ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Государственный университет морского и речного флота имени адмирала С.О. Макарова»
(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)
Воронежский филиал

Кафедра математики, информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
(код, наименование направления подготовки/специальности)
Форма обучения очная

«К ЗАЩИТЕ ДОПУЩЕН(А)»
Заведующий кафедрой
_____
(подпись)
Черняева С. Н.
(ФИО)
_____ 20___

Выпускная квалификационная работа

Обучающегося Мосина Евгения Владиславовича
(фамилия, имя, отчество)
Вид работы выпускная квалификационная работа бакалавра
(выпускная квалификационная работа бакалавра, специалиста, магистра)

Пояснительная записка

Тема Разработка приложения контроля параметров крупногабаритных и тяжеловесных транспортных средств (на примере АО "Тандер")
(полное название темы квалификационной работы, в соответствии с приказом об утверждении тематики ВКР)

Руководитель работы к.т.н., доцент Матыцина И. А.
(должность, подпись, фамилия, инициалы, дата)
Консультант _____
(при наличии) (должность, подпись, фамилия, инициалы, дата)
Консультант _____
(должность, подпись, фамилия, инициалы, дата)
Обучающийся Мосин Е. В.
(подпись, фамилия, инициалы, дата)

Воронеж2024

ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Государственный университет морского и речного флота имени адмирала С.О. Макарова»
(ФГБОУ ВО «ГУМРФ имени адмирала С.О. Макарова»)
Воронежский филиал

Кафедра математики, информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
(код, наименование направления подготовки/специальности)
Форма обучения очная

УТВЕРЖДАЮ
Заведующий кафедрой
_____
(подпись)
Черняева С. Н.
(ФИО)
_____ 2024

Задание
на выпускную квалификационную работу

Вид работы ВКР бакалавра
(ВКР бакалавра, ВКР специалиста, ВКР магистра)

Обучающемуся Мосину Евгению Владиславовичу
(фамилия, имя, отчество)

Тема Разработка приложения контроля параметров крупногабаритных и тяжеловесных транспортных средств (на примере АО "Тандер")

Утверждена приказом ректора университета от _____ 20___, № _____

Срок сдачи законченной работы _____ 20___

Исходные данные (или цель ВКР):
Разработать приложение контроля параметров крупногабаритных и тяжеловесных транспортных средств. Приложение, позволяющее на основе камер видеонаблюдения отслеживать состояние транспортных средств, их габариты, контролировать нагрузку на ось автомобиля. Разработать алгоритмы компьютерного зрения, для автоматического анализа изображений, определяя длину, ширину и вес крупногабаритного транспорта.

Перечень подлежащих исследованию, разработке, проектированию вопросов (краткое содержание ВКР):
(актуальность темы, цели и задачи ВКР; аналитический обзор литературных источников; постановка задачи исследования, разработки, проектирования; содержание процедуры исследования, разработки, проектирования; обсуждение результатов; дополнительные вопросы, подлежащие разработке; заключение – выводы по работе в целом, оценка степени решения поставленных задач, практические рекомендации; и др.)

– Введение. Актуальность выбранной темы, цель и задачи ВКР

(наименование вопроса, раздела и его краткое содержание)

– Исследовательский раздел. Современные методы контроля параметров транспортных средств. Обзор нормативных документов и стандартов. Примеры реализации систем контроля на практике. Теоретические основы автоматизированных систем контроля параметров. Основные принципы автоматизированных систем контроля. Процесс обучения модели машинного обучения.

(наименование вопроса, раздела и его краткое содержание)

– Проектный раздел. Разработка автоматизированной подсистемы контроля параметров. Постановка задачи. Архитектура системы. Диаграмма классов. Основные компоненты. Этапы разработки. Внедрение и оценка. Практическое внедрение и тестирование. Используемые классы и методы. Тест и анализ.

(наименование вопроса, раздела и его краткое содержание)

– Заключение. Выводы по работе в целом. Оценка степени решения поставленных задач

(наименование вопроса, раздела и его краткое содержание)

Практические рекомендации

Перечень графического материала (или презентационного материала):

1. Титульный лист
2. Цель и задачи ВКР
3. Описание предметной области
4. Проектирование системы
5. Проектирование системы (Продолжение)
6. Проектирование системы (Продолжение)
7. Разработка системы
8. Разработка системы (Продолжение)
9. Разработка системы (Продолжение)
10. Результаты ВКР

Консультанты по разделам ВКР (при наличии):

1. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

2. _____

(наименование раздела, ученая степень, ученое звание и должность, ФИО консультанта)

Дата выдачи задания: _____ 20___

Задание согласовано и принято к исполнению: _____ 20___

Руководитель ВКР: к.т.н., доцент Матыцина Ирина Александровна _____

(должность, ученая степень, ученое звание, ФИО) (подпись)

Обучающийся: ИТ-4 Мосин Евгений Владиславович _____

(учебная группа, ФИО) (подпись)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последние десятилетия наблюдается значительный рост объемов грузоперевозок [1], что связано с ростом мировой экономики и развитием международной торговли. В связи с этим контроль параметров крупногабаритных и тяжелых транспортных средств (КГТТС) становится все более актуальной задачей. Одним из важных аспектов этой отрасли является контроль веса грузовиков для обеспечения безопасности на дорогах и соблюдения нормативных требований. Традиционные методы взвешивания грузовиков, такие как весовые станции, требуют значительных затрат времени и ресурсов. Это приводит к необходимости разработки более эффективных и автоматизированных решений для оценки веса грузовиков.

Нарушение нормативных требований к массогабаритным параметрам таких транспортных средств может привести к значительному ущербу дорожной инфраструктуре, увеличению аварийности и увеличению эксплуатационных расходов.

Цель данной работы разработка программного обеспечения, позволяющего оценивать вес и габариты грузовиков на основе анализа их изображений. Программа должна быть способна обрабатывать изображения, выделять ключевые элементы грузовика и на основе их анализа вычислять общий вес транспортного средства. Для достижения поставленных целей необходимо решить следующие задачи:

1. Провести анализ современных методов и систем контроля параметров транспортных средств.
2. Изучить нормативные документы и стандарты, регламентирующие параметры КГТТС.
3. Разработать архитектуру автоматизированной подсистемы контроля параметров.
4. Выбрать и обосновать использование конкретных технических и программных решений.

5. Реализовать прототип подсистемы и провести его тестирование.

Объектом исследования данной работы являются крупногабаритные и тяжеловесные грузовые автомобили в сфере грузоперевозок. Предметом исследования являются методы и средства автоматизированного контроля их параметров.

Методы исследования включают анализ и синтез научно-технической литературы, моделирование процессов, проектирование и тестирование программы. В качестве инструментов разработки автоматизированных систем используются современные технологии и программные средства.

Теоретические методы включают анализ существующих решений и моделей, синтез новых подходов, разработку алгоритмов и схем. Эмпирические методы включают проведение экспериментов, тестирование разработанного программного обеспечения и анализ полученных данных.

Практическая значимость работы заключается в том, что разработанные подсистемы могут быть реализованы в реальных условиях работы, что позволит существенно повысить точность и эффективность контроля параметров КГТТС. Это, в свою очередь, способствует снижению затрат на ремонт дорог, уменьшению количества аварий с участием крупногабаритных и тяжеловесных транспортных средств и обеспечению общей безопасности дорожного движения [2].

В структуру работы входит введение, которое демонстрирует актуальность темы, определяет цели и задачи исследования, описывает объект и тему исследования, поясняет методы исследования и практическую значимость работы. Основная часть работы обзор литературы и существующих решений, теоретические основы разработки автоматизированных систем управления, процесс разработки и внедрения подсистем, результаты испытаний и экономическое обоснование. Работа завершается заключением, содержащим основные выводы и перспективы дальнейших исследований.

В последние десятилетия наблюдается значительный рост объемов грузоперевозок и это обусловлено увеличением масштабов мировой экономики и развитием международной торговли.

В связи с этим, автоматизация процесса взвешивания грузовиков позволит значительно сократить время на проведение процедуры, снизить затраты на эксплуатацию весовых станций и улучшить безопасность на дорогах за счет более частого и точного контроля за перегруженными транспортными средствами.

Однако их использование сопряжено с рядом проблем, таких как повышенное воздействие на дорожную инфраструктуру, увеличение вероятности аварийных ситуаций и рост эксплуатационных затрат.

КГТТС является одной из ключевых задач для обеспечения безопасности и сохранности дорожных покрытий. Существующие методы контроля часто недостаточно эффективны и не позволяют оперативно реагировать на нарушения нормативных требований [3]. В этом контексте разработка автоматизированной подсистемы контроля параметров представляет собой актуальную и востребованную задачу.

Разработанная автоматизированная подсистема мониторинга параметров КГТТС может использоваться как государственными, так и частными организациями, занимающимися контролем и регулированием грузовых перевозок. Внедрение этих систем оптимизирует процессы мониторинга и контроля, снижает человеческий фактор и обеспечивает более жесткий контроль за соблюдением нормативных требований.

1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1 Современные методы контроля параметров транспортных средств

Контроль параметров крупногабаритных и тяжеловесных транспортных средств (КГТТС) представляет собой сложную и многогранную задачу, требующую применения различных методов и технологий. В настоящее время существуют несколько подходов к решению данной проблемы, каждый из которых обладает своими преимуществами и недостатками.

Наиболее распространенным является использование весоизмерительных систем, устанавливаемых на дорогах и контрольно-пропускных пунктах. Они позволяют точно измерять вес транспортных средств и сравнивать его с нормативными значениями [4]. Современные весоизмерительные системы оснащены высокочувствительными датчиками и позволяют проводить измерения в режиме реального времени. Однако, они требуют значительных затрат на установку и обслуживание, а также могут быть неэффективны в условиях интенсивного движения.

Другим важным методом КГТТС является использование габаритных рамок и лазерных сканеров. Габаритные рамки позволяют измерять размеры транспортных средств и выявлять случаи превышения допустимых габаритов. Лазерные сканеры, в свою очередь, обеспечивают высокую точность измерений и позволяют создавать трехмерные модели транспортных средств. Эти методы особенно эффективны при контроле за транспортными средствами на границах и въездах в города, где необходимо оперативно выявлять нарушения.

Также стоит отметить использование систем видеонаблюдения и распознавания номеров, которые позволяют автоматически фиксировать нарушения и идентифицировать транспортные средства [5]. Эти системы широко применяются в странах с высоким уровнем автоматизации

дорожного контроля и позволяют значительно повысить эффективность работы контролирующих органов.

1.2 Обзор нормативных документов и стандартов

Контроль параметров КГТТС регламентируется рядом нормативных документов и стандартов, как на национальном, так и на международном уровнях. Важнейшими документами, определяющими требования к параметрам транспортных средств, являются национальные дорожные кодексы и правила дорожного движения.

Особое значение на международном уровне имеют Конвенция о дорожном движении, подписанная в Вене в 1968 году, и Европейское соглашение о международной дорожной перевозке опасных грузов (ADR) [6]. Эти документы устанавливают общие принципы и требования к транспортным средствам, участвующим в международных перевозках, и обеспечивают гармонизацию нормативных актов различных стран (рис.1.1).

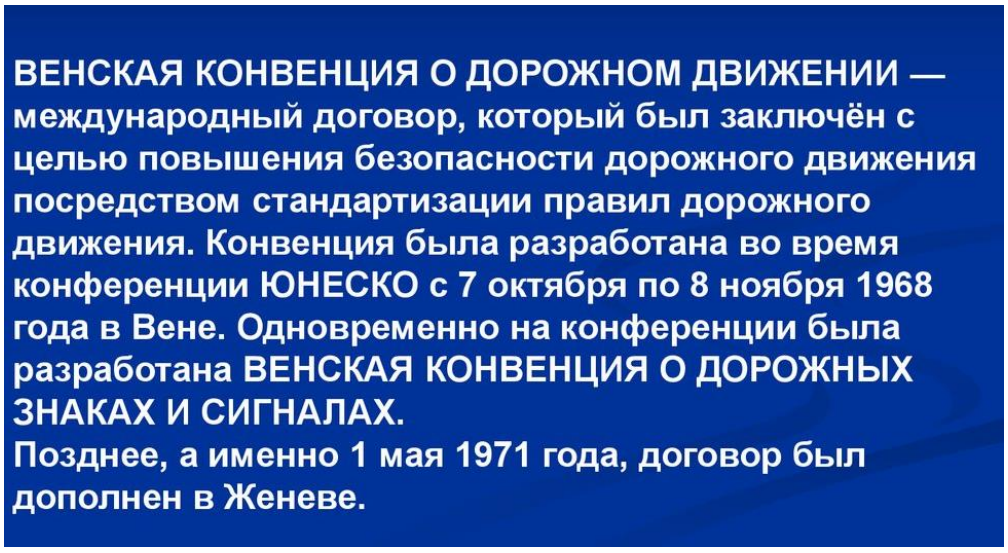

Рисунок 1.1 - Конвенция о дорожном движении.

Кроме того, в каждой стране существуют специальные стандарты и нормативы, регламентирующие параметры КГТТС. Например, в Российской Федерации такими документами являются ГОСТы и СНИПы, которые

устанавливают требования к весовым и габаритным характеристикам транспортных средств, а также правила их эксплуатации и контроля (рис. 1.2).

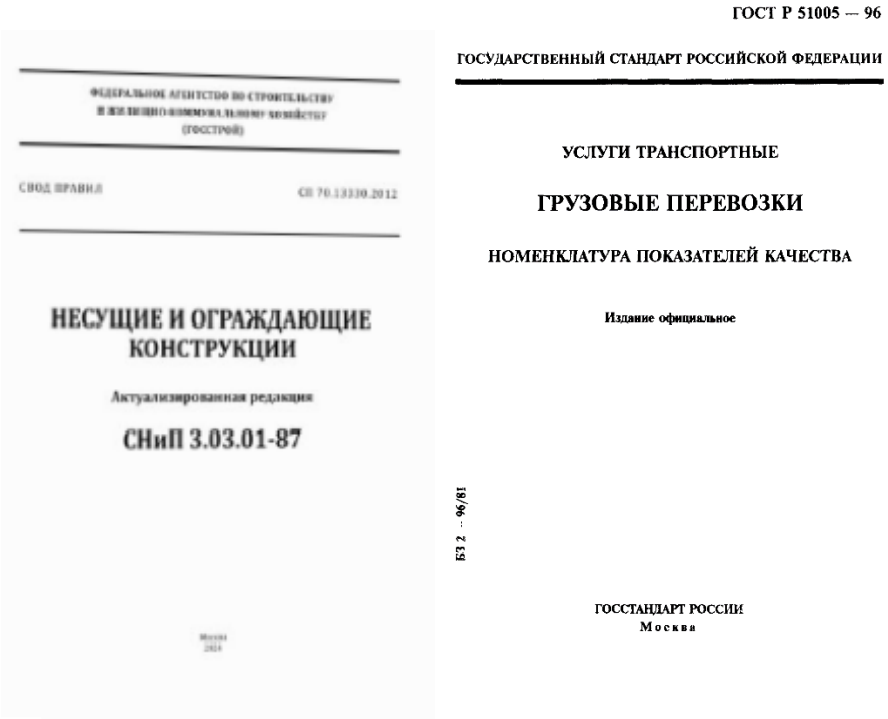

Рисунок 1.2 - ГОСТы и СНИПы.

1.3 Примеры реализации систем контроля на практике

В последние годы во многих странах мира активно внедряются автоматизированные системы контроля параметров КГТТС. Одним из примеров успешной реализации таких систем является проект "Интеллектуальная транспортная система" в Японии. В рамках этого проекта были установлены высокоточные весоизмерительные системы и габаритные рамки, а также разработано программное обеспечение для анализа данных и выявления нарушений. Система позволила значительно снизить количество нарушений и повысить безопасность дорожного движения.

Еще одним примером является проект «Weigh-in-Motion» (WiM) в США. В рамках данного проекта на автодорогах страны установлены весоизмерительные системы, которые позволяют проводить контроль веса

транспортных средств без необходимости их остановки. Система «WiM» оснащена датчиками, интегрированными в дорожное полотно, и позволяет измерять вес транспортных средств в движении с высокой точностью. Проект «WiM» показал высокую эффективность в снижении количества перегруженных транспортных средств и минимизации ущерба дорожной инфраструктуре.

В Европе также реализуются подобные проекты. В Германии, например, система контроля «EuroControl Route» позволяет осуществлять мониторинг параметров с использованием различных датчиков и камер. Система интегрирована с базами данных транспортных средств и позволяет автоматически выявлять нарушения и отправлять уведомления владельцам ТС. Внедрение данной системы способствовало высокому повышению уровня соблюдения нормативных требований и улучшению состояния дорожной инфраструктуры.

Следует отметить, что контроль параметров крупногабаритных и тяжеловесных транспортных средств является важной и актуальной задачей, требующей применения современных технологий и подходов. Анализ существующих методов и систем показывает, что наиболее эффективными являются автоматизированные системы, позволяющие проводить контроль в режиме реального времени и минимизировать влияние человеческого фактора.

Однако, несмотря на значительные достижения в данной области, остаются нерешенные проблемы и вызовы, связанные с интеграцией различных систем и стандартизацией данных. Внедрение комплексных автоматизированных подсистем требует значительных затрат и координации усилий на национальном и международном уровнях. Тем не менее, развитие и внедрение таких систем является важным шагом на пути к повышению безопасности и эффективности грузоперевозок, а также сохранению дорожной инфраструктуры.

1.4 Теоретические основы автоматизированных систем контроля параметров

1.4.1 Основные принципы автоматизированных систем контроля

Применение автоматического управления включает, помимо прочего, самолеты, роботов, гражданские инженерные сооружения, управление технологическими процессами и т. д.

Автоматическое управление сыграло жизненно важную роль в развитии техники и науки.

Искусственный интеллект (ИИ) и машинное обучение (МО) играют решающую роль в современном технологическом прогрессе. Это не просто нити, а сами ткачи, которые переопределяют ткань систем управления в различных областях. Эта трансформация заметна в производственном секторе, где они используются на производственных линиях, а также в работе автономных транспортных средств, сетях распределения энергии и планах индивидуального ухода в здравоохранении. В этой статье мы более подробно рассмотрим, как искусственный интеллект и машинное обучение меняют системы управления, расширяя инновации, основанные на технической глубине и аналитической перспективе.

В традиционной теории управления системы проектируются так, чтобы реагировать ожидаемым образом в заранее определенных условиях. Однако реальный мир совсем не предсказуем. Именно здесь на помощь приходят адаптивные системы управления на базе искусственного интеллекта и машинного обучения. Эти системы хорошо подходят для сред, характеризующихся неопределенностью и изменениями. Они постоянно учатся и корректируют свое поведение на основе обратной связи в реальном времени.

Яркий пример таких систем можно увидеть в производственном секторе, где адаптивные системы управления отслеживают и регулируют параметры производственного процесса на ходу. Эти системы используют

алгоритмы машинного обучения для прогнозирования износа оборудования, тем самым оптимизируя графики технического обслуживания для предотвращения сбоев и дорогостоящих простоев. Алгоритмы могут анализировать терабайты операционных данных, выявляя закономерности, которые предшествуют сбоям оборудования, — задача, которую люди-операторы не могут выполнить.

Прогнозируемое обслуживание

Прогнозное обслуживание является примером того, как системы на базе искусственного интеллекта преобразуют реактивные процессы в проактивные стратегии. Используя исторические данные и аналитику в реальном времени, модели искусственного интеллекта могут прогнозировать неисправности оборудования до их возникновения, планируя техническое обслуживание только при необходимости. Такой подход не только продлевает срок службы оборудования, но и оптимизирует ресурсы для технического обслуживания, что резко контрастирует с традиционными методами технического обслуживания по графику.

Принятие решений и оптимизация процессов

В области оптимизации процессов алгоритмы искусственного интеллекта (ИИ) и машинного обучения (ML) используются для анализа сложных наборов данных. Эти алгоритмы помогают определить наилучшие условия эксплуатации путем балансирования таких факторов, как эффективность, стоимость и энергопотребление. Например, в энергетических системах алгоритмы искусственного интеллекта управляют и оптимизируют потоки возобновляемой энергии на основе прогнозных моделей спроса и предложения. Это не только повышает эффективность, но и обеспечивает более плавную интеграцию возобновляемых ресурсов в энергетическую сеть, что способствует устойчивости.

Автономные системы и робототехника

Влияние искусственного интеллекта (ИИ) и машинного обучения (МО) лучше всего наблюдается в области автономных систем и робототехники. В

этой области модели машинного обучения, особенно основанные на глубоком обучении, позволяют роботам и беспилотным транспортным средствам понимать окружающую среду, принимать решения и учиться на своем прошлом опыте. Эти системы собирают данные с ряда датчиков, анализируют и реагируют на окружающую среду с уровнем сложности, имитирующим человеческую интуицию, и в то же время со скоростью и согласованностью, которые могут обеспечить только машины.

Навигация по вызовам

Внедрение искусственного интеллекта и машинного обучения в системы управления многообещающе, но также сопряжено с рядом проблем. Основными проблемами являются конфиденциальность данных, безопасность и этические последствия автоматического принятия решений. Непрозрачность некоторых алгоритмов МО, особенно моделей глубокого обучения, поднимает вопросы об интерпретируемости и подотчетности решений, принимаемых системами ИИ.

Более того, эффективность моделей искусственного интеллекта и машинного обучения зависит от качества и актуальности данных, на которых они обучаются. Чтобы обеспечить целостность и актуальность данных, предлагается внедрить надежные стратегии управления и управления данными, которые позволят постоянно решать проблемы поддержания качества данных.

Автоматизация управления стала возможна благодаря наличию современных технических средств, математического и организационного обеспечения, а также благодаря экономичности автоматизации управленческих процессов и гибкости производственной информации. Это позволяет выделить набор принципов, определяющих потенциал АСУ.

Основой организации производственного процесса на каждом предприятии и в каждом цехе является рациональное сочетание в пространстве и времени всех основных, вспомогательных и обслуживающих процессов.

Особенности и методы такого сочетания зависят от условий производства, но существуют общие принципы. Они вытекают из экономических законов и, в частности, из идеи пропорционального развития народного хозяйства.

Принцип специализации в системах автоматического управления. Специализация – это форма общественного разделения труда, планомерно развивающаяся в народном хозяйстве и определяющая распределение и обособление отраслей, предприятий, цехов, отделов, участков, линий и т. д., выпускающих определенную продукцию и выполняющих определенные процессы. Уровень специализации предприятия или подотрасли определяется сочетанием двух основных факторов: объемом производства и трудоемкостью продукции. На специализацию сильно влияют стандартизация и нормирование, которые могут увеличить масштабы производства однородной продукции. В целом специализация характеризуется большой экономической эффективностью.

Соблюдение принципа специализации заключается в распределении ограниченного числа рабочих терминов, минимально возможного количества различных задач, на каждом производственном участке вплоть до цеха.

Принцип пропорциональности в автоматизированных системах управления. Все производственные единицы в основных и вспомогательных цехах, отделениях, линиях, группах оборудования и мастерских сервисного хозяйства должны иметь пропорциональную производительность в единицу времени.

Пропорциональность производственной мощности обеспечивает полную загрузку оборудования и площадей и равномерный выпуск готовой продукции.

Если принцип пропорциональности не соблюдается, возникают узкие места и диспропорции, количество продукции и услуг в отдельных подразделениях будет недостаточным для выполнения производственных задач, что будет препятствовать дальнейшему развитию производства.

Принцип параллелизма в автоматических системах управления. Одновременное параллельное выполнение отдельных участков, этапов, фаз и операций производственного процесса расширяет фронт работ и значительно сокращает длительность производственного цикла. Параллельность проявляется в различных формах, таких как структура технических операций, сочетание основных и вспомогательных операций, одновременное выполнение нескольких технических операций.

Принцип прямоточности в автоматизированных системах управления. Продукция, выпускаемая предприятием, должна проходить все стадии и операции производственного процесса, т.е. от исходного материального начала до выхода готовой продукции, по кратчайшему пути без встречных и возвратных движений.

Соблюдение этого принципа обеспечивается при организации зданий, сооружений, цехов и машин, а также при построении технических процессов.

Вспомогательные подразделения и склады располагаются как можно ближе к сервисным и основным цехам.

Принцип непрерывности в автоматизированных системах управления. Перерывы в производстве должны быть исключены или сокращены. Это относится ко всем перерывам, в том числе внутри операций, между операциями, внутри смены и между сменами. Машина или система машин тем совершеннее, чем выше непрерывность рабочих процессов. Организация производственного процесса тем совершеннее, чем выше степень непрерывности, достигнутая в нем.

Принцип ритмичности в системах автоматического управления. Производственный процесс должен быть организован таким образом, чтобы равные или возрастающие количества продукции производились через равные промежутки времени, а все предложения и операции процесса повторялись через эти промежутки времени.

Различают начальный ритм процесса, ритм рабочих интервалов и ритм выпуска. Начальный ритм является конечным ритмом.

В многопродуктовом, единичном производстве ритмичность автоматизированного процесса достигается выпуском одинаковых или систематически возрастающих количеств продукции за равные календарные периоды или длительности.

Требование ритмичности предъявляется не только к основному производству, но и к подразделениям, отвечающим за подготовку и обслуживание производства.

Установка автоматизированной системы управления технологическим процессом должна быть направлена на соблюдение принципов организации производственного процесса, а функционирование АСУ должно обеспечивать соблюдение принципов непрерывности и ритмичности.

Автоматизация

Понятие «автоматизация» предполагает, что машинам, приборам и станкам помимо собственно производственной функции передаются функции управления и контроля, которые до этого выполнялись человеком. Современное развитие технологий позволяет автоматизировать не только физический, но и интеллектуальный труд, если он основан на формальных процессах.

За последние 7 десятилетий автоматизация предприятий прошла долгий путь, который умещается в 3 этапа:

системы автоматического контроля (САК) и системы автоматического регулирования (САР)

системы автоматизации технологических процессов (САУ)

автоматизированные системы управления технологическими процессами (АСУ ТП)

На современном уровне автоматизация систем управления производством представляет собой многоуровневую схему взаимодействия людей и машин на основе систем автоматического сбора данных и сложных вычислительных комплексов, которые неустанно совершенствуются.

В нынешних экономических условиях на передовых позициях оказываются промышленные предприятия, которые гибко реагируют на изменяющиеся условия, могут выпускать разнообразную номенклатуру, быстро наладить выпуск продукции по новым стандартам, точно исполняют сроки и объемы заказов, при этом предлагая конкурентную цену и сохраняя качество на высоком уровне. Без современных средств и систем автоматизации производства соответствовать данным требованиям практически невозможно.

Основные цели и преимущества автоматизации предприятия в современных условиях:

уменьшение числа рабочих и обслуживающего персонала, в особенности на непрестижных, «грязных», «горячих», вредных, физически трудных участках производства

улучшение качества продукции;

увеличение производительности (рост объема продукции);

создание ритмичного производства с возможностью точного планирования;

повышение эффективности производства, в том числе более рациональное использование сырья, снижение потерь, повышение скорости выпуска продукции, повышение энергоэффективности,

улучшение показателей экологичности и безопасности производства, в том числе снижение вредных выбросов в атмосферу, снижение уровня травматизма и т.п.

повышение качества управления на предприятии, согласованная работа всех уровней системы производства.

Таким образом, затраты на автоматизацию производства и предприятия непременно окупаются при условии наличия спроса на выпускаемую продукцию.

Для достижения данных целей необходимо решить следующие задачи по автоматизации производственных процессов:

внедрение современных средств автоматизации (оборудования, программ, систем управления и контроля и т.п.) (рис. 1.3);

внедрение современных методов автоматизации (принципов построения систем автоматизации).

В результате повышается качество регулирования, удобство труда оператора, коэффициент готовности оборудования. Кроме этого, упрощается получение, обработка и хранение информации о производственных процессах и работе оборудования, а также контроль качества.

Рисунок 1.3 – Автоматизация предприятий

Автоматизированные системы контроля параметров крупногабаритных и тяжеловесных транспортных средств (КГТТС) представляют собой комплекс программных и аппаратных средств, предназначенных для автоматического сбора, обработки и анализа данных о параметрах транспортных средств. Основной целью таких систем является обеспечение точного и своевременного мониторинга параметров, что способствует

соблюдению нормативных требований и повышению безопасности дорожного движения.

Принцип работы автоматизированных систем контроля основан на использовании различных датчиков и измерительных устройств, которые фиксируют параметры транспортных средств (вес, габариты, скорость и др.) в реальном времени [7].

1.4.2 Процесс обучения модели машинного обучения

Процесс обучения подразумевает следующее:

1. Подготовка необходимых данных, включая загрузку, очистку, предварительную обработку и разбиение их на обучающие и тестовые наборы.

2. Выбор нужной модели, зависит от типа задачи и характеристик данных. Это может быть сверточная нейронная сеть для задачи классификации изображений.

3. Обучение модели. На этом этапе образец подается на вход обучающего набора реальных данных и в процесс, который настраивает ее параметры, чтобы минимизировать ошибку прогнозирования.

4. Оценка модели. После обучения «манекена», анализируется ее производительность в тестовом наборе, чтобы понять, насколько хорошо она реагирует на новые данные.

5. Настройка гиперпараметра. Настраиваем готовую модель, чтобы улучшить ее производительность и возможности обобщения.

Примеры алгоритмов машинного обучения для обработки изображений

Сверточная нейронная сеть (СНС): широко используемый алгоритм классификации и обнаружения объектов на изображении.

СНС является одним из типов нейронных сетей, разработанным для эффективного анализа преимущественно двумерных и трехмерных данных, таких как изображения (в том числе RGB-изображения). Архитектуру СНС

впервые предложил французский исследователь в области искусственного интеллекта и машинного обучения Ян Лекун в конце 1980-х годов. В 1989 году он, совместно с коллегами, впервые использовал СНС для распознавания рукописных цифр [9].

СНС обладает следующими полезными преимуществами:

– Высокая временная эффективность по сравнению с перцептроном благодаря меньшему количеству настраиваемых параметров.

– Улучшенные способности выделения отдельных элементов на изображении (углы, кривые, прямые, яркие области и т. д.) за счет использования нескольких карт признаков на одном слое.

– Способность формирования высокоуровневых признаков на основе низкоуровневых в пределах одного класса благодаря применению ядер свертки небольшого размера, вместо соединения нейронов двух соседних слоев по принципу «каждый с каждым», как в полносвязном перцептроне.

В настоящее время СНС широко используются для распознавания и классификации объектов на двумерных изображениях, анализа спектрограмм и других двумерных наборов данных.

Сверточный автокодер. Используется для уменьшения размерности, а также для задач восстановления и фильтрации сложных изображений.

Сети глубокой семантической сегментации. Применяются для точной сегментации изображений на уровне пикселей.

Конкурирующие сети производителя «Wan». Используется для создания нового изображения на основе указанного экземпляра.

Различные методы машинного обучения могут быть использованы для создания интеллектуальных систем, которые могут распознавать, анализировать и делать выводы на основе визуальных данных.

1.4.3 Алгоритмы и модели, используемые в автоматизированных системах контроля

Эффективность автоматизированной системы управления параметрами КГТТС сильно зависит от используемого алгоритма и модели обработки данных, а именно от:

Использования для анализа больших объемов данных и определения закономерностей [10]. Например, для автоматического распознавания типов транспортных средств.

Распределения ресурсов и принятия решений. Например, линейное планирование используется для составления маршрутов транспортных средств.

Статистические методы, такие как анализ временных рядов и кластеризация, используются для анализа данных и нахождения аномалий [11], благодаря чему можно выявить нетипичные ситуации.

Базы данных и системы управления данными используют реляционные и нереляционные базы для хранения и обработки больших объемов информации. Система управления базами данных (СУБД) обеспечивает быстрый доступ к данным, их цельность и безопасность.

Системы реального времени используются для обеспечения оперативного управления и принятия решений, и данные обрабатываются по мере их поступления.

1.4.4 Технические требования к автоматизированным системам контроля

Для обеспечения надежной и эффективной работы автоматизированных систем контроля параметров КГТТС необходимо соблюдать ряд технических требований:

Все измерительные устройства должны обеспечивать высокую точность измерений параметров транспортных средств [12].

Приложение должно быть надежным и устойчивым к отказам.

Система должна быть способна обрабатывать увеличивающиеся объемы данных и справляться с ростом числа транспортных средств.

Для защиты данных от несанкционированного доступа и атак необходимо применять современные методы шифрования и аутентификации.

Интероперабельность - система должна быть совместимой с другими системами и стандартами чтобы интегрировать её в существующую инфраструктуру и обеспечивать обмен данными с внешними системами.

Пользовательские интерфейсы должны быть понятными и удобными в использовании, обеспечивая быстрый доступ к необходимой информации и возможностям управления.

1.4.5 Примеры успешного применения автоматизированных систем контроля

Применение автоматизированных систем контроля параметров КГТТС демонстрирует высокую эффективность в различных странах мира. Например, в США широко используется система «Weigh-in-Motion» (WiM), которая позволяет контролировать вес транспортных средств в движении, не создавая помех для трафика [13]. В Германии система «EuroControl Route» интегрирована с базами данных и обеспечивает мониторинг параметров КГТТС на всей территории страны, что значительно повысило уровень соблюдения нормативных требований.

В России также ведется активное внедрение автоматизированных систем контроля. Примером служит проект "Платон", который сочетает в себе функции взимания платы за проезд по федеральным дорогам и контроля параметров транспортных средств [14].

1.4.6 «Emgu CV»

EMGU — это оболочка C# для OpenCV. Она отличается от других оболочек тем, что написана исключительно на C# и не использует небезопасный код. EMGU открывает для разработчиков C# библиотеку функций программирования OpenCV (библиотека компьютерного зрения с открытым исходным кодом), в основном предназначенную для компьютерного зрения в реальном времени. OpenCV изначально был разработан Intel и теперь поддерживается Willow Garage. Текущие версии для архитектур x86 и x64.

Поскольку EMGU — это оболочка для кода C++. Существует два типа библиотек динамической компоновки (DLL). Есть EMGU с синтаксисом EMGU, которые всегда ссылаются на имя, и OpenCV, которые различаются.

Основные требования

Как и в любой библиотеке C#, в проекте есть несколько важных DLL, на которые необходимо ссылаться:

Emgu.CV.dll

Emgu.CV.UI.dll

Emgu.Util.dll

Описание

Emgu CV — это кроссплатформенная .Net-оболочка библиотеки обработки изображений OpenCV. Разрешение вызова функций OpenCV из .NET-совместимых языков, таких как C#, VB, VC++, IronPython и т. д. Оболочку можно скомпилировать в Mono и запустить на устройствах Windows, Linux, Mac OS X, iPhone, iPad и Android.

Преимущество Emgu CV

1. Кросс-платформенный

Emgu CV полностью написан на C#. Преимущество состоит в том, что его можно скомпилировать в Mono и, следовательно, он может работать на

любой платформе, которую поддерживает Mono, включая Linux, Mac OS X, iOS и Android.

2. Cross Language и поставляется с примером кода.

Emgu CV можно использовать на нескольких разных языках, включая C#, VB.NET, C++ и IronPython.

3. Класс изображения с общим цветом и глубиной.

4. Автоматический сбор мусора.

5. Сериализуемое изображение XML

6. XML-документация и поддержка IntelliSense.

7. Возможность использования класса Image или прямого вызова функций из OpenCV.

Можно создать изображение, вызвав CvInvoke.cvCreateImage, но вместо этого предлагается создать объект Image <TColor, TDepth>. Использование управляемого класса Image <TColor, TDepth> имеет несколько преимуществ.

8. ▪ Память автоматически освобождается сборщиком мусора.

9. ▪ Класс Image <TColor, TDepth> можно проверить с помощью визуализатора отладчика.

10. ▪ Класс Image <TColor, TDepth> содержит расширенный метод, недоступный в OpenCV, например, общие операции с пикселями изображения, преобразование в растровое изображение и т. д.

Обзор архитектуры

Emgu CV имеет два слоя оболочки, как показано на рисунке 1.4.

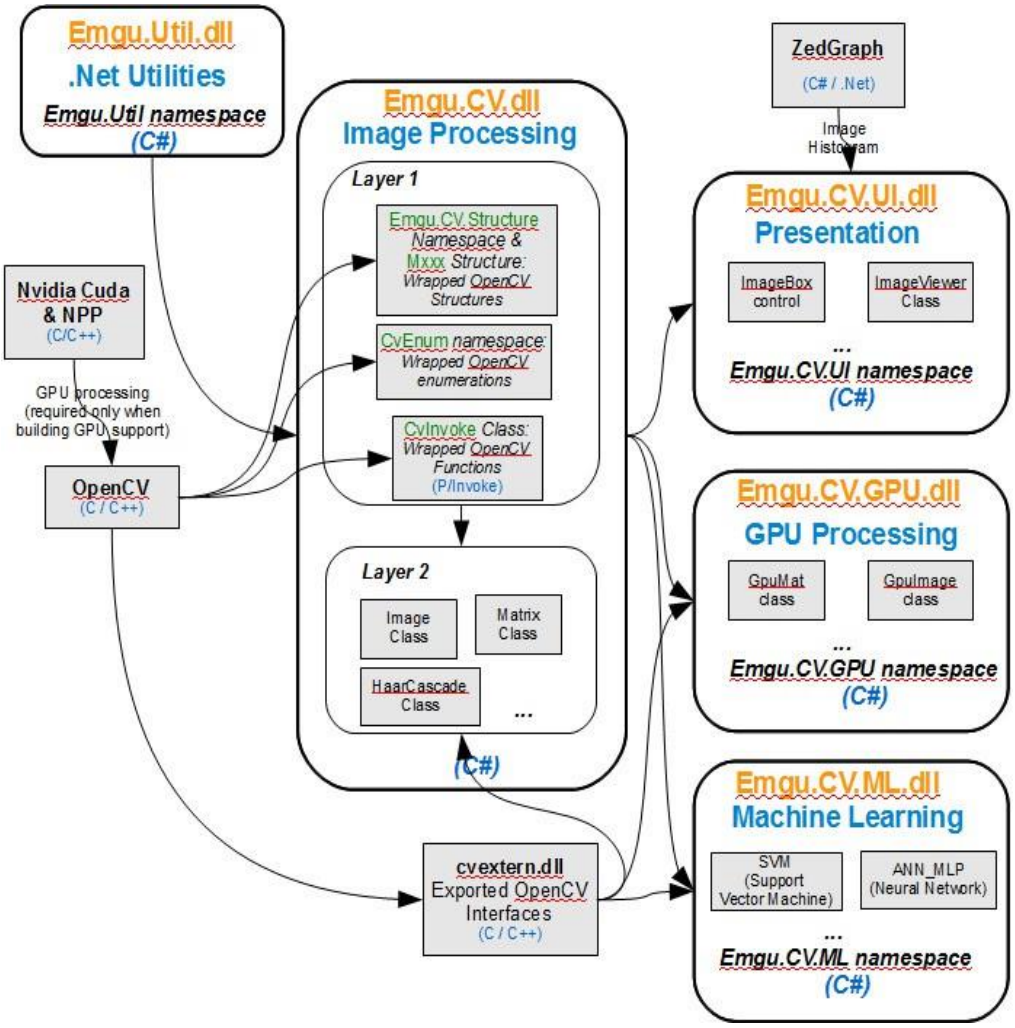

Рисунок 1.4 – Архитектура Emgu CV

– Базовый уровень (уровень 1) содержит сопоставления функций, структур и перечислений, которые напрямую отражают таковые в OpenCV.

– Второй уровень (уровень 2) содержит классы, сочетающие в себе преимущества мира .NET.

Emgu CV также заимствует некоторые существующие структуры в .Net для представления структур в OpenCV:

– Структура .Net

– Структура OpenCV

– System.Drawing.Point CvPoint

– System.Drawing.PointF CvPoint2D32f

– System.Drawing.Size CvSize

– System.Drawing.Rectangle CvRect

– Сопоставление перечислений — Emgu.CV.CvEnum

Пространство имен CvEnum обеспечивает прямое сопоставление с перечислениями OpenCV. Например, CvEnum.IPL_DEPTH.IPL_DEPTH_8U имеет то же значение, что и IPL_DEPTH_8U в OpenCV; оба из которых равны 8.

Глубина и цвет как общий параметр

Изображение определяется общими параметрами: цветом и глубиной. Чтобы создать 8-битное беззнаковое изображение в оттенках серого, в Emgu CV это делается путем вызова «image <Gray, Byte> image = new image <Gray, Byte> (width, height);»

Этот синтаксис не только позволяет узнать цвет и глубину изображения, но также ограничивает способы использования функций и фиксирует ошибки во время компиляции. Например, функция SetValue(TColor color, Image<Gray, Byte> Mask) в классе Image<TColor, TDepth> будет принимать цвета только одного типа, а маска должна иметь 8-битное беззнаковое изображение в оттенках серого. Любые попытки использовать в качестве маски 16-битное изображение с плавающей запятой или изображение без оттенков серого приведут к ошибке времени компиляции.

Создание изображения

Использование управляемого класса Image <TColor, TDepth>.

– Память автоматически освобождается сборщиком мусора.

– Класс Image <TColor, TDepth> можно проверить с помощью визуализатора отладчика.

– Класс Image <TColor, TDepth> содержит расширенный метод, недоступный в OpenCV, например, общие операции с пикселями изображения, преобразование в растровое изображение и т. д.

1. Цвет изображения

Первый общий параметр класса Image определяет цвет типа изображения. Например, image <gray, ...> img1;

указывает, что img1 представляет собой одноканальное изображение в оттенках серого.

Поддерживаемые типы цвета:

– Серый
– Bgr (Синий Зеленый Красный)
– Bgra (Синий Зеленый Красный Альфа)
– Hsv (значение насыщенности оттенка)
– Hls (оттенок, яркость, насыщенность)
– Lab (CIE L*a*b*)
– Luv (CIE L*u*v*)
– Xyz (CIE XYZ.Rec 709 с точкой белого D65)
– Ycc (YCrCb JPEG)

2. Глубина изображения

Глубина изображения задается с помощью второго универсального параметра Depth. Типы глубины, поддерживаемые в Emgu CV

– Byte
– SByte
– Single (float)
– Double
– UInt16
– Int16
– Int32 (int)

3. Создание нового изображения

Создать изображение размером 480x320 с цветом Bgr и 8-битной беззнаковой глубиной. Код на C# будет:

Image <Bgr, Byte> img1 = new Image <Bgr, Byte> (480, 320);

Если указать значение фона изображения в синем цвете. Код на C# будет:

Image <Bgr, Byte> img1 = new Image <Bgr, Byte> (480, 320, new Bgr (255, 0, 0));

4. Чтение изображения из файла

Image <Bgr, Byte> img1 = new Image <Bgr, Byte>("MyImage.jpg");

5. Создание изображения из растрового изображения

Также возможно создать Image <TColor, TDepth> из объекта .Net Bitmap. Код на C# будет:

Image <Bgr, Byte> img = new Image <Bgr, Byte>(bmp);

Где bmp — растровое изображение

6. Автоматический сбор мусора

Класс Image <TColor, TDepth> автоматически отвечает за управление памятью и сборку мусора.

Как только сборщик мусора решит, что ссылки на объект Image <TColor, TDepth> больше нет, он вызовет метод Dispose, который освобождает неуправляемую структуру IplImage.

Время, когда сборщик мусора решит удалить образ, не гарантируется. При работе с большим изображением вызывается метод Dispose() для явного освобождения объекта. Альтернативно используется ключевое слово using в C#, чтобы ограничить область изображения. Используя (Image <Gray, Single> image = new Image <Gray, Single> (1000, 800)) будет размещено изображение и освобождена память

Методы

– Метод XYZ в классе Image <TColor, TDepth> соответствует функции OpenCV cvXYZ. Например, функция Image <TColor, TDepth>.Not() соответствует функции cvNot, возвращающей результирующее изображение.

– Метод _XYZ обычно аналогичен методу XYZ, за исключением того, что операция выполняется на месте, а не возвращает значение.

Например, функция Image <TColor, TDepth>._Not() выполняет побитовую инверсию на месте.

1. Перегрузка операторов

Операторы + - * / могут быть перегружены поэтому совершенно законно писать такие коды, как: Image <Gray, Byte> image3 = (image1 + image2 - 2.0) * 0.5;

2. Общая операция

Одним из преимуществ использования Emgu CV является возможность выполнять типовые операции.

Пример. Есть изображение в оттенках серого, состоящее из байтов Image <Gray, Byte> img1 = new Image <Gray, Byte> (400, 300, new Gray (30));

Чтобы инвертировать все пиксели этого изображения, можно вызвать функцию Not.

image <Gray, Byte> img2 = img1.Not();

В качестве альтернативы также можно использовать универсальный метод Convert, доступный из класса Image <TColor, TDepth>.

Image<Gray, Byte> img3 = img1.Convert<Byte> (delegate (Byte b) {return (Byte) (255-b);});

Результирующее изображение img2 и img3 содержит одинаковое значение для каждого пикселя.

Использование универсальных операций не представляет большого преимущества. Фактически, OpenCV имеет реализацию функции Not и с точки зрения производительности она лучше, чем универсальная версия эквивалентного вызова функции Convert. Однако бывают случаи, когда универсальные функции обеспечивают гибкость лишь с незначительным снижением производительности.

К примеру, есть Image <Gray, Byte> img1 с набором пикселей. Надо создать одноканальное изображение с плавающей запятой того же размера, где каждый пиксель нового изображения соответствует старому изображению, описанному с помощью следующего делегата.

delegate (byte b) {return (Single) Math.cos (b * b / 255.0);}

Эту операцию можно выполнить в Emgu CV следующим образом:

Image<Gray, Single> img4 = img1.Convert<Single> (delegate (Byte b) {return (Single) Math.cos (b * b / 255.0);});

Синтаксис прост. С другой стороны, эту операцию в OpenCV выполнить сложно, поскольку эквивалентная функция, такая как Math.cos, недоступна.

3. Преобразование цвета и глубины

Преобразовать Image <TColor, TDepth> между разными цветами и глубинами очень просто. Например, есть Image <Bgr, Byte> img1 и надо преобразовать его в изображение Single в оттенках серого:

Image<Gray, Single> img2 = img1.Convert<Gray, Single> ();

4. Отображение изображения

Использование ImageBox

Emgu CV рекомендует использовать элемент управления ImageBox для отображения по следующим причинам.

– ImageBox — это высокопроизводительный элемент управления для отображения изображений. По возможности он отображает растровое изображение, которое разделяет память с объектом изображения, поэтому копирование памяти не требуется (очень быстро).

– Пользователь сможет проверить значения пикселей изображения, частоту кадров видео, типы цветов при отображении изображения.

– Простые операции с изображением удобно выполнять всего несколькими щелчками мыши.

5. Преобразование в растровое изображение

Класс Image имеет функцию ToBitmap (), которая возвращает объект Bitmap, который можно отобразить в элементе управления PictureBox с помощью формы Windows.

6. Преобразование в XML

Можно использовать код для преобразования изображения:

Image <Bgr, Byte> в XmlDocument: StringBuilder sb = new StringBuilder ();

(new XmlSerializer (typeof (Image<Bgr, Byte>))). Serialize (new StringWriter (sb), o);

XmlDocument xDoc = new XmlDocument ();

xDoc.LoadXml(sb.ToString());

7. Преобразование из XML

XmlDocument xDoc в Image <Bgr,Byte> Image<Bgr, Byte> image = (Image<Bgr, Byte>)

(new XmlSerializer(typeof(Image<Bgr, Byte>))).Deserialize(new XmlNodeReader(xDoc));

Пример использования «Emgu CV»

Ниже приведен пример (рис. 1.5), показывающий загрузку изображения, его преобразование в черно-белое и отображение на форме «Windows Forms»:

```
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using System;
using System.Drawing;
using System.Windows.Forms;

namespace EmguCVExample
{
    public partial class MainForm : Form
    {
        private ImageBox pictureBox;
        private Button loadButton;

        public MainForm()
        {
            InitializeComponent();

            pictureBox = new ImageBox();
            pictureBox.Dock = DockStyle.Fill;
            Controls.Add(pictureBox);

            loadButton = new Button();
            loadButton.Text = "Load Image";
            loadButton.Dock = DockStyle.Bottom;
            loadButton.Click += LoadButton_Click;
            Controls.Add(loadButton);
        }

        private void LoadButton_Click(object sender, EventArgs e)
        {
            using (OpenFileDialog openFileDialog = new OpenFileDialog())
            {
                openFileDialog.Filter = "Image files (*.jpg, *.jpeg, *.png) | *.jpg; *.jpe

                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    string filePath = openFileDialog.FileName;
                    Mat img = CvInvoke.Imread(filePath, ImreadModes.Color);
                    pictureBox.Image = img.ToBitmap();
```

Рисунок 1.5 – Использование библиотеки машинного обучения

При нажатии на кнопку «Загрузить» открывается диалоговое окно для выбора изображения, после чего оно загружается и отображается на форме.

2. ПРОЕКТНЫЙ РАЗДЕЛ

2.1 Разработка автоматизированной подсистемы контроля параметров

2.1.1 Постановка задачи

Функции программы:

– Подсчёт количества объектов на одном изображении с возможностью подсчета параметров;

– Загрузка и отображение изображений;

– Обработка изображений;

– Вычисление параметров;

– Предсказание веса;

– Измерение размеров;

– Интеграция и отображение результатов;

Требования к интерфейсу:

– Интерфейс должен обеспечивать необходимый функционал;

– Интерфейс должен быть интуитивно понятен пользователю;

– Реализация посредством WindowsForms.

– Требования к программной платформе:

– 64-разрядная операционная система Windows 7, 8, 10

2.1.2 Архитектура системы

Архитектура приложения разделена на несколько слоев, каждый выполняет определенные функции (рис. 2.1):

1. Презентационный слой отвечает за взаимодействие с пользователем и отображение результатов.

2. Уровень обработки данных содержит методы и алгоритмы обработки и анализа изображений.

3. Уровень машинного обучения применяет модель для оценки веса грузовика на основе изображения.

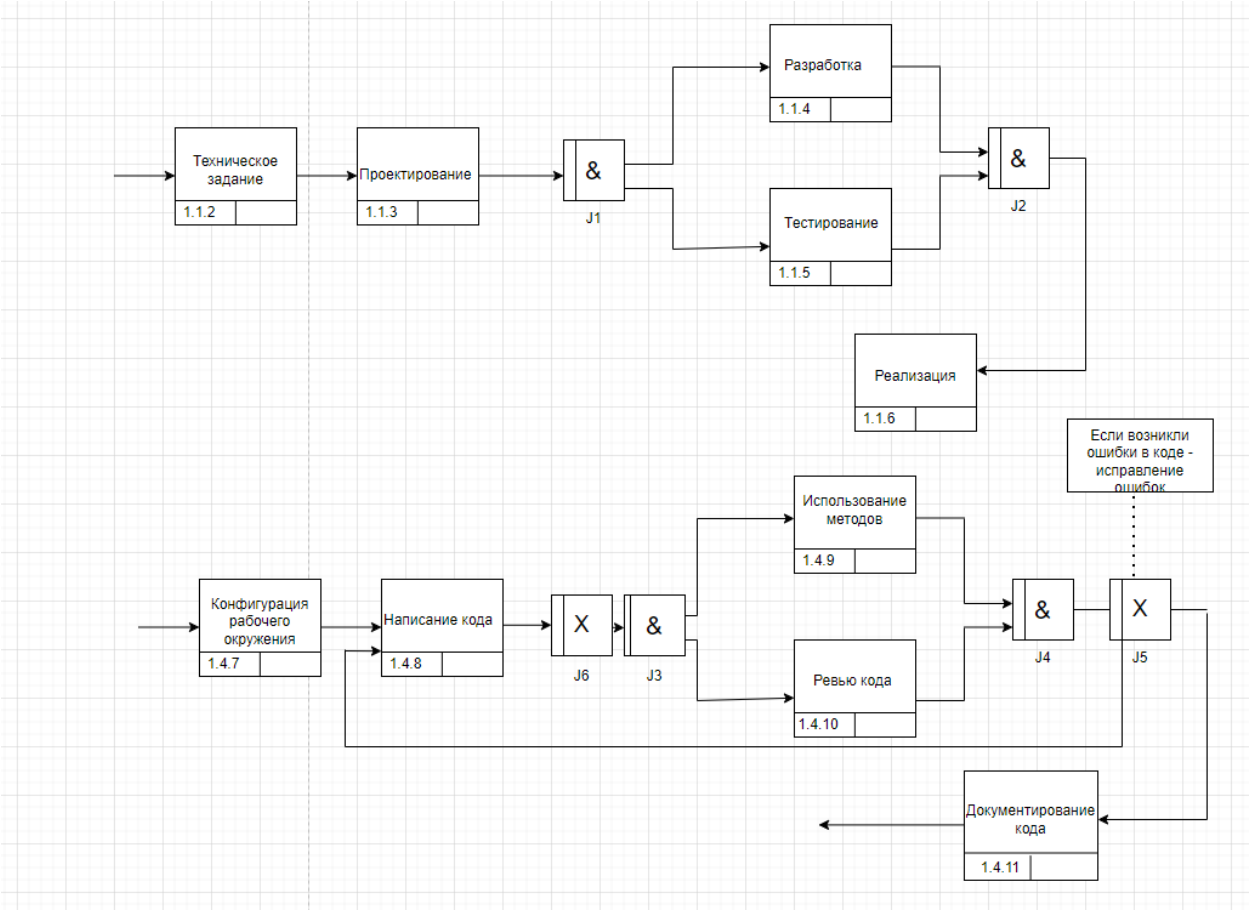

Рисунок 2.1 – Диаграмма IDF3 и ее декомпозиция

Архитектура приложения позволяет эффективно организовать процесс обработки изображений и применение методов машинного обучения для решения конкретных задач.

2.1.3 Диаграмма классов

Диаграмма классов приложения, это визуальное представление основных классов и их взаимосвязей.

Дабы предоставить пользователю нужные данные необходимо для начала составить запрос, скоординироваться с оператором и в конце приступить к анализу данных (рис. 2.2-2.3).

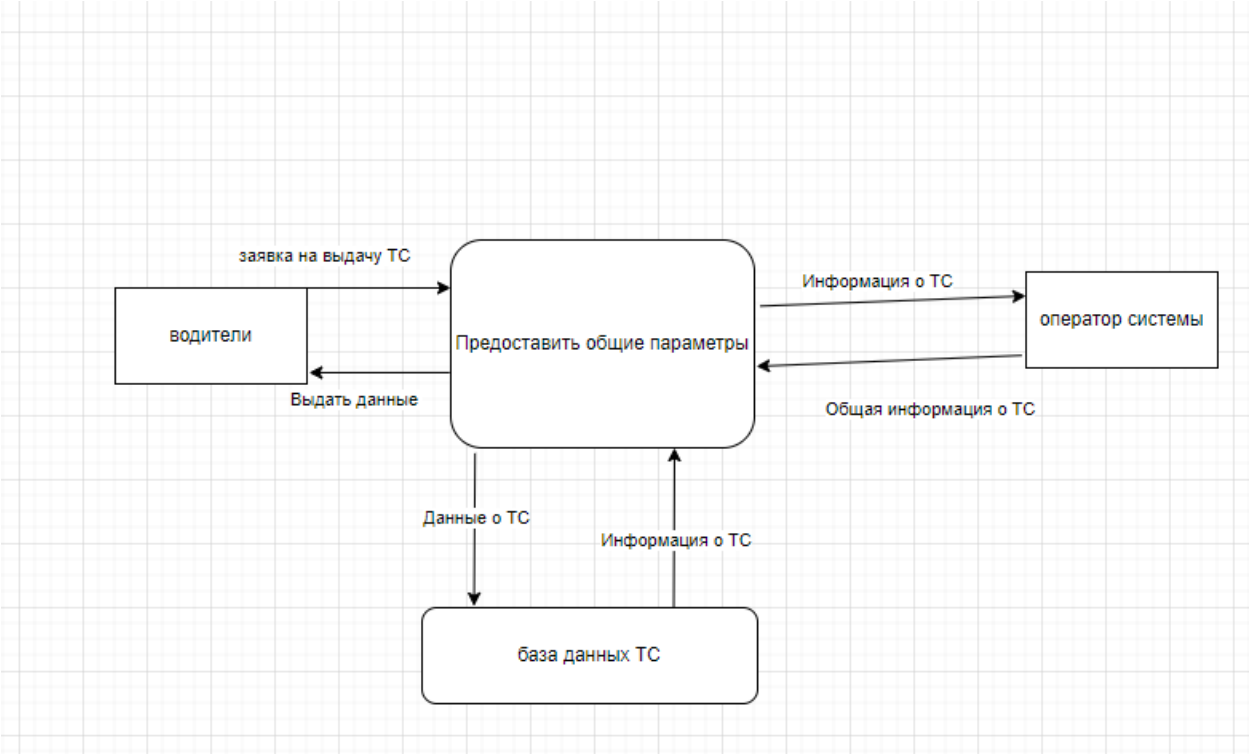

Рисунок 2.2 – DFD диаграмма интеграции в приложение

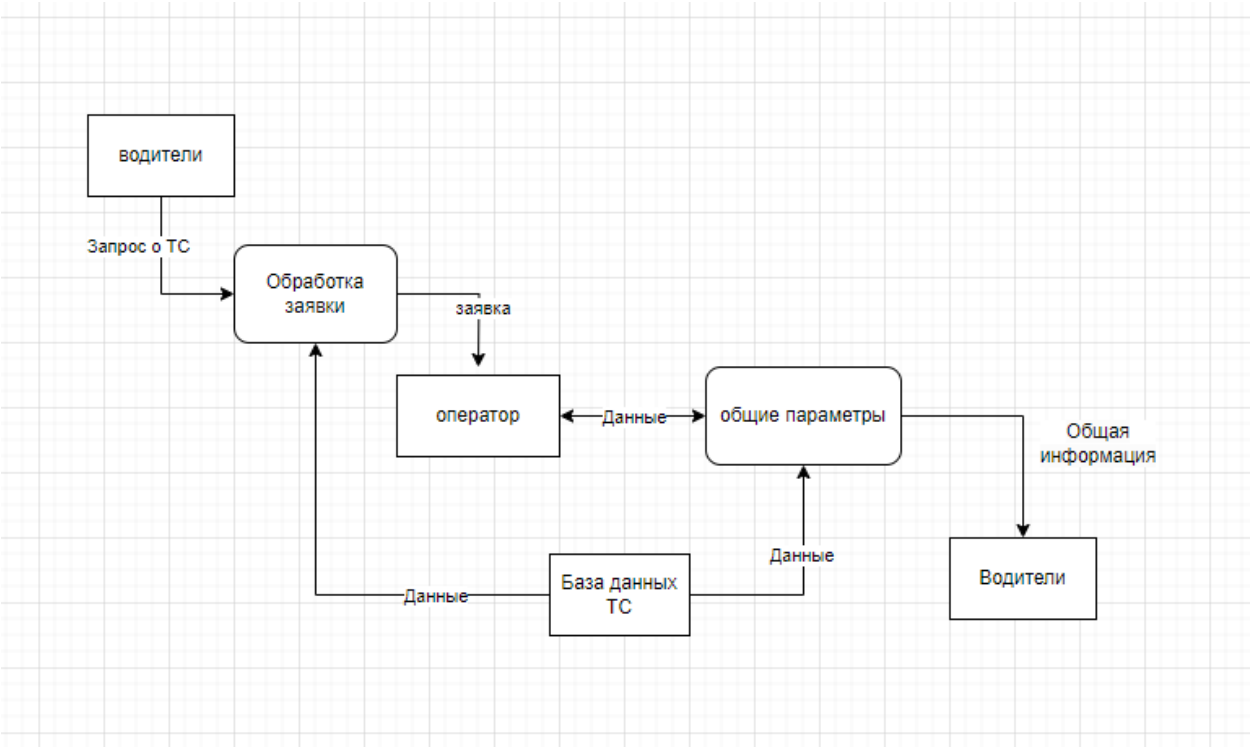

Рисунок 2.3 – Декомпозиция DFD диаграммы

Изначально в программе пользователь может загружать изображение, видеть проанализированный результат, в который входят размер, вес и аналогичные составляющие.

Диаграмма прецедентов (рис.2.4).

Диаграмма кратко объясняет какие действия пользователь может предпринимать и как на это реагирует система.

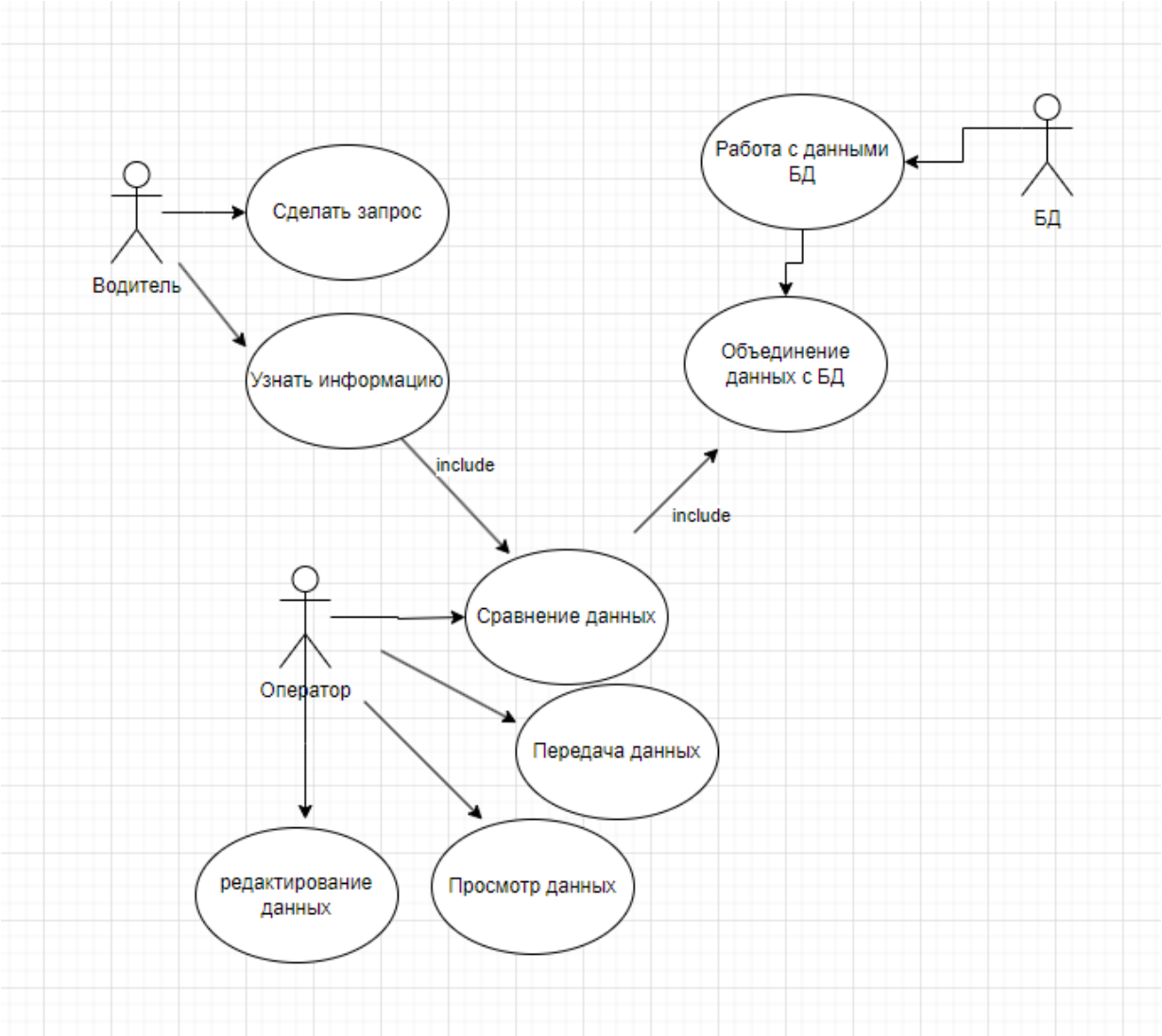

Рисунок 2.4 – Диаграмма прецедентов

На диаграмме наглядно показано какие действия может выполнить пользователь.

1. Узнать информацию о загруженном изображении;
2. Запустить обработку и анализ изображения;
3. Увидеть результат анализа и расчитанные параметры;

2.1.4 Основные компоненты

1. .Net Framework это платформа для разработки программного обеспечения, разработанная Microsoft для строительства и эксплуатации Windows Приложения. Платформа .Net состоит из инструментов разработчика, языков программирования и библиотек для создания настольных и веб-приложений.

2. Windows Forms — интерфейс программирования приложений, отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде.

3. Emgu CV - это кросс-платформенная обертка над OpenCV, написанная на C#.

– Используется в задачах компьютерного зрения и обработки изображений.

– Может использоваться под Windows, Linux, Mac OS, iOS, Android. Совместима с любым языком .NET.

4. OpenCV — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на Си/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков.

Компоненты Windows Forms

Windows Forms предлагает использование многих административных компонентов для создания пользовательского интерфейса.

Главные модули управления приложения:

Program.cs – Создание кода для входа в программу;

Form1.cs – Сюда входит интерфейс программы и весь написанный код;

Button – Кнопка для загрузки изображения;

PictureBox – Выводит изображение на экран;

Label – Таблички для отображения информации;

TextBox – Поле для текста;

OpenFileDialog – окно для выбора файла;

Данные модули были использованы при написании приложения, что позволило создать пользовательский интерфейс приложения (рис. 2.5).

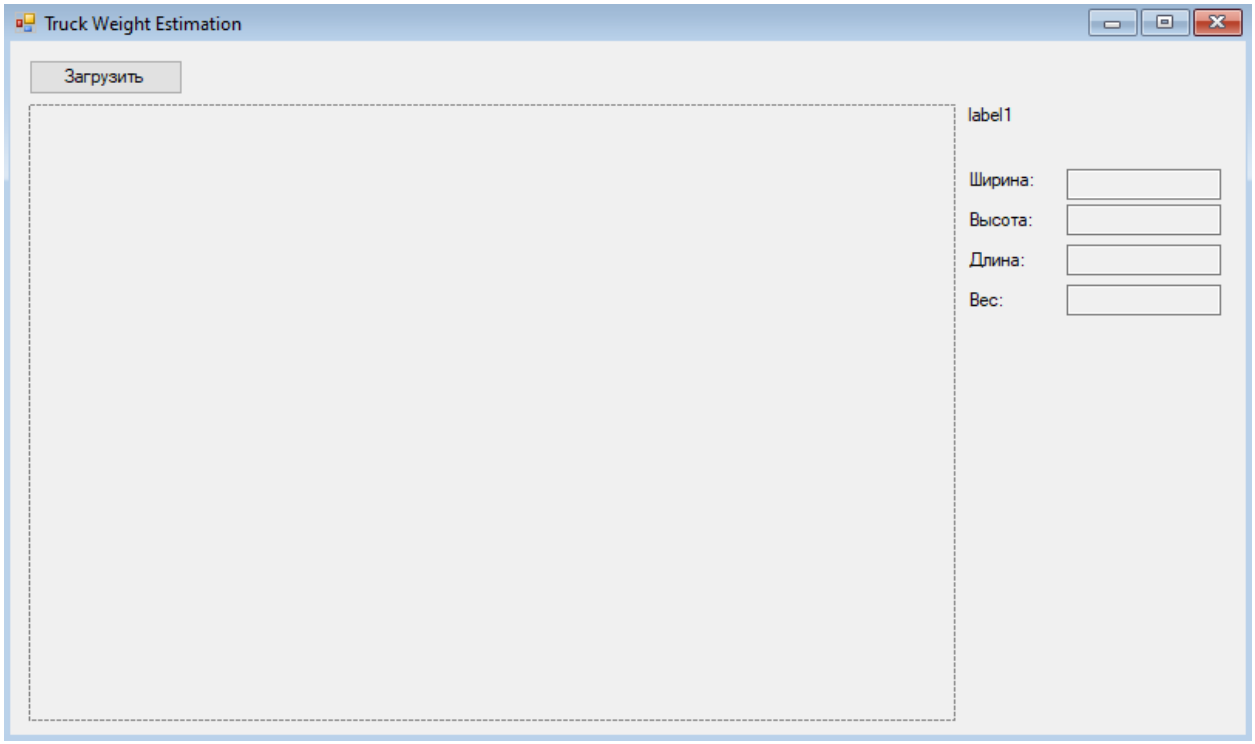

Рисунок 2.5 – Дизайн приложения

2.1.5 Этапы разработки

На первом этапе для системы устанавливаются ключевые показатели эффективности и конечные условия к требованию системы.

На втором этапе разрабатываются алгоритмы и модели, создается архитектура, выбирается технология и инструменты.

На третьем этапе все внимание уделяется к сборке и интеграции компонентов, тестируются и разрабатываются аппаратные компоненты.

На четвертом этапе проверяется соответствие ко всем требованиям и проводится проверка системы.

На пятом этапе проводится обучение персонала (операторов, технических специалистов), разрабатываются инструкции по эксплуатации и руководств по использованию, завершается проверка системы.

На заключительном этапе создается техническая поддержка и обслуживание ПО, устраняются всевозможные ошибки и баги.

2.1.6 Внедрение и оценка

При внедрении важным аспектом являются подготовка и координация, так же необходимо учитывать масштабируемость компании, наличия оборудования, а, так же функции обслуживания и сопровождения технической части.

Если большинство аспектов удовлетворяет требованиям, то успешное внедрение повысит безопасность дорожного движения, сократит нарушения ПДД к минимуму, улучшит состояние дорожной инфраструктуры и обеспечит соблюдение норм и правил.

Для лучшей координации и эксплуатации необходимо совершенствовать параметры мониторинга на основе будущих отзывов.

2.2 Практическое внедрение и тестирование

2.2.1 Используемые классы и методы

В программе используются классы и методы из подключенной библиотеки Emgu.CV. Перечень используемых классов изображен в таблице 2.1, перечень используемых методов отображен в таблице 2.2.

Таблица 2.1 – Используемые классы библиотеки.

№	Класс	Описание
1	Emgu.CV.Mat	Матрица изображения, используемая для хранения и обработки изображений
2	Emgu.CV.CvInvoke	Класс для вызова методов OpenCV
3	Emgu.CV.Image<TColor, TDepth>	Обобщенный класс для работы с изображениями

4	Emgu.CV.Structure.MCvScalar	Структура для представления цвета
5	Emgu.CV.Util.VectorOfVectorOfPoint	Контейнер для хранения векторов точек, представляющих контуры
6	Emgu.CV.CvEnum.ImreadModes	Перечисление для указания режима чтения изображений
7	Emgu.CV.CvEnum.RetrType	Перечисление для указания типа ретривации контуров
8	Emgu.CV.CvEnum.ChainApproxMethod	Перечисление для указания метода аппроксимации контуров
9	Emgu.CV.Structure.Bgr	Структура для представления цветового пространства BGR
10	Emgu.CV.Structure.Gray	Структура для представления оттенков серого

Таблица 2.2 – Используемые методы библиотеки.

№	Метод	Описание
1	CvInvoke.Imread(string filename, ImreadModes flags)	Чтение изображения из файла с указанным режимом
2	CvInvoke.CvtColor(IInputArray src, IOutputArray dst, ColorConversion code)	Преобразование цветового пространства изображения
3	CvInvoke.GaussianBlur(IInputArray src, IOutputArray dst, Size ksize, double sigmaX)	Применение гауссовского размытия к изображению
4	CvInvoke.Canny(IInputArray image, IOutputArray edges, double threshold1, double threshold2)	Обнаружение границ с использованием алгоритма Кенни
5	CvInvoke.FindContours(IInputOutputArray image, IOutputArrayOfArrays contours, IOutputArray hierarchy, RetrType mode, ChainApproxMethod method)	Поиск контуров на бинарном изображении
6	CvInvoke.BoundingRectangle(VectorOfPoint points)	Вычисление ограничивающего прямоугольника для заданного контура
7	CvInvoke.Rectangle(IInputOutputArray img, Rectangle rect, MCvScalar color, int thickness)	Рисование прямоугольника на изображении
8	Image<TColor, TDepth>.Convert<TNewColor>()	Преобразование изображения в другой цветовой формат
9	Image<TColor, TDepth>.ThresholdBinary(Gray thresh, Gray maxValue)	Применение бинарного порогового преобразования к изображению
10	Image<TColor, TDepth>.ToBitmap()	Преобразование изображения в формат Bitmap для отображения в Windows Forms

Класс Form1 представляет собой основной класс формы Windows Forms, содержащий элементы управления и логику обработки событий.

– Form1() – конструктор класса, инициализирующий элементы управления на форме.

– btnLoadImage_Click(object sender, EventArgs e) – метод-обработчик события для загрузки изображения.

– CalculateAndDisplayResults(Bitmap image) – основной метод для обработки изображения и вычисления параметров.

– BitmapToImage<TColor, TDepth>(Bitmap bitmap) – преобразование изображения из формата Bitmap в формат Emgu CV.

Класс ImageProcessor отвечает за выполнение операций по обработке изображений, таких как преобразование в оттенки серого, бинаризация и обнаружение контуров.

– ConvertToGrayScale(Image<Bgr, byte> img) – преобразование цветного изображения в оттенки серого.

– BinarizeImage(Image<Gray, byte> grayImage) – преобразование изображения в бинарное.

– FindContours(Image<Gray, byte> binaryImage) – обнаружение контуров на бинарном изображении.

– CalculateWheelSizes(VectorOfVectorOfPoint contours) – вычисление размеров.

Класс WeightPredictor отвечает за предсказание веса

– PredictWeight(double totalArea, int numberOfWheels) – метод для предсказания веса грузовика.

– CalculateAverageWheelSize(List<Size> wheelSizes) – вычисление среднего размера.

– CalculateTruckWeight(double axleLoad, int numberOfWheels) – оценка общего веса грузовика.

Работа с программным обеспечением

Запуская программу, пользователь видит интерфейс (рис.2.6), на котором есть PictureBox (рис. 2.7) для отображения изображения, и Button (рис. 2.8) для загрузки.

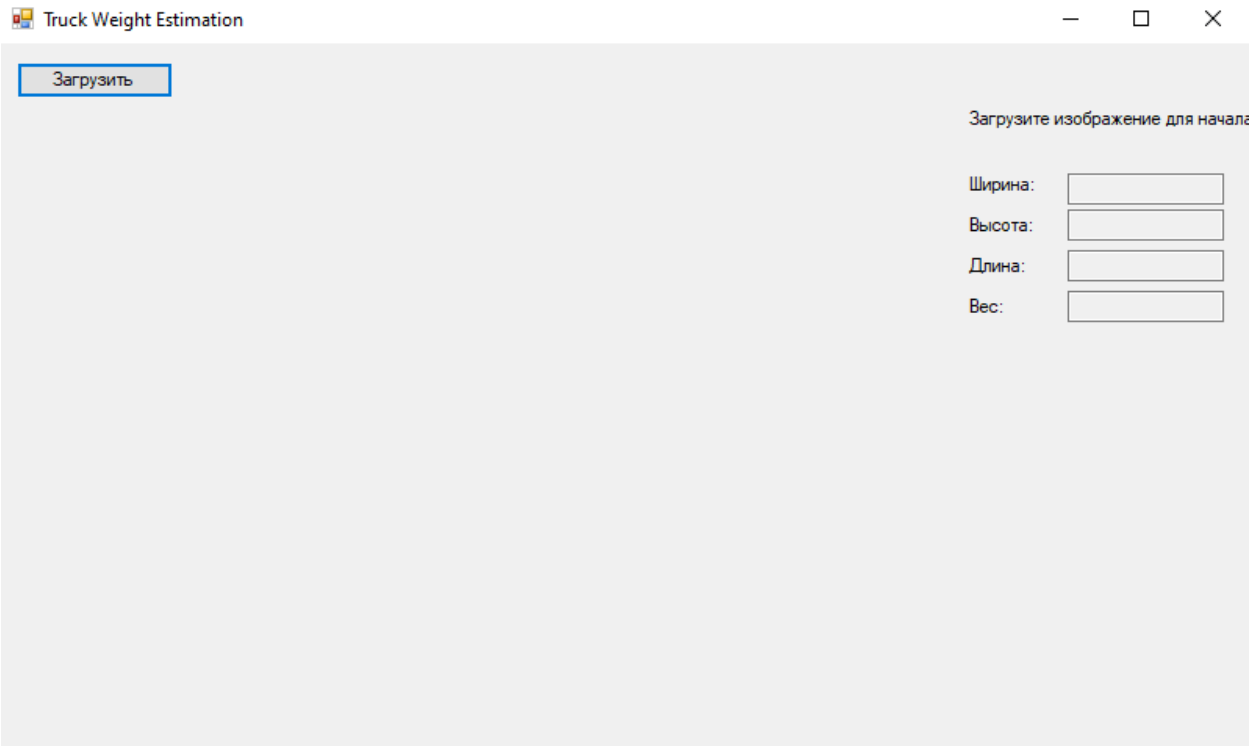

Рисунок 2.6 – Основная форма

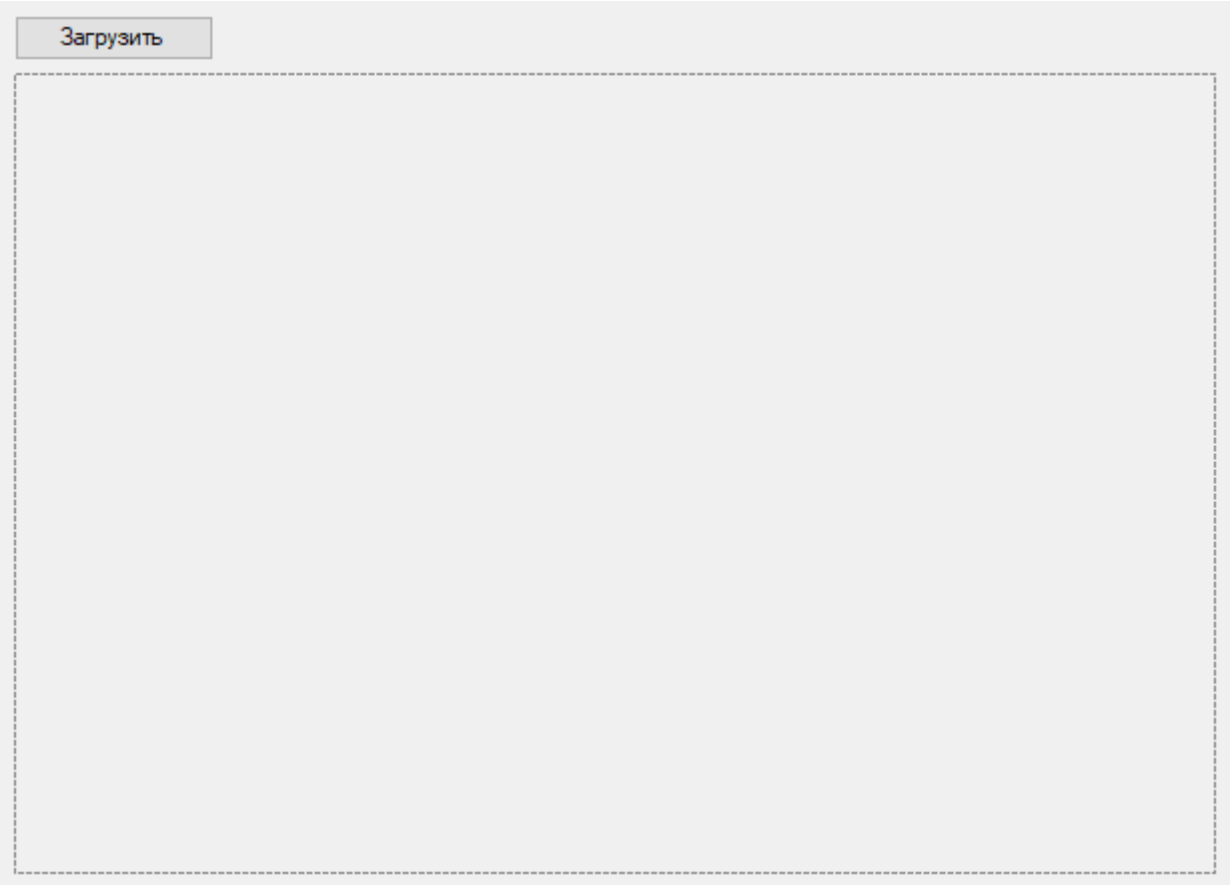

Рисунок 2.7 – Поле рисунка.

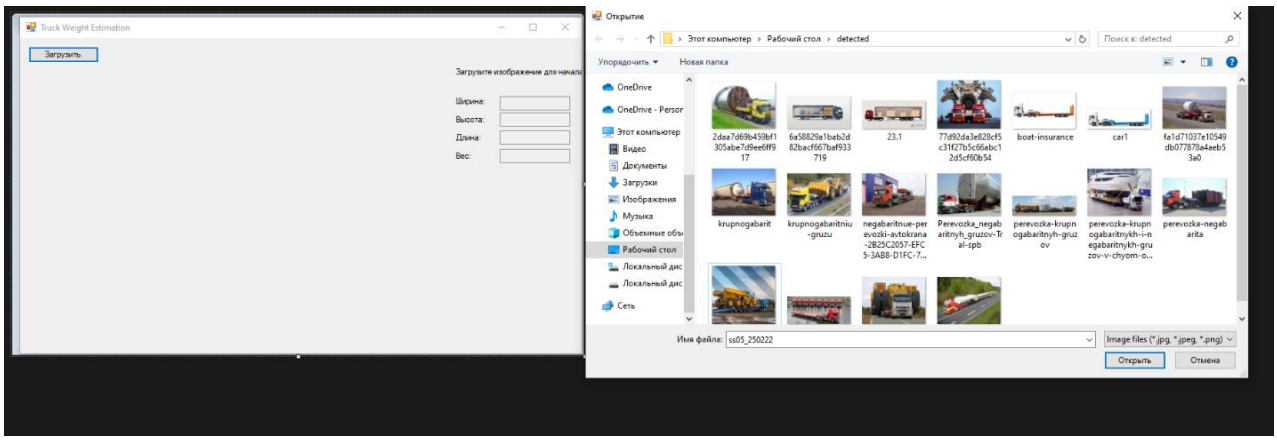

Рисунок 2.8 – Загрузка с локального диска

При нажатии на кнопку «Загрузить» (рис 2.9) открывается окно «OpenFileDialog», которое позволяет пользователю выбрать и загрузить фото с локального диска (рис.2.10).

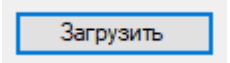

Рисунок 2.9 – Кнопка запуска.

```
Ссылок: 1
private void btnLoadImage_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "Image files (*.jpg, *.jpeg, *.png) | *.jpg; *.jpeg; *.png";
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        string filePath = openFileDialog.FileName;
        loadedImage = new Bitmap(filePath);
        pictureBox.Image = loadedImage;
        CalculateAndDisplayResults(loadedImage);
    }
}
```

Рисунок 2.10 – Код вызова изображения

После выбора файла изображение загружается в память и выводится в PictureBox (рис. 2.11):

Рисунок 2.11 – Отображение изображения в PictureBox

После загрузки, с помощью библиотеки для дальнейшей обработки изображение преобразуется формат «Emgu», (рис. 2.12):

```
// Преобразование изображения в формат Emgu CV
Image<Bgr, byte> img = BitmapToImage<Bgr, byte>(image);
Image<Gray, byte> grayImage = img.Convert<Gray, byte>();
Image<Gray, byte> binaryImage = grayImage.ThresholdBinary(new Gray(128), new Gray(255));
```

Рисунок 2.12 – Вызов библиотеки «Emgu CV»

На этом этапе картинка сначала преобразуется в оттенки серого (Convert<Gray, byte>()), а затем бинаризуется с помощью порогового значения (ThresholdBinary(new Gray(128), new Gray(255))) (рис. 2.13).

```
// Обнаружение контуров
VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint();
Mat hierarchy = new Mat();
CvInvoke.FindContours(binaryImage, contours, hierarchy, RetrType.External, ChainApproxMethod.ChainApproxSimple);
```

Рисунок 2.13 - Обнаружение контуров

Далее начинается работа функции CvInvoke.FindContours, которая принимает бинаризованное изображение и возвращает список контуров, после этого нужно провести анализ (рис. 2.14).

```
double totalArea = 0.0;
List<Point> centers = new List<Point>();
List<Size> wheelSizes = new List<Size>(); // Список размеров колес
Rectangle truckRect = Rectangle.Empty;
int truckWidth = 0;
int truckHeight = 0;
int truckLength = 0;
int truckLe = 0;
```

Рисунок 2.14 - Анализ объектов

Далее для каждого контура вычисляется прямоугольник, описывающий его границы BoundingRectangle, если он соответствует параметрам, то программа говорит что это колесо.

По информации о размерах оценивается нагрузка на оси и общий вес грузовика (рис. 2.15):

```
double loadPerWheel = 10.0;
double totalLoadInTons = totalArea / loadPerWheel;

int numberOfWheels = wheelSizes.Count;
Size averageWheelSize = CalculateAverageWheelSize(wheelSizes);

double truckWeight = CalculateTruckWeight(totalLoadInTons, numberOfWheels);
```

Рисунок 2.15 - Вычисление нагрузок и веса

Далее, из полученных данных, функция CalculateAverageWheelSize вычисляет средний размер колеса, показанный на рисунке 2.16.

```
Ссылок: 1
private Size CalculateAverageWheelSize(List<Size> wheelSizes)
{
    int totalWidth = 0;
    int totalHeight = 0;

    foreach (Size size in wheelSizes)
    {
        totalWidth += size.Width;
        totalHeight += size.Height;
    }

    int averageWidth = totalWidth / wheelSizes.Count;
    int averageHeight = totalHeight / wheelSizes.Count;

    return new Size(averageWidth, averageHeight);
}
```

Рисунок 2.16 - Нахождение размера колеса

На рисунке 2.17 показан вывод результатов с использованием элемента управления «Label»:

```
pictureBox.Image = img.ToBitmap();
lblResult.Text = $"Общая нагрузка на оси: {totalLoadInTons} кг\nОценочный вес грузовика: {truckWeight} т\n";
textBoxWidth.Text = $"{realTruckWidth:F2} м";
textBoxHeight.Text = $"{realTruckHeight:F2} м";
textBoxLength.Text = $"{realTruckLength:F2} м";
textBoxWeight.Text = $"{truckWeight:F2} т";
```

Рисунок 2.17 - Вывод результатов

На рисунке 2.18 программа выдает готовое изображение

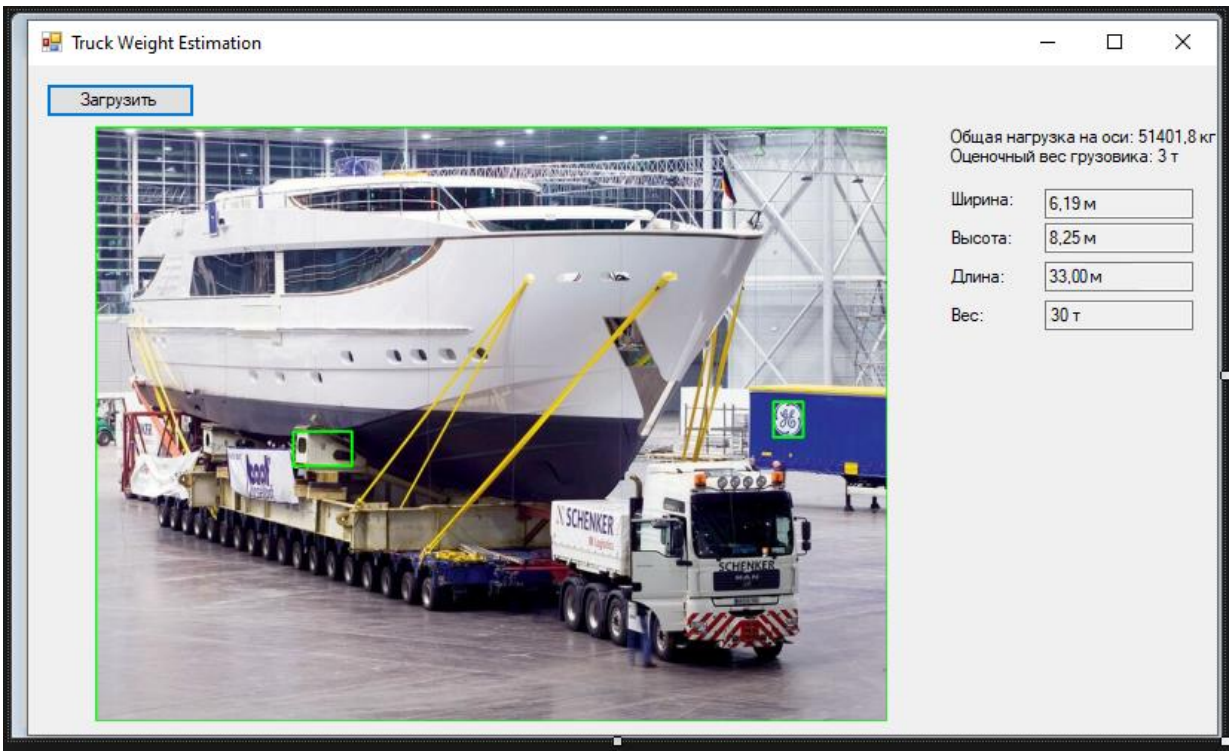

Рисунок 2.18 – Обработанное изображение

Если пользователь захочет, он может повторно загружать изображения и обновлять данные.

В конечном итоге, результаты могут быть экспортированы в файл (PDF или CSV).

2.2.2 Тест и анализ

Анализ результатов необходим для проведения оценки производительности. Каждый сеанс сохраняется в системе и может использоваться для отчетов.

Процесс тестирования программного продукта включает такие этапы как:

1. Изучение и анализ предмета.
2. Планирование.
3. Исполнение.

Специфика тестирования начинается еще до утверждения спецификации и продолжается на стадии разработки (кодирования) программного обеспечения. Конечной целью этапа изучения и анализа является получение ответов на два вопроса: какие функциональности предстоит протестировать, как эти функциональности работают.

Планирование происходит на стадии разработки (кодирования) программного обеспечения. На этой стадии перед тестировщиком стоит задача поиска компромисса между объемом тестирования, который возможен в теории, и объемом, который возможен на практике.

Выполнение тестирования происходит на стадии тестирования и представляет собой практический поиск дефектов с использованием тестовой документации.

Функциональное тестирование

Функциональные тесты основаны на функциях и возможностях, а также на взаимодействии с другими системами, и могут проводиться на всех уровнях тестирования:

– компонентное/модульное тестирование;
– интеграционное тестирование;
– системное тестирование;
– тестирование применений (acceptance testing)

Функциональные типы тестирования рассматривают внешнее поведение системы.

Основная задача функционального тестирования - подтвердить, что разрабатываемый продукт обладает всеми функциональными возможностями, необходимыми заказчику.

Как и любой другой вид тестирования, функциональное тестирование имеет свои преимущества и недостатки.

Преимущества функционального тестирования:

– оно имитирует реальное использование системы.

Недостатки функционального тестирования:

– возможность пропустить логические ошибки в приложении;

– вероятность избыточного тестирования.

1. Тестирование систем безопасности и контроля доступа

Стратегия тестирования безопасности направлена на проверку безопасности системы и обеспечение всеобъемлющего подхода, когда речь идет о защите приложения, предотвращении хакерских атак, вирусов и несанкционированного доступа к конфиденциальным данным.

Общая стратегия безопасности основывается на трех основных принципах:

– политика конфиденциальности;

– целостность;

– доступность.

В него входит:

– измерение времени выполнения выбранных операций при индивидуальной интенсивности этих операций;

– определяется максимальное количество пользователей, которые одновременно могут работать с приложением.

2. Тестирование установки

Тестирование установки направлено на проверку успешной установки и конфигурации приложения, а также обновления или удаления приложения.

ЗАКЛЮЧЕНИЕ

В результате выпускной квалификационной работы была реализована разработка приложения контроля параметров крупногабаритных и тяжеловесных транспортных средств. Изучены методы обработки изображений, рассмотрены возможности значимых аналогов, закреплены знания языка программирования C#, получен опыт подключения библиотек в среду Visual Studio 2022 Community, в частности Emgu CV. Все задачи ВКР выполнены. Программа соответствует требованиям и готова для внедрения в программное обеспечение.

В дальнейшем возможно расширение функционала программы, вплоть до:

– Возможности создания отчета о вычислительной работе;

– Возможности выбора алгоритма расчета объектов;

– Повышения точности для частых случаев возникновения каких-либо помех (некоторые объекты накладываются друг на друга, большее изменение яркости и т. д.).

Данная программа показывает высокую эффективность, надежность и соответствие последним требованиям, Внедрение такой системы позволит улучшить безопасность дорожного движения, состояние дорожной инфраструктуры и упростить управление крупными и тяжеловесными транспортными средствами.

В заключение следует отметить, что автоматизированная подсистема мониторинга параметров КГТТС является важным инструментом обеспечения эффективности дорожного движения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Варнавский В. Г. Глобальная транспортно-логистическая инфраструктура //Мировая экономика и международные отношения. – 2020. – Т. 64. – №. 1. – С. 5-14.

2. Балгабеков Т. К. Транспортные системы и перевозочный процесс. – 2019.

3. Вильданова Л. З., Яруллин Р. Р. ФАКТОРЫ, ВЛИЯЮЩИЕ НА ЭФФЕКТИВНОСТЬ ОРГАНИЗАЦИИ СИСТЕМЫ ВНУТРЕННЕГО КОНТРОЛЯ //Международный журнал гуманитарных и естественных наук. – 2023. – №. 6-1 (81). – С. 169-172.

4. Халмухамедов А. С., Хакимов Ш. К., Саматов Р. Г. ПЕРСПЕКТИВЫ ИСПОЛЬЗОВАНИЯ АВТОМАТИЧЕСКОЙ СИСТЕМЫ ВЕСОГАБАРИТНОГО КОНТРОЛЯ ТРАНСПОРТНЫХ СРЕДСТВ ПОДСИСТЕМЫ ИТС В РЕСПУБЛИКЕ УЗБЕКИСТАН //Экономика и социум. – 2022. – №. 1-2 (92). – С. 292-306.

5. Бахтиярова Е. А., Батаева Г. Б. Система распознавания номеров вагонов //Вестник Казахской академии транспорта и коммуникаций им. М. Тынышпаева. – 2019. – №. 4. – С. 308-313.

6. Майоров В. И., Юсупов М. Ф. К вопросу о правовом регулировании организации дорожного движения в России и зарубежных странах //Право и государство: теория и практика. – 2019. – №. 11 (179). – С. 241-244.

7. Бурченков В. В. Автоматизированные системы контроля подвижного состава. – 2020.

8. Фатеенков Д. В. Использование ML. NET для классификации пользовательских сообщений //Постулат. – 2022. – №. 1 январь.

9. Бредихин А. И. Алгоритмы обучения сверточных нейронных сетей //Вестник Югорского государственного университета. – 2019. – №. 1 (52). – С. 41-54.

10. Клюева И. А. Современные возможности и примеры внедрения машинного обучения //Оригинальные исследования. – 2021. – Т. 11. – №. 7. – С. 12-32.

11. Шелухин О. И., Раковский Д. И. Выбор метрических атрибутов редких аномальных событий компьютерной системы методами интеллектуального анализа данных //T-Comm-Телекоммуникации и Транспорт. – 2021. – Т. 15. – №. 6. – С. 40-47.

12. Тарабрин В. Ф. и др. УСТРОЙСТВО ИЗМЕРЕНИЯ ПАРАМЕТРОВ ДВИЖЕНИЯ ТРАНСПОРТНЫХ СРЕДСТВ. – 2020.

13. Филиппова К. В., Ильинская Е. В. ЗАРУБЕЖНЫЙ ОПЫТ ВНЕДРЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ ТРАНСПОРТНЫХ СИСТЕМ //СОВРЕМЕННЫЕ ПРОБЛЕМЫ ЭКОНОМИЧЕСКОГО РАЗВИТИЯ ПРЕДПРИЯТИЙ, ОТРАСЛЕЙ, КОМПЛЕКСОВ, ТЕРРИТОРИЙ. – 2021. – С. 205-208.

14. Медникова О. В., Матвиевская Т. Б. Дигитализация рынка транспорта и логистики: интеграция информационных систем. Российский опыт внедрения цифровых технологий в организации логистических процессов //Вестник академии знаний. – 2021. – №. 4 (45). – С. 197-204.

15. Козин Е. С., Базанов А. В. Система идентификации водителя транспортного средства на основе биометрических данных //Интеллект. Инновации. Инвестиции. – 2020. – №. 4. – С. 133-142.

16. Хрипченко М. С., Бусарин Э. Н., Кораблев Р. А. Метод повышения безопасности дорожного движения транспортных средств и снижение риска возникновения дорожно-транспортного происшествия. – 2019.

17. Чубейко С. В., Пузина Д. М. Сравнительная характеристика современных кроссплатформенных сред для разработки серверной части и API мобильного приложения //Труды Ростовского государственного университета путей сообщения. – 2019. – №. 2. – С. 101-105.

18. Петлицкий А. Ю. Разработка локальной корпоративной сети в образовательном учреждении. – 2019.

19. Николенко В. Системная инженерия на раз-два. – Litres, 2022.

20. Редкин П. А., Алёшкин А. С. Программный комплекс распределенного тестирования веб-приложений //International Journal of Open Information Technologies. – 2024. – Т. 12. – №. 4. – С. 125-132.

21. Коломытцева А. А. Разработка информационной системы для анализа ключевых показателей эффективности (KPI) персонала в сети филиалов компании «ДПО»: магистерская диссертация: дис. – 2023.

ПРИЛОЖЕНИЕ

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Emgu.CV.Util;
using System.Drawing.Imaging;
namespace wes
{
public partial class Form1 : Form
{
private Bitmap loadedImage;
public Form1()
{
InitializeComponent();
lblResult.Text = "Загрузите изображение для начала.";
}
private void btnLoadImage_Click(object sender, EventArgs e)
{
OpenFileDialog openFileDialog = new OpenFileDialog();
if (openFileDialog.ShowDialog() == DialogResult.OK)
{
string filePath = openFileDialog.FileName;
loadedImage = new Bitmap(filePath);
pictureBox.Image = loadedImage;
```

```
CalculateAndDisplayResults(loadedImage);

}

}

private void CalculateAndDisplayResults(Bitmap image)

{

// Преобразование изображения в формат Emgu CV

Image<Bgr, byte> img = BitmapToImage<Bgr, byte>(image);

Image<Gray, byte> grayImage = img.Convert<Gray, byte>();

Image<Gray, byte> binaryImage = grayImage.ThresholdBinary(new Gray(128), new Gray(255));

// Обнаружение контуров

VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint();

Mat hierarchy = new Mat();

CvInvoke.FindContours(binaryImage, contours, hierarchy, RetrType.External, ChainApproxMethod.ChainApproxSimple);

double totalArea = 0.0;

List<Point> centers = new List<Point>();

List<Size> wheelSizes = new List<Size>();

for (int i = 0; i < contours.Size; i++)

{

Rectangle boundingRect = CvInvoke.BoundingRectangle(contours[i]);

if (boundingRect.Width > 30 && boundingRect.Height > 30)

{

CvInvoke.Rectangle(img, boundingRect, new MCvScalar(0, 255, 0), 2);

totalArea += boundingRect.Width * boundingRect.Height;

Point center = new Point(boundingRect.X + boundingRect.Width / 2, boundingRect.Y + boundingRect.Height / 2);

centers.Add(center);

wheelSizes.Add(new Size(boundingRect.Width, boundingRect.Height));
```

```
}

}

double loadPerWheel = 10.0;

double totalLoadInTons = totalArea / loadPerWheel;

int numberOfWheels = wheelSizes.Count;

Size averageWheelSize = CalculateAverageWheelSize(wheelSizes);

double truckWeight = CalculateTruckWeight(totalLoadInTons, numberOf

Wheels);

pictureBox.Image = img.ToBitmap();

lblResult.Text = $"Общая нагрузка на оси: {totalLoadInTons} кг\nОценочный вес грузовика:
{truckWeight} т\nКоличество колес: {numberOfWheels}\nРазмер среднего колеса:
{averageWheelSize.Width}x{averageWheelSize.Height}";

}

private Size CalculateAverageWheelSize(List<Size> wheelSizes)

{

int totalWidth = 0;

int totalHeight = 0;

foreach (Size size in wheelSizes)

{

totalWidth += size.Width;

totalHeight += size.Height;

}

int averageWidth = totalWidth / wheelSizes.Count;

int averageHeight = totalHeight / wheelSizes.Count;

return new Size(averageWidth, averageHeight);

}

private double CalculateTruckWeight(double axleLoad, int numberOfWheels)

{

double wheelWeight = 500; // Пример веса одного колеса в кг
```

```
double truckWeight = wheelWeight * numberOfWheels / 1000;
return truckWeight;
}
private Image<TColor, TDepth> BitmapToImage<TColor, TDepth>(Bitmap bitmap)
where TColor : struct, IColor
where TDepth : new()
{
Image<TColor, TDepth> image = new Image<TColor, TDepth>(bitmap.Width, bitmap.Height);
BitmapData bitmapData = bitmap.LockBits(new Rectangle(0, 0, bitmap.Width, bitmap.Height),
ImageLockMode.ReadOnly, bitmap.PixelFormat);
CvInvoke.cvSetData(image, bitmapData.Scan0, bitmapData.Stride);
bitmap.UnlockBits(bitmapData);
return image;
}
}
}
namespace TruckDimensions
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
using (OpenFileDialog openFileDialog = new OpenFileDialog())
{
```

```
openFileDialog.Filter = "Image files (*.jpg, *.jpeg, *.png) | *.jpg; *.jpeg; *.png";
if (openFileDialog.ShowDialog() == DialogResult.OK)
{
string filePath = openFileDialog.FileName;
ProcessImage(filePath);
}
}
}
private void ProcessImage(string filePath)
{
Mat img = CvInvoke.Imread(filePath, ImreadModes.Color);
pictureBox1.Image = img.ToBitmap();
CvInvoke.CvtColor(img, img, ColorConversion.Bgr2Gray);
CvInvoke.GaussianBlur(img, img, new Size(5, 5), 1.5);
Mat edges = new Mat();
CvInvoke.Canny(img, edges, 100, 200);
VectorOfVectorOfPoint contours = new VectorOfVectorOfPoint();
CvInvoke.FindContours(edges, contours, null, RetrType.List,
ChainApproxMethod.ChainApproxSimple);
Rectangle truckRect = Rectangle.Empty;
int truckWidth = 0;
int truckHeight = 0;
int truckLength = 0;
int truckLe = 0;
for (int i = 0; i < contours.Size; i++)
{
Rectangle rect = CvInvoke.BoundingRectangle(contours[i]);
if (rect.Width > truckWidth)
```

```
{
truckWidth = rect.Width;
}
if (rect.Height > truckHeight)
{
truckHeight = rect.Height;
}
int length = Math.Max(rect.Width, rect.Height);
if (length > truckLength)
{
truckLength = length * Width / 2;
truckLe = truckLength;
}
}
double scaleFactor = 0.01; // 1 пиксель = 1 см
double realTruckWidth = truckWidth * scaleFactor;
double realTruckHeight = truckHeight * scaleFactor * 2;
double realTruckLength = truckLe * scaleFactor;
textBoxWidth.Text = $"{realTruckWidth:F2} м";
textBoxHeight.Text = $"{realTruckHeight:F2} м";
textBoxLength.Text = $"{realTruckLength:F2} м";
}
}
}
```

(Form1.Designer.cs)

```
namespace wes
{
partial class Form1
```

```
{
private System.ComponentModel.IContainer components = null;
private System.Windows.Forms.PictureBox pictureBox;
private System.Windows.Forms.Button btnLoadImage;
private System.Windows.Forms.Label lblResult;
private System.Windows.Forms.TextBox textBoxWidth;
private System.Windows.Forms.TextBox textBoxHeight;
private System.Windows.Forms.TextBox textBoxLength;
protected override void Dispose(bool disposing)
{
if (disposing && (components != null))
{
components.Dispose();
}
base.Dispose(disposing);
}
private void InitializeComponent()
{
this.pictureBox = new System.Windows.Forms.PictureBox();
this.btnLoadImage = new System.Windows.Forms.Button();
this.lblResult = new System.Windows.Forms.Label();
this.textBoxWidth = new System.Windows.Forms.TextBox();
this.textBoxHeight = new System.Windows.Forms.TextBox();
this.textBoxLength = new System.Windows.Forms.TextBox();
((System.ComponentModel.ISupportInitialize)(this.pictureBox)).BeginInit();
this.SuspendLayout();
this.pictureBox.Location = new System.Drawing.Point(12, 12);
this.pictureBox.Name = "pictureBox";
```

```
this.pictureBox.Size = new System.Drawing.Size(776, 426);
this.pictureBox.TabIndex = 0;
this.pictureBox.TabStop = false;
this.btnLoadImage.Location = new System.Drawing.Point(12, 444);
this.btnLoadImage.Name = "btnLoadImage";
this.btnLoadImage.Size = new System.Drawing.Size(126, 23);
this.btnLoadImage.TabIndex = 1;
this.btnLoadImage.Text = "Загрузить изображение";
this.btnLoadImage.UseVisualStyleBackColor = true;
this.btnLoadImage.Click += new System.EventHandler(this.btnLoadImage_Click);
this.lblResult.AutoSize = true;
this.lblResult.Location = new System.Drawing.Point(12, 470);
this.lblResult.Name = "lblResult";
this.lblResult.Size = new System.Drawing.Size(126, 13);
this.lblResult.TabIndex = 2;
this.lblResult.Text = "Загрузите изображение для начала.";
this.textBoxWidth.Location = new System.Drawing.Point(156, 446);
this.textBoxWidth.Name = "textBoxWidth";
this.textBoxWidth.Size = new System.Drawing.Size(100, 20);
this.textBoxWidth.TabIndex = 3;
this.textBoxWidth.Text = "Ширина";
this.textBoxHeight.Location = new System.Drawing.Point(262, 446);
this.textBoxHeight.Name = "textBoxHeight";
this.textBoxHeight.Size = new System.Drawing.Size(100, 20);
this.textBoxHeight.TabIndex = 4;
this.textBoxHeight.Text = "Высота";
this.textBoxLength.Location = new System.Drawing.Point(368, 446);
this.textBoxLength.Name = "textBoxLength";
```

```
this.textBoxLength.Size = new System.Drawing.Size(100, 20);
this.textBoxLength.TabIndex = 5;
this.textBoxLength.Text = "Длина";
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(800, 500);
this.Controls.Add(this.textBoxLength);
this.Controls.Add(this.textBoxHeight);
this.Controls.Add(this.textBoxWidth);
this.Controls.Add(this.lblResult);
this.Controls.Add(this.btnLoadImage);
this.Controls.Add(this.pictureBox);
this.Name = "Form1";
this.Text = "Truck Dimensions and Load Estimation";
((System.ComponentModel.ISupportInitialize)(this.pictureBox)).EndInit();
this.ResumeLayout(false);
this.PerformLayout();
}
}
}
```