается частота преобразования, равная отношению одной секунды в миллисекундах к числу отсчетов;

5. Создается переменная, содержащая параметры запуска утилиты ffmpeg, в которую записываются путь к видеозаписи, частота преобразования, путь к папке, в которую необходимо сохранить полученные изображения;

6. Создается и инициализируется переменная процесса для преобразования;

7. Устанавливаются параметры процесса: название запускаемого приложения(ffmpeg), аргументы запуска, режим с отображением консоли на экран для контролирования стадии преобразования;

8. Процесс запускается;

9. Программа ожидает завершения работы процесса;

10. Переменная флага результата устанавливается в единичное значение;

11. Возврат флага результата.

4.2.2 Получение субтитров из видеозаписи

Для получения субтитров, содержащих географические координаты и время записи, используется функция `ConvertVidToSubs` из класса `FFMPEGConverter`. Функция выполняет следующие шаги:

1. Создается переменная, в которой хранится путь к текстовому файлу, в котором будут находится полученные субтитры;
2. Создается переменная, содержащая параметры запуска утилиты ffmpeg, в которую записываются путь к видеозаписи, частота преобразования, путь к папке, в которую необходимо сохранить полученные изображения;
3. Создается и инициализируется переменная процесса для преобразования;
4. Устанавливаются параметры процесса: название запускаемого приложения(ffmpeg), аргументы запуска, режим с отображением консоли на экран для контролирования стадии преобразования;
5. Процесс запускается;
6. Программа ожидает завершения работы процесса;
7. Возврат строки, содержащей путь к файлу с субтитрами.

4.2.3 Считывание файла с субтитрами и его преобразование к единому виду

После получения субтитров из видеозаписи необходимо преобразовать записи из текстового файла к формату, пригодному для дальнейшей обработки. Для этого используется функция `ParseSubtitleFile` из класса `FFMPEGConverter`. Работа функции состоит из следующих шагов:

1. Происходит проверка текстового файла по введенному пути на предмет его существования.
2. Создается переменная строчного типа, в которую происходит считывание всего текстового файла;
3. Создается регулярное выражение для разбиения всей строки на отдельные записи, соответствующие каждому кадру видеозаписи;
4. С помощью регулярного выражения из строки получается коллекция записей;
5. Создается список, в котором будут находится географические координаты распознаваемых знаков;
6. Далее в цикле с помощью функции `ParseMovementPoint` этого же класса производится преобразование каждой полученной записи в объект класса `MovementPoint`, который хранит географические координаты;
7. Каждый полученный объект класса `MovementPoint` добавляется в список координат;
8. Возврат заполненного списка координат.

4.2.4 Функция создания объекта, содержащего географические координаты

Для преобразования части строки, которая содержит географические координаты к объекту вызывается функция `ParseMovementPoint`. Данная функция работает следующим образом:

1. Сначала часть текстового файла разбивается на отдельные записи, которые в файле разделены между собой запятыми;
2. После этого для каждой записи производится преобразование даты в удобный для восприятия формат;
3. Затем производится преобразование географической широты в десятичный формат с помощью вызова функции `ConvertDegreesAndDecimalMinutesStringToDecimal`, логика работы которой описана ниже;
4. То же самое выполняется и для переменной, отражающей географическую долготу;
5. И в конце полученный результат преобразуется в объект класса movementPoint;
6. Возврат полученного объекта из функции.

4.2.5 Функция преобразования географических координат к десятичному формату

Для того, чтобы привести координаты к единому десятичному формату, удобному для применения, используется функция `ConvertDegreesAndDecimalMinutesStringToDecimal`. Работает функция следующим образом:

1. Изначально координаты хранятся в дробном виде, разделенном точкой;
2. Вычисляем целые градусы;
3. Затем вычисляем целые минуты;
4. После этого приводим всё к целому количеству градусов, учитывая, что в одном градусе шестьдесят минут.
5. Возвращаем полученное значение.

4.3 Поиск контуров на изображении

Для поиска контуров геометрических фигур на изображении используются методы класса `ShapeDetection`.

Метод `detectShapes`, позволяющий находить и выделять контуры геометрических фигур, таких как: треугольник, квадрат, прямоугольник, шестиугольник, круг, работает следующим образом:

Создается объект `countors` класса `VectorOfVectorOfPoint`, который будет хранить результаты поиска контуров;

1. Выполняется сглаживание введенного изображения с помощью фильтра Гаусса;

Затем вызывается метод `FindCountors` из библиотеки `EmguCV`, который выполняет поиск контуров геометрических фигур на изображении и возвращает объект класса `VectorOfVectorOfPoint`;

2. После этого для каждого найденного контура из массива countors выполняются следующие шаги;
3. Вычисляется периметр контура;
4. Производится аппроксимация контура для выявления прямоугольных границ найденной фигуры;
5. Затем рисуется рамка, ограничивающая найденную фигуру;

6. После этого вычисляется количество линий в контуре, для распознавания геометрической фигуры (3 грани – треугольник, 4 – четырехугольник, 6 – шестиугольник, и так далее).

7. Далее название найденной фигуры пишется на изображении;

8. И в конце производится возврат полученного изображения с нарисованными границами и названиями геометрических фигур.

Для поиска определенного вида геометрических фигур предназначен метод `detectShape` класса `ShapeDetection`. Данный метод работает следующим образом:

1. Сначала полученное изображение переводится в оттенки серого цвета;

2. Затем выполняется сглаживание для уменьшения количества помех на исходном изображении;

3. Далее для поиска окружностей вызывается метод `HoughCirlces` из библиотеки `EmguCV`;

4. Результат поиска заносится в массив, содержащий объекты класса Circles;

5. После этого происходит преобразование исходного изображения с помощью вызова функции `Canny`, который работает на основе алгоритма Кэнни;

6. Затем для поиска линий вызывается метод `HoughLinesP`, результат работы которого заносится в массив `lines` класса `LineSegment2D`;

7. Далее для поиска треугольников и четырехугольников выполняется вызов функции `FindCountors`, результат работы которого заносится в созданный массив `countors` класса `VectorOfVectorOfPoint`;

8. Затем для каждого найденного контура выполняются следующие действия;

9. Производится аппроксимация контура для выявления прямоугольных границ найденной фигуры;

10. Затем необходимые контуры изображаются на исходном рисунке;

11. Метод возвращает полученное изображение, которое содержит контуры определенных геометрических фигур;

4.4 Поиск совпадений на нескольких изображениях

1. Для поиска одного изображения на другом изображении написан класс `ShapeComparation`.Единственный метод `FindMatch`, который и производит поиск признаков одного изображения на другом работает так:

2. Создаются объекты класса `VectorOfKeyPoint` для каждого из изображений;

3. Создается экземпляр класса `KAZE` из библиотеки `EmguCV`, в котором находится программная реализация алгоритма поиска подобных областей на изображениях с помощью алгоритма `KAZE`;

4. Производится поиск ключевых точек на одном изображении и вычисляется его дескриптор;

5. Затем те же действия выполняются для другого изображения;

6. После этого создается объект класса `DescriptorMatcher` для поиска общих областей между изображениями;

7. С помощью этого объекта вызывается метод KnnMatch для поиска совпадающих областей;

8. Производится отсеивание повторяющихся совпадающих областей;

9. Производится отсеивание совпадающих областей, ориентация которых не совпадает с большинством других областей;

10. Выделяются найденные общие области;

11. Возврат из функции;

4.5 Поиск дорожных знаков с помощью каскадов Хаара

После получения необходимых изображений и координат для проведения операции детекции используются методы класса `DetectFolder` и `SignsHaarCascade`.

Методы класса `DetectFolder` позволяют производить поиск дорожных знаков на совокупности фотографий, найденных в определенной директории.

Метод `DetectAll` класса `DetectFolder` работает следующим образом:

1. Создается коллекция, содержащая каскады, с помощью которых будет происходить поиск знаков;

2. Создается список, в котором будут находится пути ко всем папкам с изображениями, которые были найдены в введенной директории;

3. Происходит заполнение коллекции каскадов, путем создания объектов класса `SignsHaarCascade`;

4. В цикле каждое изображение приводится к единому формату `Mat`;

5. Для каждого каскада из списка вызывается метод `DetectAll` класса `SignsHaarCascade`, работа которого описана ниже;

6. При нахождении хотя бы одного дорожного знака, по введенному для сохранения полученных результатов пути производится создание директории с соответствующим названию исходного изображения названием;

7. Производится сохранение по заданному пути результатов поиска;

8. Возврат из функции.

4.5.1 Поиск дорожных знаков с помощью каскадов Хаара

Поиск знаков на определенном изображении с помощью определенного каскада выполняется с помощью метода `DetectAll`.Работа метода `DetectAll` класса `SignsHaarCascade` происходит по следующему алгоритму:

1. Создается список, в котором будут находится части изображений, содержащих дорожные знаки;

2. С помощью каскада и вызова функции `DetectMultiScale` из библиотеки EmguCV производится поиск регионов на изображении, которые содержат дорожные знаки;

3. Для каждого найденного региона создается изображение и добавляется в список;

4. Функция возвращает полученный список.

4.6 Преобразование изображений к единому формату

Для того, чтобы изображения можно было классифицировать с помощью нейронной сети их необходимо привести единому виду:
– Размер 32 на 32 пикселя;
– Единственный цветовой канал (оттенки серого);
– Выровненное по контрасту.
Для данных целей используются следующие методы класса `ImgOps`.

4.6.1 Изменение размеров изображения

Для того, чтобы размеры изображений были одинаковы и равны 32 на 32 пикселя используется метод `InterpolationResize`. Данный метод вызывает функцию `Resize` из библиотеки EmguCV. В нее передаются: само изображение, желаемый размер изображения после преобразования и выбирается метод интерполяции. В данном случае выбирается бикубическая интерполяция. Метод возвращает полученное изображение размером 32 на 32 пикселя.

4.6.2 Изменение цветового пространства изображения

После изменения размеров изображения необходимо уменьшить число цветовых каналов для увеличения скорости классификации. Для этого используется метод `RGBtoGrey`, который из цветного изображения получает изображение в градациях серого. Данный метод работает так:

Сначала производится преобразование изображение в объект класса `Image<Rgb, Byte>`;

Затем вызывается метод `Convert<Gray, Byte>`, который выполняет непосредственно само преобразование;

1. Метод возвращает полученное изображение в градациях серого.

4.6.3 Контрастное выравнивание изображения

Последней стадией подготовки к классификации изображения является контрастное выравнивание. Оно необходимо для того, чтобы избежать появления неосвещенных и на оборот слишком освещенных областей на изображении, что могло бы привести к ухудшению качественных характеристик классификации нейронной сети. Для этого используется метод `ContrastAlignment`. Данный метод вызывает функцию `CLAHE` из библиотеки EmguCV. В нее передаются: само изображение, фактор контрастности, который предотвращает перенасыщенность изображения,

особенно в однородных областях, а также размер окна фильтра. Метод возвращает полученное изображение.

4.7 Нейронная сеть

Классификация подготовленных изображений происходит с помощью сверточной нейронной сети, программная реализация которой находится в классе `CNN`.

4.7.1 Создание нейронной сети

Создание экземпляра программной реализации нейронной сети выполняется с помощью вызова метода `CreateCNN` класса `CNN`. Этот метод работает так:

1. Производится начальная инициализация параметров нейронной сети, а именно: точность, количество слоев, флаг обученности нейронной сети и путь к сохранению параметров сети;
2. Затем производится добавление слоев нейронной сети: входного слоя, сверточных слоев, слоев пулинга, активации, полносвязных слоев, слоя `softmax` с соответствующими параметрами для каждого из этих слоев;
3. После этого сохраняется количество классов распознаваемых дорожных знаков;
4. Возврат из функции.

4.7.2 Обучение нейронной сети

Обучение сверточной нейронной сети (вычисление подходящих весовых коэффициентов для нейронов) производится с помощью метода `TeachCNN` класса `CNN`. Этот метод принимает пути к обучающей и тренировочной выборкам, ожидаемую точность, скорость обучения, размер пачки изображений для обучения. В данном методе выполняются следующие шаги:

1. Производится проверка количества слоев: если количество слоев равно нулю, производится инициализация сети;
2. Выполняется создание обучающего и тестового наборов изображений;

Производится создание экземпляра класса `SgdTrainer<double>`, который является «учителем» для нейронной сети, который производит обучение методом градиентного спуска;

3. Задаются параметры «учителя», такие как скорость обучения и размер пачки изображений для обучения;
4. До тех пор, пока достигнутая точность меньше необходимой выполняются следующие действия;

5. Создается пачка из обучающей выборки с заранее заданным размером;
6. Каждое изображение пропускается через сеть для проверки и корректировки весовых коэффициентов;
7. Затем создается пачка из тестовой выборки с заранее заданным размером;
8. Каждое изображение пропускается через сеть для тестирования и вычисления достигнутой точности распознавания;
9. Выводятся достигнутая точность и номер текущей итерации обучения;
10. Конец цикла;
11. Флаг обученности сети выставляется в единичное состояние;
12. Возврат из функции.

4.7.3 Сохранение данных нейронной сети

Сохранение данных нейронной сети позволяет избежать постоянного проведения обучения с помощью сохранения результатов в текстовый файл с возможностью последующей их загрузки. Сохранение данных выполняется с помощью вызова метода `SaveCNN`. Данный метод работает следующим образом:

1. Производится проверка количества слоев: если количество слоев равно нулю, выполняется возврат из функции;
2. Производится проверка флага обученности нейронной сети: если количество слоев равно нулю, выполняется возврат из функции;
3. Выполняется преобразование параметров обученной сети к строковому формату пригодному для сохранения результатов;
4. Выполняется создание переменной, в которой хранится название файла, в которое включены ключевые параметры нейронной сети;
5. Если файл с таким названием существует, производится его удаление;
6. Выполняется создание нового текстового файла с заранее определенным названием;
7. Производится запись данных нейронной сети в текстовый файл;
8. Возврат из вызова функции.

4.7.4 Загрузка данных нейронной сети

Загрузка данных ранее обученной нейронной сети выполняется с помощью вызова функции `LoadCNN`. Данный метод работает следующим образом:

1. Сначала производится проверка существования текстового файла пути загрузки;
2. Затем происходит считывание всего текстового файла в строковую переменную;

3. После этого выполняется попытка преобразования строки к виду экземпляра нейронной сети;

4. Если это не удалось, выполняется возврат из вызова функции с результатом равным минус одному, иначе выполняется шаг под номером 5;

5. Производится замена текущего экземпляра класса, загруженным из файла;

6. Метод возвращает количество слоев в загруженной нейронной сети;

4.7.5 Загрузка выборок для обучения нейронной сети

Загрузка выборок осуществляется с помощью вызова функции `Load` класса `DataSets`. Работа функции заключается в следующем:

1. Выполняется считывание по заданному пути папки с тренировочными изображениями, используя метод `Load` класса `ImageReader`, логика работы которого описана ниже;

2. Выполняется считывание по заданному пути папки с тестовыми изображениями и файлом номеров классов, соответствующих этим изображениям, используя метод `Load` класса `ImageReader`, логика работы которого описана ниже;

3. Производится проверка количества загруженных изображений;

4. Если не загружено ни одного изображения, то производится возврат из вызова данной функции с результатом `false`, иначе выполняется шаг под номером 5;

5. Выполняется создание тренировочного набора данных из загруженной обучающей выборки;

6. Выполняется создание тестового набора данных из загруженной тестовой выборки;

7. Производится возврат из функции с результатом `true`;

4.7.6 Загрузка выборки из папки по определенному пути

Загрузка каждой выборки состоит из загрузки изображений, номеров классов этих изображений и объединение изображения с номером соответствующего ему класса. При каждой загрузке обучающей выборки происходит обновление файла с номерами классов. Это делается для возможности пополнения обучающей базы новыми изображениями и новыми классами изображений. Загрузка изображений производится по этому алгоритму:

1. Создается список, который будет содержать изображения в формате массива байт;

8. Выполняется создание экземпляра класса `ImageFolder` по введенному пути;

2. Каждое найденное изображение преобразуется в массив байт и добавляется в список;

3. Функция возвращает полученный список;

Обновление файла с номерами с последующей их загрузкой выполняется следующим образом:

1. Создается список, который будет содержать номера классов в целочисленном формате;
2. Проверяется наличие файла с таким же названием и если он существует – производится его удаление;
3. Создается новый файл для сохранения новых номеров классов изображений;
4. Для каждого изображения считывается его название;
5. Выбирается часть названия, которая включает в себя номер класса, к которому принадлежит изображение;
6. Полученный номер записывается в файл и добавляется в список;
7. Функция возвращает заполненный список;

4.8 Сохранение полученных результатов

Каждый найденный на изображении дорожный знак трансформируется в объект класса `Result` для дальнейшего сохранения на локальном хранилище и дальнейшего экспорта в базу данных.

Результаты работы программы являются массивом элементов вида Result.

Для сохранения результатов на локальном хранилище используется функция `ExportToFile` класса `ResultExport`. Данный метод записывает результаты в файл формата xml по введенному пути с помощью методов класса `XmlSerializer`. Запись результатов происходит следующим образом:

1. Создается экземпляр класса `XmlSerializer`;
2. Комбинируется путь к папке для сохранения результатов и имени файла;
3. С помощью потока записи в файл `FileStream` и метода `Serialize` класса `XmlSerializer` производится запись массива результатов;
4. Возврат из вызова функции.

4.9 Экспорт результатов в базу данных

Экспорт результатов в базу данных происходит после того, как результаты были сохранены на локальном хранилище. Для этого используются методы класса `ResultExport`.

4.9.1 Экспортирование папки с результатами

Сначала вызывается метод `ExportFolder`, который позволяет проводить экспорт всех файлов формата xml, которые были сохранены ранее

в выбранной директории. В него передается путь к директории. Все происходит следующим образом:

1. Сначала обновляется объект `xmlFolder` класса `XmlFolder`, в котором хранятся пути к файлам результатов;

2. Затем пути файлов загружаются в массив `files`;

3. После этого вызывается метод `ExportToDB` этого же класса, логика работы которого описана ниже;

4. Выполняется возврат из вызова функции.

4.9.2 Экспортирование файла с результатами

Экспортирование файла формата xml с результатами работы программы происходит с помощью вызова метода `ExportToDB`. Данный метод выполняет следующие действия:

1. Для считывания массива результатов из файла вызывается метод `UploadFromFile` этого же класса, логика работы которого описана ниже;

Затем создается объект `dt` класса `DataTable`, который является промежуточным звеном в цепочке экспортирования;

После этого добавляются все нужные столбцы таблицы `dt` в базе данных, которые соответствуют полям класса `Result`;

Затем каждый элемент массива добавляется в соответствующую строчку таблицы dt;

2. Далее производится попытка подключения к удаленной базе данных с помощью введенных параметров;

3. При успешном подключении таблица приводится к пригодному для экспортирования виду;

4. Далее с помощью консоли производится вызов программы для соединения с базой данных и занесение результатов;

5. В конце выполняется закрытие соединения;

6. Соединение с базой данных закрывается;

7. Возврат из вызова функции.

5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Испытание программного средство проводилось путем взаимодействия с приложением для выявления ошибок и несоответствий в заявленных функциональных возможностях и реальных возможностях готового продукта.

Объектом испытаний является программный продукт в том виде, в котором он предоставлен на прикрепленном компакт-диске.

5.1 Содержимое компакт-диска

На компакт-диске находятся следующие файлы:

1. Файл формата exe для установки программного средства;

2. Несколько файлов формата xml, в которых хранятся параметры каскадов Хаара для поиска дорожных знаков;

3. Файл формата txt, в котором находятся параметры обученной сверточной нейронной сети, состоящей из 13 слоев и способной классифицировать 11 видов дорожных знаков;

4. Файл с презентацией, описывающей основные возможности программного средства.

5.2 Тестирование программного средства

Тестирование программного средства проводилось путем выполнения всех действий, необходимых для полной обработки видеозаписи и сохранения полученных результатов.
Изображение главного экрана приложения находится на рисунке 5.1.

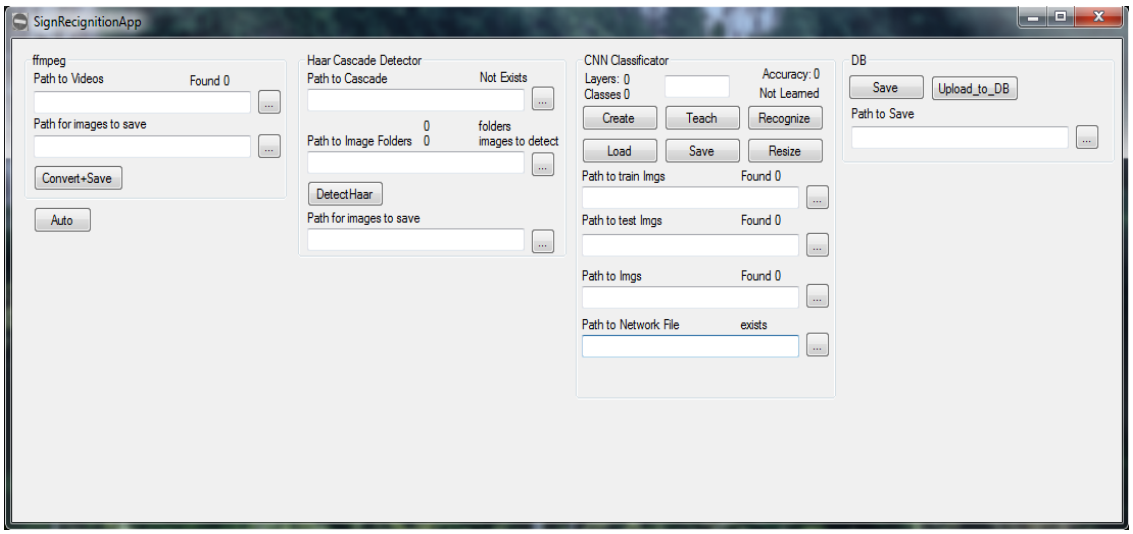


Рисунок 5.1 – Главный экран приложения

На главном экране можно увидеть иконку и название приложения, поля ввода путей к файлам, поле ввода точности для обучения нейронной сети, кнопки открытия директорий для указания путей к файлам и папкам, кнопки для выполнения других действий. Необходимо провести тестирование всех компонентов приложения для выявления допущенных ошибок.

5.2.1 Тестирование полей ввода данных

Поля ввода принимают любые значения и не выполняют проверку содержимого. При изменении значения поля ввода пути к файлу или папке выполняется попытка загрузки содержимого и при ее выполнении изменяется соответствующее поле, в котором отображается состояние этих файлов.

5.2.2 Тестирование кнопок

Тестирование кнопки выбора пути к папке с видеозаписями проводилось методом проверки реакции на те, либо иные действия и сравнения с ожидаемой реакцией.

Открытие папки, с видеозаписями правильного формата.

6 Руководство пользователя

Разработанное программное средство представляет собой desktop приложение для операционной системы (ОС) Windows.

6.1 Конфигурация системы

В связи с тем, что приложение разработано для запуска на ОС семейства Windows не старше 7 версии, для работы программы достаточно конфигурации ПК, способной производить запуск и использование ОС Windows 7. Также принимая во внимание то, что приложение разработано используя .Net Framework 4.5 для его запуска необходимо наличие установленной платформы не старше этой версии. В программе также используется набор программ ffmpeg version N-93565-g0ad0533e91, который должен быть установлен для полноценной работы программы.

6.2 Установка приложения

Для установки приложения необходимо запустить файл .exe расположенный на прикрепленном компакт-диске. Иконка файла установки приложения изображена на рисунке 6.1.

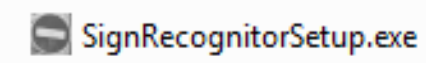


Рисунок 6.1 – Иконка установочного файла приложения

После запуска приложения появляется начальное окно установки (см. рисунок 6.2) на котором производится выбор языка установки (по умолчанию установлен язык ОС). После выбора языка нажимаем на кнопку «ОК».

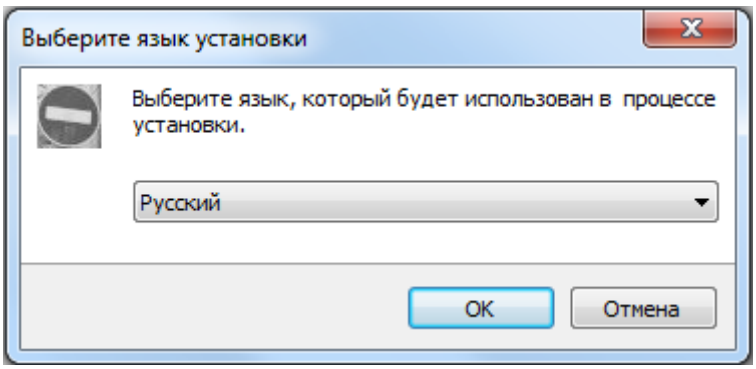


Рисунок 6.2 – Экран выбора языка установки

Затем отображается экран выбора пути установки приложения. Здесь можно увидеть название и устанавливаемую версию программного продукта, путь установки приложения и требуемое свободное место для проведения установки (см. рисунок 6.3).

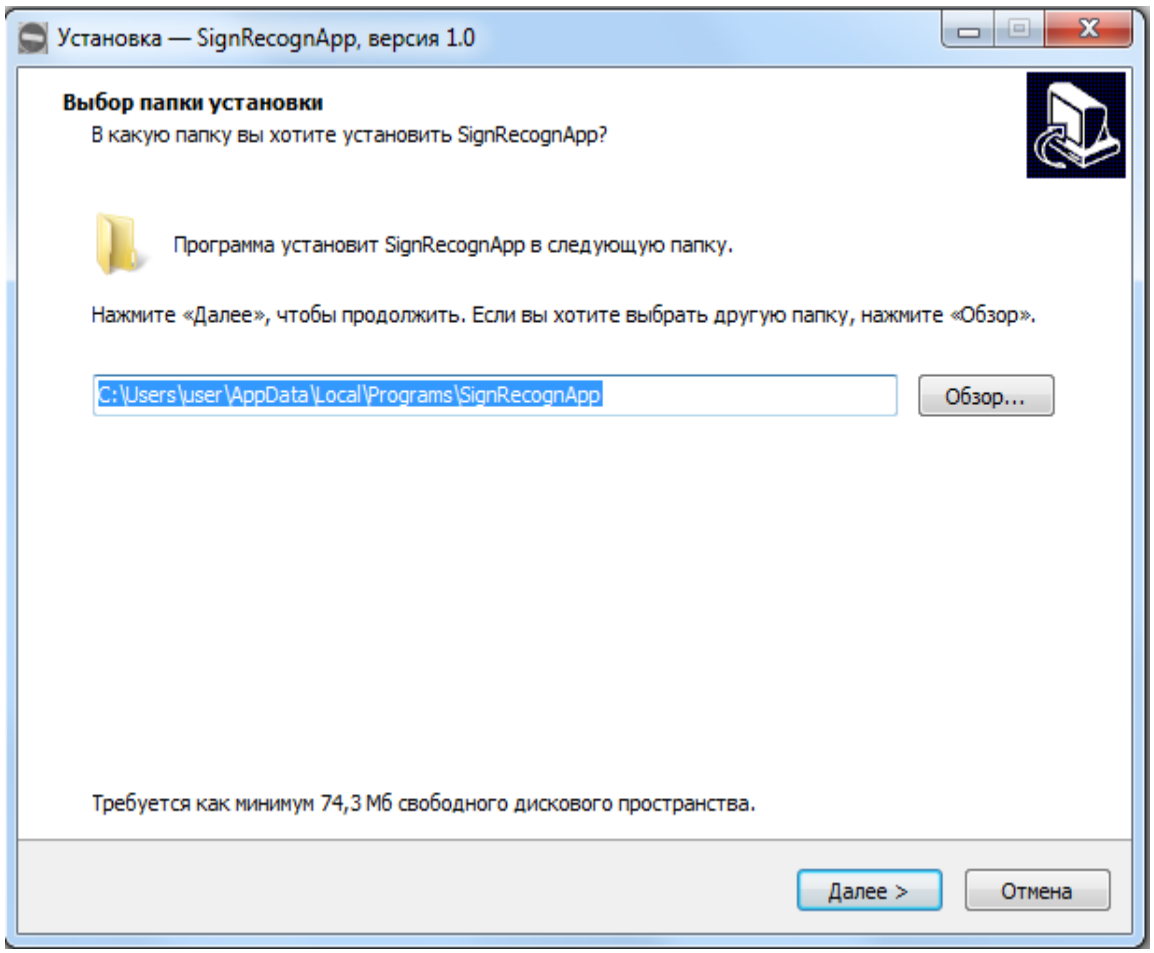


Рисунок 6.3 – Экран выбора пути установки приложения

После выбора языка нажимаем на кнопку «Далее».

Далее появляется экран, на котором предлагается возможность создания ярлыка приложения на рабочем столе (см. рисунок 6.4). После этого нажимаем кнопку «Далее».

После этого отображается экран отображения настроек установки приложения, на котором видны все параметры установки перед проведением непосредственной установки программного продукта (см. рисунок 6.5). После проверки всех выбранных параметров установки нажимаем кнопку «Установить».

Следующим шагом установки является экран отображения состояния, на котором можно увидеть текущую стадию установки, текущий устанавливаемый файл. На данном этапе производится установка компонентов программы и необходимые для ее запуска библиотеки. Изображение данного экрана отображено на рисунке 6.6.

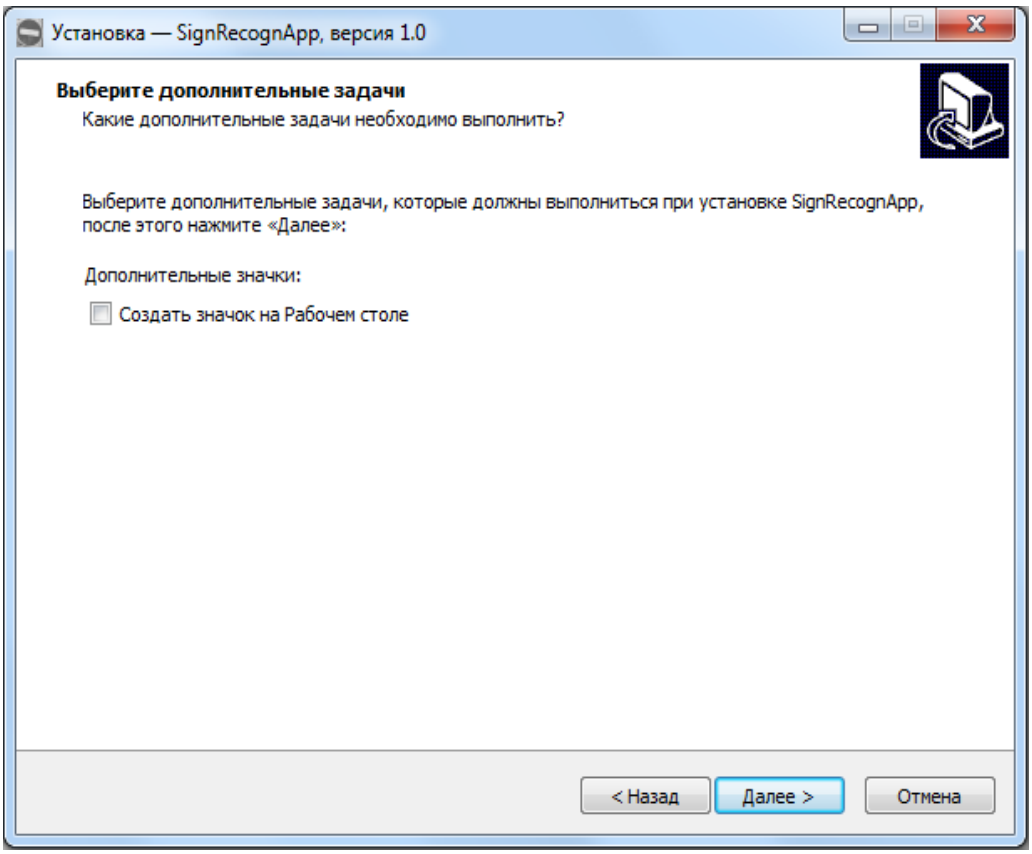


Рисунок 6.4 – Экран создания ярлыка на рабочем столе

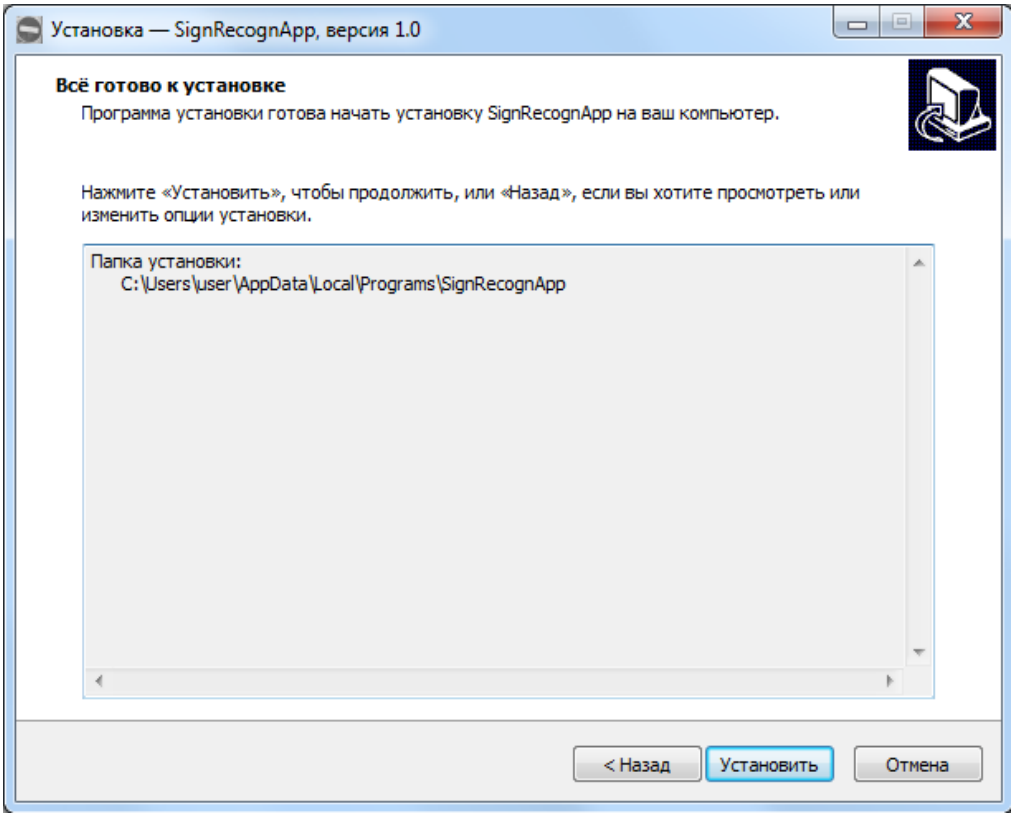


Рисунок 6.5 – Экран создания ярлыка на рабочем столе

Последним отображаемым экраном является экран результатов установки, на котором предлагается возможность запуска приложения сразу после закрытия экрана установки (см. рисунок 6.7). Нажимаем кнопку «Завершить».

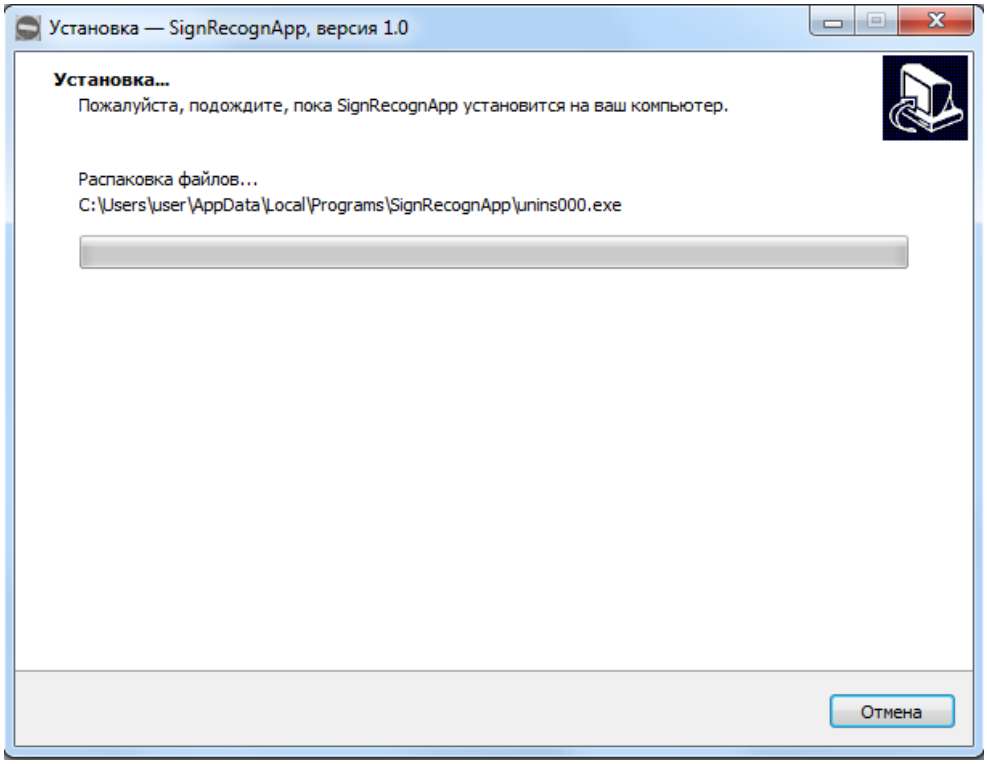


Рисунок 6.6 – Экран установки

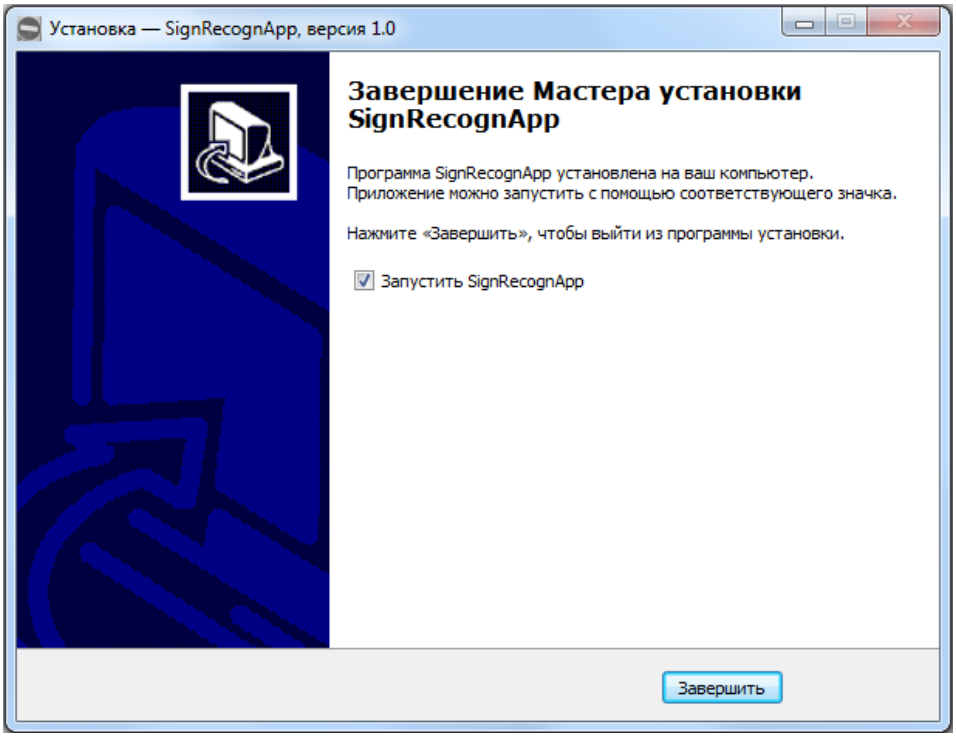


Рисунок 6.7 – Экран завершения установки

После выполнения установки, приложение может быть найдено по выбранному ранее пути. Также при выборе соответствующего пункта на рабочем столе будет создан ярлык для запуска программного продукта (см. рисунок 6.8).

Рисунок 6.7 – Ярлык запуска приложения с рабочего стола

6.3 Запуск приложения

Запуск приложения производится с помощью исполняемого файла формата exe из папки с установленным приложением, либо с помощью ярлыка на рабочем столе (выборе соответствующего пункта во время установки). Также произвести запуск установленного приложения можно через меню Пуск.

6.4 Работа с приложением

Приложению для работы необходима начальная настройка. Пользователю предстоит выбрать путь к набору видеозаписей и пути сохранения результатов работы программы. Обработка исходных данных производится в двух режимах: полуавтоматическом и автоматическом. В полуавтоматическом режиме самостоятельно производится запуск каждой стадии обработки данных, и указываются пути сохранения промежуточных данных. В автоматическом режиме работа возможна при инициализации лишь минимальных начальных настроек. Для работы приложения необходимы файлы конфигурации каскадов Хаара для поиска знаков и файл конфигурации сверточной нейронной сети. Их можно получить и с помощью данного средства, однако их производство будет занимать продолжительное время.

Изображение главного экрана приложения, на котором пронумерованы все элементы управления и состояния приложения, находится на рисунке 5.1. Там можно увидеть доступные элементы управления, такие как поля ввода и кнопки управления работой приложения. Также там существуют строки состояния выбранных директорий и файлов, а также состояния экземпляра программной реализации нейронной сети.

Управление работой приложения производится с помощью элементов управления. Далее будет приведена их короткая характеристика.

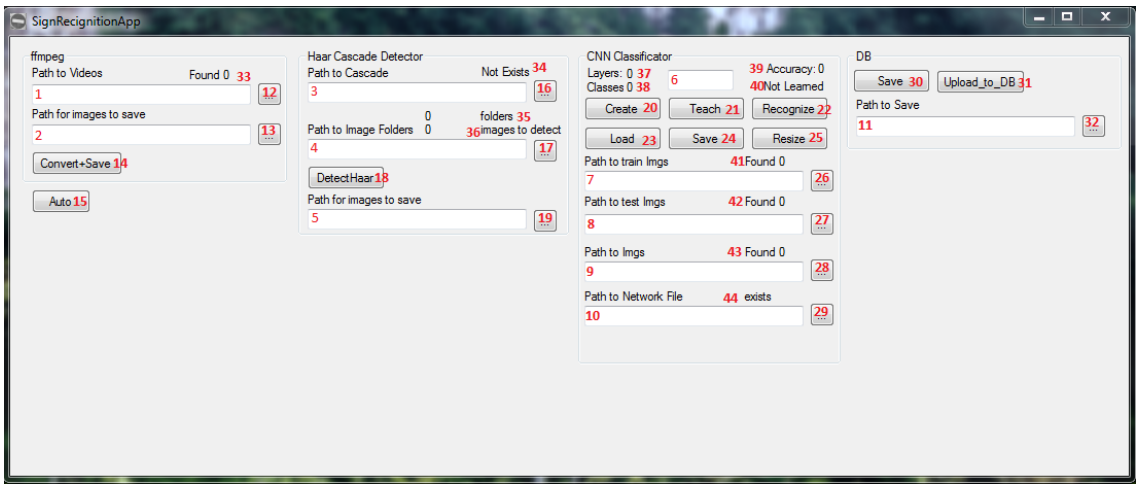


Рисунок 5.1 – Главный экран приложения

6.4.1 Описание полей ввода

Поле 1 используется для ввода и отображения пути к папке с исходными видеозаписями.

Поле 2 используется для ввода и отображения пути к папке, в которую необходимо сохранить полученные после преобразования видеозаписей изображения.

Поле 3 используется для ввода и отображения пути к файлам каскадов Хаара, с помощью которых будет производиться поиск дорожных знаков на полученных после преобразования изображениях.

Поле 4 используется для ввода и отображения пути к директории, в которой будет производиться считывание изображений для поиска дорожных знаков.

Поле 5 используется для ввода и отображения пути к папке, в которую необходимо сохранить полученные после поиска дорожных знаков частей изображений.

Поле 6 используется для ввода и отображения минимальной точности для обучения сверточной нейронной сети.

Поле 7 используется для ввода и отображения пути к папке с обучающими изображениями для настройки весовых коэффициентов сверточной нейронной сети.

Поле 8 используется для ввода и отображения пути к папке с тестовыми изображениями для настройки весовых коэффициентов сверточной нейронной сети.

Поле 9 используется для ввода и отображения пути к папке с изображениями, на которых необходимо провести классификацию найденных дорожных знаков.

Поле 10 используется для ввода и отображения пути к текстовому файлу формата txt, в котором находятся весовые коэффициенты и другие параметры сверточной нейронной сети для выполнения их загрузки в нейронную сеть.

Поле 11 используется для ввода и отображения пути к директории для сохранения результатов работы программы.

6.4.2 Описание кнопок управления

Кнопка 12 позволяет выбрать путь к папке с видеозаписями для проведения их дальнейшей обработки.

Кнопка 13 позволяет указать путь к папке, в которую будут сохранены полученные после преобразования изображения, разделенные на папки в соответствии с названиями исходных видеозаписей.

Кнопка 14 служит для вызова программы кадрирования видеозаписи и сохранения полученных изображений и файла с координатами по выбранному пути. Для запуска процедур поля 1 и 2 должны быть заполнены правильными путями к директориям, а папка с видеозаписями должна содержать хотя бы один файл соответствующего формата.

Кнопка 15 производит автоматическую обработку исходных видеозаписей, поиск дорожных знаков, их классификацию и сохранение результатов. Для автоматической работы необходимо иметь заполненные поля 1, 2, 3, 5, 10, 11, а также содержать хотя бы один файл соответствующего формата по пути в поле 1, хотя бы один файл каскада Хаара по пути 3, текстовый файл с параметрами сверточной нейронной сети по пути в поле 10.

Кнопка 16 позволяет выбрать путь к папке с каскадами Хаара в текстовых файлах формата txt для проведения операции детекции дорожных знаков на изображения.

Кнопка 17 позволяет выбрать путь к папке изображениями для проведения операции детекции дорожных знаков.

Кнопка 18 запускает процедуру поиска дорожных знаков с помощью каскадов Хаара на изображениях из папки по заданному пути. Для запуска процедуры поиска поля 3, 4 и 5 должны быть заполнены правильными путями к директориям. Папка с каскадами по пути в поле 3 должна содержать хотя бы один файл каскада формата txt. Папка с изображениями по пути в поле 4 должна содержать хотя бы одно изображение соответствующего формата.

Кнопка 19 позволяет указать путь к папке, в которую будут сохранены полученные после проведения поиска дорожных знаков фрагментов изображений.

Кнопка 20 производит запуск процедуры создания экземпляра программной реализации нейронной сети, добавление слоев и инициализацию начальных параметров.

Кнопка 21 производит запуск процедуры обучения экземпляра программной реализации нейронной сети с помощью тренера методом градиентного спуска, на основе базы тренировочных и тестовых изображений в оттенках серого и размером 32 на 32 пикселя. Если же база не соответствует данным параметрам, производится преобразование к стандартному виду. Для проведения обучения нейронная сеть должна быть инициализировано (в элементах 37 и 38 значения больше нуля) и поля 7 и 8 должны быть заполнены.

Кнопка 22 производит запуск процедуры классификации папки с изображениями, расположенной по выбранному пути, с помощью экземпляра программной реализации нейронной сети. Для запуска процедуры распознавания нейронная сеть должна быть заранее инициализирована и обучена, либо загружена (в поле 40 значение «learned»), также поле 9 должно содержать путь к папке с хотя бы одним изображением для классификации.

Кнопка 23 вызывает процедуру загрузки параметров нейронной сети из текстового файла, расположенного по пути в поле 10. Поле 10 должно быть заполнено.

Кнопка 24 вызывает процедуру сохранения параметров заранее обученной нейронной сети в текстовый файл по пути в поле 10. Поле 10 должно быть заполнено.

Кнопка 25 позволяет произвести изменение входных изображений к стандартному размеру.

Кнопка 26 позволяет указать путь к папке, из которой будут загружены обучающие изображения для выполнения процедуры обучения инициализированного экземпляра нейронной сети.

Кнопка 27 позволяет указать путь к папке, из которой будут загружены тестовые изображения для выполнения процедуры обучения инициализированного экземпляра нейронной сети.

Кнопка 28 позволяет указать путь к папке, из которой будут загружены изображения для выполнения классификации с помощью обученного экземпляра программной реализации нейронной сети

Кнопка 29 позволяет указать путь к текстовому файлу, в который будут сохранены параметры обученной нейронной сети, либо из которого параметры нейронной сети будут загружены.

Кнопка 30 вызывает процедуру сохранения результатов классификации дорожных знаков на изображениях из выбранной папки в папку, расположенную по пути, указанном в поле 11. Поле 11 должно быть заполнено.

Кнопка 31 вызывает процедуру загрузки результатов классификации дорожных знаков на изображениях из выбранной папки в удаленную базу данных.

Кнопка 32 позволяет указать путь к текстовому файлу, в который будут сохранены результаты работы программы.

6.4.3 Описание элементов отображения состояния

Текст элемента 33 отображает количество обнаруженных видеозаписей формата mp4 в папке, расположенной по пути указанном в поле 1.

Текст элемента 34 отображает количество текстовых файлов, содержащих каскады Хаара в папке, расположенной по пути указанном в поле 3.

Текст элемента 35 отображает количество папок, содержащих изображения, полученные после преобразования видеозаписей, по пути

указанном в поле 4.

Текст элемента 36 отображает количество изображений, полученных после преобразования видеозаписей, по пути указанном в поле 4.

Текст элемента 37 отображает количество слоев инициализированного экземпляра программной реализации сверточной нейронной сети.

Текст элемента 38 отображает количество распознаваемых классов инициализированного экземпляра программной реализации сверточной нейронной сети.

Текст элемента 39 отображает достигнутую точность классификации инициализированного и обученного экземпляра программной реализации сверточной нейронной сети.

Текст элемента 40 отображает состояние обученности экземпляра программной реализации сверточной нейронной сети.

Текст элемента 41 отображает количество обнаруженных обучающих изображений для настройки весовых коэффициентов экземпляра программной реализации сверточной нейронной сети в папке, расположенной по пути указанном в поле 7.

Текст элемента 42 отображает количество обнаруженных тестовых изображений для настройки весовых коэффициентов экземпляра программной реализации сверточной нейронной сети в папке, расположенной по пути указанном в поле 8.

Текст элемента 43 отображает количество обнаруженных изображений для классификации с помощью экземпляра программной реализации сверточной нейронной сети в папке, расположенной по пути указанном в поле 9.

Текст элемента 44 отображает состояние текстового файла с параметрами нейронной сети по пути, указанном в поле 10.

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ ПРОГРАММНОГО СРЕДСТВА РАСПОЗНАВАНИЯ, АНАЛИЗА И УЧЕТА КОМПОНЕНТОВ ДОРОЖНОЙ ИНФРАСТРУКТУРЫ

7.1 Характеристика программного средства распознавания, анализа и учета компонентов дорожной инфраструктуры

Разрабатываемое средство представляет собой приложение для ОС семейства Windows. На основе базы видеозаписей в формате MP4, полученных с автомобильных видеорегистраторов, будет производится сбор информации о расположенных на проезжей части дорожных знаках и составление базы данных с информацией о видах знаков и их географическом расположении. Приложение позволяет проводить раскадровку видеозаписей, фильтрацию и изменение размеров полученных изображений. Так же имеется возможность сохранение полученных данных на жесткий диск и экспортирование этих данных в базу данных при необходимости.

Полных аналогов данного средства не было найдено. Имеются либо проекты в области детекции или классификации изображений, либо приложения на ОС Android, которые выполняют функцию видеорегистратора с возможностью предупреждения водителя о дорожных знаках впереди автомобиля. Они работают в реальном времени и не имеют возможности экспортирования найденных дорожных знаков для дальнейшего использования.

Разрабатываемое программное средство предназначено для использования в навигационных и логистических системах в крупных компаниях, которым необходимо составлять маршруты с учетом скоростного режима и других важных аспектов дорожной инфраструктуры. Данное средство разрабатывается для нужд государственной компании и будет применятся для пополнения существующего проекта создания фотокарт дорожными знаками.

К преимуществам данного программного средства можно отнести:

– высокий процентный показатель распознавания знаков;

– возможность автоматической обработки;

– понятный интерфейс;

– сохранение путей к файлам и папкам;

– сохранение результатов в локальное хранилище и возможность их последующей записи в базу данных;

– способность подстраиваться под количество классов распознаваемых знаков, что дает возможность самостоятельно обновлять базу дорожных знаков при изменении нормативных документов.

7.2 Расчет затрат на разработку программного средства

Основная заработная плата исполнителей ($З_о$) рассчитывается по формуле (7.1)

$$З_о = \sum_{i=1}^{n} Т_{чi} * ТР_{чi} * К_i, \quad (7.1)$$

где n – количество исполнителей, занятых сотрудников;

$Т_{чi}$ – Часовая заработная плата *i*-го исполнителя, руб./ч;

$ТР_{чi}$ – трудоемкость работ *i*-го исполнителя, ч;

$К_i$ – коэффициент премирования.

Результаты расчета основной заработной платы исполнителей представлены в таблице 7.1. За норму рабочего времени на 2019 год примем 165 часов в месяц, коэффициент премирования возьмем равным 1,24.

Таблица 7.1 – Расчет основной заработной платы исполнителей

Исполнитель	Месячная заработная плата($Т_м$), руб.	Часовая заработная плата($Т_ч$), руб.	Трудоемкость работ (ТР), ч.	Коэффициент премий(К)	Заработная плата (З), руб.
1 Ведущий-разработчик	2000,00	12,12	130	1,24	1953,94

2 Инженер-программист	1000,00	6,06	280	1,24	2104,24
Основная заработная плата($З_о$)					4058,18

Величину дополнительной заработной платы исполнителей вычислим по формуле (7.2):

$$З_д = \frac{З_о * Н_д}{100}, \quad (7.2)$$

где $Н_д$ – норматив дополнительной заработной платы (15%).

Дополнительная заработная плата составит:

$$З_д = 4058{,}18 \cdot 20/100 = 811{,}64 \text{ руб.}$$

Отчисления в фонд социальной защиты населения и на обязательное страхование ($З_{сз}$) определяются по в соответствии с действующими законодательными актами по формуле (7.3):

$$З_{сз} = \frac{(З_о + З_д) * Н_{сз}}{100}, \quad (7.3)$$

где $Н_{сз}$ – норматив отчислений в фонд социальной защиты населения(34%) и на обязательное страхование (0,6%), суммарно 34,6%.

Размер отчислений в фонд социальной защиты населения и на обязательно страхование составит:

$$З_{сз} = (4058{,}18 + 811{,}64) * \frac{34{,}6}{100} = 1614{,}75 \text{ руб.}$$

Расходы по статье «Машинное время» ($Р_м$) определяем по формуле (7.4):

$$Р_м = Ц_м * ТР_о, \quad (7.4)$$

где $Ц_м$ – цена одного машинного времени, м-ч, 2,5 руб.;

$ТР_о$ – общая трудоемкость работы, ч.

Расходы на использование машинного времени составят:

$$Р_м = 2{,}5 \cdot (130 + 280) = 1025{,}00 \text{ руб.}$$

Полная сумма затрат на разработку программного средства ($З_р$) получим путем сложения всех рассчитанных статей затрат:

$$З_р = 4058{,}18 + 608{,}72 + 1614{,}75 + 820 = 7579{,}78 \text{ руб.}$$

7.3 Расчет экономической эффективности реализации на рынке программного средства распознавания, анализа и учета компонентов дорожной инфраструктуры

Разрабатываемое программное средство планируется распространять через сеть Интернет путем продажи заинтересованным организациям лицензий на пользование продуктом сроком на 1 год по цене 650 рублей. Предполагается, что в среднем в год лицензии на пользование продуктом будут приобретать 10 организаций. Таким образом, чистая прибыль, полученная от реализации программного средства на рынке ($П_ч$) будет рассчитываться по формуле (7.5):

$$П_ч = (Ц * N - НДС) * (1 - Н_п), \quad (7.5)$$

где Ц – цена одной лицензии, руб.;
N – ожидаемое количество приобретенных лицензий;
НДС – налог на добавленную стоимость;
$Н_п$ – ставка налога на прибыль.
Сумму налога на добавленную стоимость рассчитаем по формуле (7.6):

$$НДС = \frac{Ц*N*Н_{дс}}{(100\%+Н_{дс})}, \quad (7.6)$$

где Ндс – ставка налога на добавленную стоимость согласно действующему законодательству, (20%).

Таким образом, величина налога на добавленную стоимость составит:

$$НДС = (650 \cdot 10 \cdot 20/(100+20)) = 1083{,}33 \text{ руб.}$$

Прибыль составит:

$$П_ч = (650 * 10 - 3000) \cdot (1 - 0{,}18) = 4441{,}67 \text{ руб.}$$

7.4 Расчет показателей эффективности инвестиций в разработку программного средства распознавания, анализа и учета компонентов дорожной инфраструктуры

Сравнивая величину годового экономического эффекта в виде прогнозируемой прибыли ($П_ч$) с величиной инвестиций (полной суммы затрат на разработку ($З_р$)), можно сделать вывод, что инвестиции не окупятся за один год. Поэтому, для расчета эффективности инвестиций необходимо выполнить расчеты чистого дисконтированного дохода (ЧДД), срока окупаемости ($Т_{ок}$) и рентабельности инвестиций ($Р_и$). Чистый дисконтированный доход (ЧДД) рассчитывается по формуле (7.7):

$$ЧДД = \sum_{t=1}^{n}(P_t * \alpha_t - З_t * \alpha_t), \quad (7.7)$$

где n – расчетный период, лет;
P_t – результат (экономический эффект), полученный в году t, руб.;
$З_t$ – затраты (инвестиции) в году t, руб.;

α_t – коэффициент дисконтирования, определяемый по формуле (7.8):

$$\alpha_t = \frac{1}{(1+E_n)^t}, \quad (7.8)$$

где E_n – норма дисконта, на расчетный 2019 год равная 0,15;

t – порядковый номер года в расчетном периоде (шаг расчета). Срок окупаемости проекта – момент, когда суммарный дисконтированный результат (эффект) станет равным или превысит дисконтированную сумму инвестиций. То есть, определяется через какой период времени инвестиционный проект начнет приносить инвестору прибыль. Рентабельность инвестиций ($Р_и$) рассчитывается как отношение суммы дисконтированных результатов (эффектов) к осуществленным инвестициям (7.9):

$$Р_и = \frac{\sum_{t=0}^{n} P_t * \alpha_t}{\sum_{t=0}^{n} З_t * \alpha_t}, \quad (7.9)$$

Расчет показателей эффективности представлен в таблице 7.2. За нулевой шаг расчета был принят 2019 год. Разрабатываемое программное средство выйдет на рынок в сентябре 2019 года, поэтому ожидаемый прирост чистой прибыли в 2019 году составит 25% от годового.

Таблица 7.2 – Оценка экономической эффективности инвестиций

Показатель	Шаги расчета			
1	2			
	0	1	2	3
Результат				
1 Прирост чистой прибыли, руб.	1110,42	4441,67	4441,67	4441,67
Затраты (инвестиции)				
2 Инвестиции в разработку, руб.	-7579,78	0,00	0,00	0,00
Экономический эффект				
4 Чистый поток наличности (ЧПН), руб.	-6469,36	4441,67	4441,67	4441,67
5 То же самое нарастающим итогом, руб.	-6469,36	-2027,69	2413,97	6 855,64
6 Коэффициент дисконтирования	1,00	0,87	0,76	0,66
7 Дисконтированный ЧПН, руб.	-6469,36	3862,32	3358,54	2920,47
8 Чистый дисконтированный доход, руб.	3671,97			

9 Внутренняя норма доходности	33,73%
10 Индекс рентабельности	1,48
11 Срок окупаемости	1 год 6 месяцев

Проведя анализ таблицы 7.2 можно сделать вывод о том, что разработка программного средства является экономически целесообразной и выгодной. ЧДД является величиной положительной и равен 3671,97 руб., ВНД превышает ставку дисконта, а индекс рентабельности больше 1,0. Инвестиции в разработку окупятся спустя 1 год и 6 месяцев после выхода продукта на рынок.