минобрнауки россии

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Информационных технологий |
| наименование института (факультета) |
| Математическое и программное обеспечение ЭВМ |
| наименование кафедры |

ОТЧЕТ

по преддипломной практике

Листов 65

Студента Богданова Александра Павловича группы 1ПИб-01-41оп

Место прохождения практики

АО «Северсталь - Менеджмент»

Студент \_\_\_\_\_\_

(подпись)

Руководитель практики

от кафедры МПО ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность) (подпись, Ф.И.О.)

2022 год

Содержание

[Введение 4](#_Toc103854795)

[Раздел 1. Общая характеристика базы практики и организации информационно-производственной деятельности на предприятии 5](#_Toc103854796)

[Раздел 2. Систематизация и обобщение научной и производственно-технологической информации по теме исследования 7](#_Toc103854797)

[Раздел 3. Сравнительный анализ возможных вариантов реализации научно-технической информации по теме исследования 17](#_Toc103854798)

[Раздел 4. Реализация некоторых из возможных путей решения поставленной в техническом задании на ВКР задачи 19](#_Toc103854799)

[4.1 Проектирование классов пакета «Database Sending» 20](#_Toc103854800)

[4.1.1 Исходная диаграмма классов 20](#_Toc103854801)

[4.1.2 Уточнённая диаграмма классов 20](#_Toc103854802)

[4.1.3 Детальная диаграмма классов 21](#_Toc103854803)

[4.2 Выбор стратегии тестирования, разработка тестов, программа и методика испытаний 23](#_Toc103854804)

[4.2.1 Объект и цель испытаний 23](#_Toc103854805)

[4.2.2 Требования к информационному, аппаратно-программному обеспечению 24](#_Toc103854806)

[4.2.2.1 Требования к функциональным характеристикам 24](#_Toc103854807)

[4.2.2.2 Требования к надежности 24](#_Toc103854808)

[4.2.2.3 Требования к составу и параметрам технических средств 25](#_Toc103854809)

[4.2.2.4 Требования к программной документации 25](#_Toc103854810)

[4.2.3 Состав, порядок и методы испытаний 26](#_Toc103854811)

[4.2.4 Результаты проведения испытаний 27](#_Toc103854812)

[Раздел 5. Технико-экономическое обоснование выполняемой разработки 34](#_Toc103854813)

[5.1. Расчет себестоимости продукта 34](#_Toc103854814)

[5.2. Расчет цены программного продукта 39](#_Toc103854815)

[5.3. Расчет экономической эффективности 39](#_Toc103854816)

[Заключение 44](#_Toc103854817)

[Список литературы 45](#_Toc103854818)

[Приложение 1. Техническое задание 45](#_Toc103854819)

[Приложение 2. Текст программы 54](#_Toc103854820)

[Приложение 3. Спецификации 61](#_Toc103854821)

## Введение

Преддипломная практика проходила в дистанционном режиме на предприятии АО «Северсталь менеджмент». Сроки проведения практики: 18.04.2022- 15.05.2022.

Цель преддипломной практики: приобретение опыта профессиональной эксплуатации современного оборудования и программного обеспечения для производственной деятельности, сбор, анализ и обработка информации для выпускной квалификационной работы.

Задачи преддипломной практики соотносятся со следующими видами и задачами профессиональной деятельности бакалавра:

* предпроектное обследование объекта проектирования, системный анализ предметной области, их взаимосвязей;
* техническое проектирование;
* рабочее проектирование;
* выбор исходных данных для проектирования;
* расчет экономической эффективности;
* разработка средств реализации информационных технологий (методические, информационные, математические, алгоритмические, технические и программные);
* сбор, анализ научно-технической информации, отечественного и зарубежного опыта по тематике исследования.

В процессе практики бакалавры изучают методы и способы обработки информации, овладевают методикой исследования актуальной научной проблемы, приобретают опыт решения задач профессиональной деятельности на предприятии.

Преддипломная практика базируется на освоении следующих модулей: «Технология разработки программного обеспечения», «Информационные и компьютерные сети», «Проектирование систем управления данными», «Инструментальные средства программирования», «Модуль: Профильный», а также учебной и производственной практики.

Данная практика закладывает основу для успешного выполнения выпускной квалификационной работы.

Для успешного прохождения преддипломной практики необходимы следующие умения:

* применять ЭВМ для решения профессиональных задач;
* применять основные методы и инструменты разработки программного обеспечения;
* выполнять расчет экономической эффективности;
* формализовать предметную область программного проекта и разрабатывать спецификации для компонентов программного продукта.
* Навыки:
* работы в коллективе;
* использования различных технологий разработки программного обеспечения;
* владения стандартами и моделями жизненного цикла.

## Раздел 1. Общая характеристика базы практики и организации информационно-производственной деятельности на предприятии

ПАО «Северсталь» – одна из крупнейших в мире вертикально- интегрированных сталелитейных и горно-металлургических компаний с активами в России, США, Латвии, Украине, Польше, Италии, Либерии, а также инвестициями в других регионах. На предприятиях Компании работает около 61 тысячи человек [11].

«Северсталь» поставляет свою продукцию ведущим российским и мировым компаниям топливно-энергетического сектора, автомобилестроения, машиностроения, строительства. Ценные бумаги Компании котируются на Московской бирже (МБ), а глобальные депозитарные расписки представлены на Лон донской фондовой бирже (LSE). В структуру Компании входят заводы по производству стали и продуктов из стали в России («Северсталь Российская Сталь») и за рубежом («Северсталь Интернэшнл»), а также предприятия по добыче и обогащению железорудного сырья и коксующегося угля («Северсталь Ресурс») [11] .

Ключевые конкурентные преимущества Компании – вертикальная интеграция, полная самообеспеченность железной рудой и коксующимся углем и эффективное управление издержками. «Северсталь» – один из немногих производителей стали в мире, обеспеченных и железной рудой, и коксующимся углем [11].

Компания полностью покрывает свои потребности в сырье для производства стали в России, а также продает сырье на рынках в России и за рубежом. Вертикальная интеграция позволяет Компании снижать издержки, обеспечивать конкурентоспособную себестоимость производства, укрепляет устойчивость Компании к колебаниям рыночной конъюнктуры.

АО «Северсталь-Менеджмент» — единоличный исполнительный орган ПАО «Северсталь» [11].

Одним из подразделений «Северсталь-Менеджмент» является центр развития бизнес-системы «Северстали» (БСС). БСС представляет собой совокупность проектов и инициатив, направленных на обеспечение долгосрочных конкурентных преимуществ за счет безупречной производственной деятельности, организации труда и клиентоориентированности. Бизнес-система «Северстали» стала результатом более чем десятилетней работы по оптимизации производства и стандартизации внутренних процессов, выразившейся во множестве проектов, направленных на достижение максимальной эффективности труда и производства [11].

Данные меры наряду с проектами по изменению корпоративной культуры были объединены в систему проектов под названием Бизнес-система «Северстали».

Бизнес-система охватывает пять основных направлений развития:

* постоянное совершенствование;
* безопасность труда;
* клиентоориентированность;
* люди «Северстали»;
* бизнес-стандарт.

Развитие БСС это реализация кросс-функциональных проектов и проектов по расшивке «узких мест», изменение системы целеполагания, постоянные улучшения на местах.

Приоритетными направлениями в настоящий момент являются проекты с ключевыми и перспективными клиентами, снижение издержек, управление надежностью оборудования, потоковые проекты, энергоэффективность.

## Раздел 2. Систематизация и обобщение научной и производственно-технологической информации по теме исследования

Отслеживание действий сотрудников на предприятии «Северсталь», работающих в потенциально опасных зонах – крайне сложный и ресурсоёмкий процесс. Отсутствие надлежащего контроля может привести к возникновению чрезвычайных ситуаций, ставящих под угрозу жизнь и здоровье рабочих. Поэтому проблема контроля безопасности на предприятиях на сегодняшний день остаётся актуальной. В особенности это касается сотрудников, взаимодействующих с заводскими станками и другими видами промышленной техники.

Для контроля безопасности на промышленных предприятиях компания-заказчик использует видеокамеры, способные при регистрации нарушения прервать работу агрегата или подать соответствующий сигнал. Однако, данное решение применимо только к некоторому небольшому числу агрегатов. Следовательно, агрегаты, за которыми не ведётся подобное наблюдение, обладают гораздо меньшим уровнем контроля безопасности

Жизненный цикл (ЖЦ) – базовое понятие методологии проектирования информационной системы. Непрерывный процесс, который начинается с момента решения о необходимости создания программного обеспечения (ПО), и заканчивается после полного изъятия из эксплуатации [1].

Моделью жизненного цикла для разрабатываемой программного обеспечения, выбрана V-модель (разработка через тестирование).

В этой модели тестирование продукта обсуждается, проектируется и планируется на ранних этапах ЖЦ разработки. План испытания приемки заказчиком разрабатывается на этапе планирования, а компоновочного испытания системы — на фазах анализа, разработки проекта и т.д. Этот метод позволяет выявить огрехи на самых ранних стадиях развития проекта и минимизировать ошибки в дальнейшем [2].

Схема этой модели представляет собой букву V, из-за которой она, собственно, и получила своё название: левая часть отражает процесс дизайна программного обеспечения, а правая — процесс разработки и прямого тестирования (рис. 1).

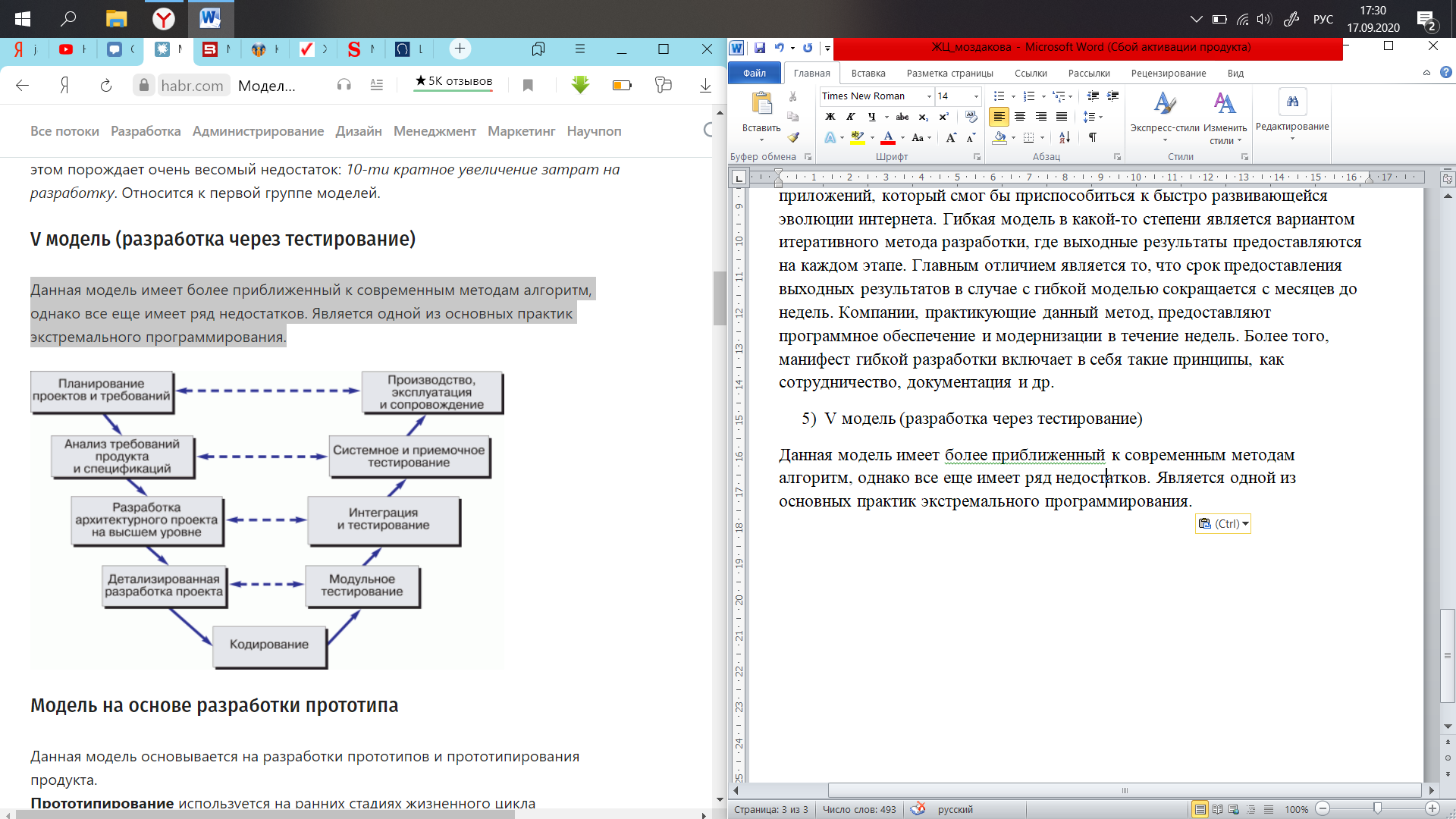


Рис.1. V-модель жизненного цикла

Эта модель жизненного цикла выбрана, так для данной системы доступна и известна информация о методе реализации решения и технологи, т.е уже имеется инструмент, который надо реализовать, его следует автоматизировать и улучшить, следовательно, вся информация и работа заранее известна. Так же все требования к системе тоже уже имеются и можно начинать выполнение.

В данной работе для проектирования системы был выбран следующий стек технологий:

* для обнаружения человека на изображении используется связка библиотеки компьютерного зрения EmguCV и нейронной сети COCO;
* в качестве языка программирования использовался язык C# и платформа .Net;
* пользовательский интерфейс реализован на системе WPF;
* Microsoft Visual Studio 2019 в качестве среды разработки.

Для обработки видеопотока и его вывода на экран использовалась библиотека компьютерного зрения EmguCV. EmguCV — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков [4].

Для реализации программного обеспечения использовался язык C#, который работает на платформе .Net. C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270 [5].

Пользовательский интерфейс реализован на платформе WPF. Windows Presentation Foundation (WPF) — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML [6].

В качестве среды разработки ПО использовалась среда Microsoft Visual Studio. Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight [6].

В разработке программного обеспечения (ПО) будет использоваться паттерн MVVM.

Паттерн MVVM (Model-View-ViewModel) позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения [10].

Данный паттерн был представлен Джоном Госсманом в 2005 году как модификация шаблона Presentation Model и был первоначально нацелен на разработку приложений в WPF.

MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

Модель описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления [10].

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление — это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

Хотя окно (класс Window) в WPF может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

Однако иногда в файле связанного кода все может находиться некоторая логика, которую трудно реализовать в рамках паттерна MVVM во ViewModel.

Представление не обрабатывает события за редким исключением, а выполняет действия в основном посредством команд.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны [10]. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели [10].

Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

Для реализации алгоритма детекции частей тела и интерфейса программы была выбрана среда разработки Microsoft Visual Studio 2019. Языком разработки был выбран С#. По рекомендации заказчика, для работы с нейронными сетями была выбрана библиотека EmguCV. Для использования нейронной сети использовался фреймворк DNN Caffe.

В нашем техническом решении детекции частей тела используется подход «глубокого обучения», который позволяет классифицировать поданное на вход изображение (или сигнал) в соответствии с предварительной настройкой (обучением) нейронной сети [9].

В большинстве технических решений, основанных на глубоком обучении, используются свёрточные нейронные сети (CNN), наиболее известные своей способностью распознавать паттерны, присутствующие на изображениях. На сегодняшний день свёрточные нейронные сети достигли точности, превосходящей человеческий уровень. CNN используют фильтры, чтобы определять, какие особенности, такие как края, присутствуют на всем изображении. Фильтр — это просто матрица значений, называемых весами, которые обучены обнаруживать определенные особенности. Фильтр перемещается по каждой части изображения, чтобы проверить, присутствует ли признак, который он должен обнаруживать. Чтобы предоставить значение, показывающее, насколько достоверно наличие определенного признака, фильтр выполняет операцию свертки, которая представляет собой поэлементное произведение и сумму двух матриц. Если признак присутствует в части изображения, операция свертки между фильтром и этой частью изображения приводит к получению действительного числа с высоким значением. Если признак отсутствует, результирующее значение будет низким [9]. Пример операции свёртки представлен на рис. 2.

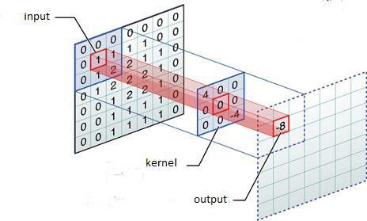


Рис. 2. Операция свёртки

Нейронная сеть OpenPose определяет части тела человека и их расположение в пространстве на видеозаписях и статичных изображениях. Сеть определяет положение туловища, рук, ног и других частей тела через двумерные координаты, не связывая, к какому именно человеку они относятся, а затем присваивает части отдельным людям. Также OpenPose обладает функционалом для распознания ключевых точек лица и рук. По словам разработчиков, OpenPose устойчив к перекрытию частей тела, в том числе при взаимодействии человека с объектом [8]. Пример подобной ситуации представлен на рис. 3.

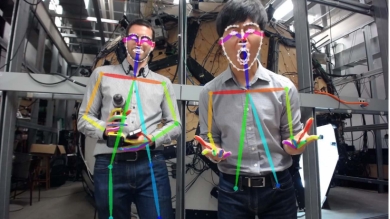


Рис. 3. Пример работы OpenPose

Схема работы OpenPose представлена на рис. 4.

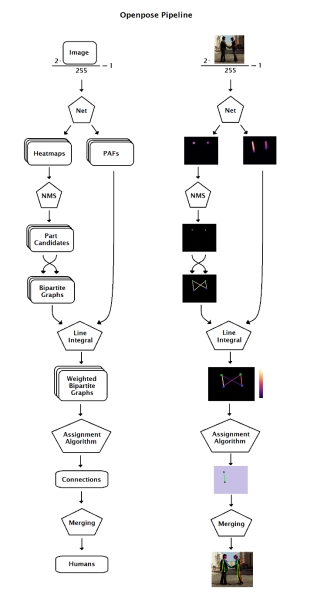


Рис. 4. Схема работы OpenPose

Далее описан алгоритм обнаружения частей тела людей при помощи OpenPose, который применяется в данном решении.

Граф, отмечающий части тела каждого человека, состоит из частей и пар. Часть тела – найденная нейронной сетью точка. Пара – соединение двух точек линией для образования части «скелета» [9].

Тепловая карта представляет собой матрицу, которая показывает уверенность сети в том, что определенный пиксель содержит определенную часть. Есть 18 (+1) тепловых карт, связанных с каждой из частей тела и проиндексированных. из этих 18 матриц извлекается расположение частей тела.

Поля сходства частей (Part Affinity Fields) — это матрицы, которые дают информацию о положении и ориентации пар. Они существуют парами: для каждой части есть PAF в направлении «x» и PAF в направлении «y». Существует 38 индексированных PAF. Объединение частей в пары происходит благодаря этим 38 матрицам [8].

После создания OpenPose тепловых карт и полей сходства частей происходит извлечение расположения деталей из тепловой карты при помощи алгоритма не максимального подавления (NMS):

* Выбирается первый пиксель тепловой карты;
* Пиксель окружается окном со стороной 5. В этой области находится максимальное значение;
* Значение центрального пикселя области подставляется вместо максимума;
* Окно сдвигается на один пиксель, шаги 1-4 повторяются до полного охвата всей тепловой карты;
* Результат сравнивается с исходной тепловой картой. Пиксели с одинаковым значением являются искомыми пиками. Все остальные пиксели подавляются, получая значение 0;
* После всего процесса ненулевые пиксели обозначают местоположение кандидатов в части тела;
* После нахождения кандидатов для каждой из частей тела, их нужно соединить в пары. Для каждой возможной пары создаётся полный двудольный граф, вершины которого – все возможные кандидаты пары, а рёбра – все возможные соединения [8].

Для нахождения нужных связей в полученном графе необходимо решить «проблему присваивания». Для этого каждому ребру графа нужно присвоить вес при помощи линейного интеграла, представленного на рис. 5.

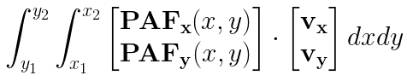


Рис. 5. Линейный интеграл оценки соединений

Данный линейный интеграл даёт каждому соединению оценку, исходя из соответствующих этому соединению полей сходства частей PAF. Это позволяет решить задачу присваивания.

Решение задачи присваивания:

* Отсортировать каждое возможное соединение по его баллу;
* Связь с наивысшим баллом действительно является последней связью;
* Перейти к следующему возможному подключению. Если никакие части этого соединения не были назначены окончательному соединению ранее, это окончательное соединение;
* Шаг 3 повторяется до нахождения всех связей.

Пример решения задачи присваивания представлен на рис. 6.

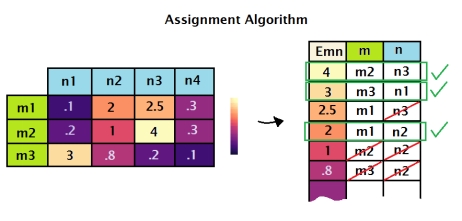


Рис. 6. Пример решения задачи присваивания

Последний шаг – преобразование найденных связей в «скелеты» людей на изображении. Изначально создаётся множество людей H, где каждый человек (набор связей) состоит из одной связи, а число людей равно числу связей. Затем, все люди попарно проверяются, и, если люди H1 и H2 имеют общий индекс части тела с одинаковыми координатами, это значит, что они являются одним и тем же человеком. Содержимое H2 добавляется в H1, а H2 удаляется из множества. Это происходит до тех пор, пока в множестве не останется людей, использующих одну и ту же часть тела в связях [8].

На выходе каждый человек обозначается, как набор частей, где каждая часть содержит свой индекс, свои относительные координаты и свою оценку.

## Раздел 3. Сравнительный анализ возможных вариантов реализации научно-технической информации по теме исследования

Предлагаемое решение заключается в создании программного обеспечения, которое в режиме реального времени анализировало бы видеопоток, поступающий с видеокамеры и в случае выявления нарушения останавливало бы работу агрегата, за которым произошло нарушение (если возможно), подавало бы предупредительный сигнал, а также формировало отчёт.

В ходе анализа были найдены следующие аналоги:

* на выставке «Безопасность и Охрана труда — 2018», компания КРОК представила работу передовых IT-систем для проведения предрейсовых и предсменных осмотров, контроля ношения работниками средств индивидуальной защиты, отслеживания физического состояния и локального позиционирования на объектах, отображения событий на 3D-модели здания и оценки поведения водителей в режиме реального времени;
* внутри компании «Северсталь» в цехе выплавки запущена в работу модель, которая фиксирует нахождение человека в подконвертерной зоне во время продувки. В случае фиксации нарушения модель автоматически отправляет снимок по электронной почте начальнику цеха и мастеру, а также включается сирена в подконвертерной зоне.
* внутри компании «Северсталь» в цехе выплавки запущена в работу модель, которая фиксирует нахождение человека в опасной зоне на машинах подачи кислорода. Система анализирует изображение с видеокамер и в режиме онлайн при помощи специальных алгоритмов определяет, находится ли работник в опасной зоне в то время, когда конвертер не пустой. В случае фиксации нарушения модель автоматически отправляет снимок по электронной почте начальнику цеха и мастеру;
* The industrial machine vision представляет собой комплексную интеграцию оптического, электронного, сенсорного и программного обеспечения в производственный процесс. Основная цель данной системы заключается в обеспечении безопасности, и проверке качества промышленного продукта.
* Система контроля персонала «Амальгама». «Амальгама» записывает изображение с экрана монитора как видео, реагируя на события и архивируя видеоданные в стандартном видеоформате.

Основные возможности:

1. запись монитора синхронизирована с записью обычных камер и микрофонов, а также других устройств;
2. данные со многих компьютеров могут писаться по сети на отдельный сервер;
3. возможна работа через Интернет.

Кроме контроля за деятельностью персонала, Амальгама позволяет:

1. хранить в памяти все ваши действия за компьютером;
2. совместно с видеоаналитикой выявлять несанкционированные операции;
3. следить за детьми;
4. контролировать и управлять стационарными компьютерами дистанционно;
5. записывать ход научных экспериментов;
6. осуществлять в Интернете демонстрационные on-line трансляции;
7. проводить для клиентов обучение и объяснения по работе с программами.

Проведя анализ аналогов, можно сделать вывод, что не один из программных продуктов не выполняет выдвигаемые заказчиком функции (смотреть в прил. 1). Данные системы отвечают за контроль качества работы персоналом в офисных помещениях, что не соответствует специфике требуемого ПО или же не обладает обширным функционалом, обнаруживая части тела. В результате данного вывода было принято решение разработки собственной системы.

## Раздел 4. Реализация некоторых из возможных путей решения поставленной в техническом задании на ВКР задачи

В табл.1 представлены задачи, выполненные в ходе прохождения преддипломной практики.

Табл.1

Задачи

|  |  |  |
| --- | --- | --- |
| Номер | Описание задачи | Ссылка |
| 1 | Выполнение расчётов в разделе «Технико-экономическое обоснование разработки» расчетно-пояснительной записки | Раздел 5 |
| 2 | Проектирование пакета «Database Sending» | Пункт 4.1 |
| 3 | Тестирование приложения и оформление раздела «Выбор стратегии тестирования, разработка тестов, программа и методика испытаний» расчетно-пояснительной записки | Пункт 4.2 |
| 4 | Написание спецификаций | Приложение 3 |

## 4.1 Проектирование классов пакета «Database Sending»

## 4.1.1 Исходная диаграмма классов

Исходная диаграмма классов пакета «Database Sending» представлена на рис. 7, а её описание в табл. 2.

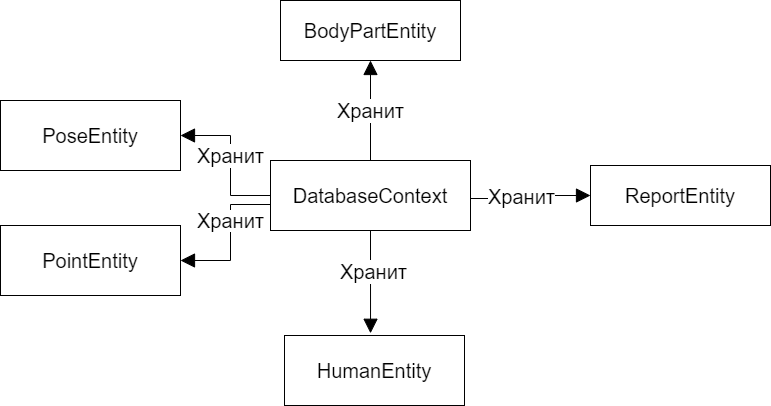


Рис. 7. Исходная диаграмма классов пакета «Database Sending»

Таблица 2

Описание классов пакета «Database Sending»

|  |  |
| --- | --- |
| Класс | Описание |
| DatabaseContext | класс, описывающий схему базы данных |
| PoseEntity | класс, описывающий таблицу поз |
| PointEntity | класс, описывающий таблицу точек |
| HumanEntity | класс, описывающий таблицу людей |
| ReportEntity | класс, описывающий таблицу отчётов |
| BodyPartEntity | класс, описывающий таблицу частей тела |

## 4.1.2 Уточнённая диаграмма классов

Уточнённая диаграмма классов пакета «Database Sending» представлена на рис. 8.

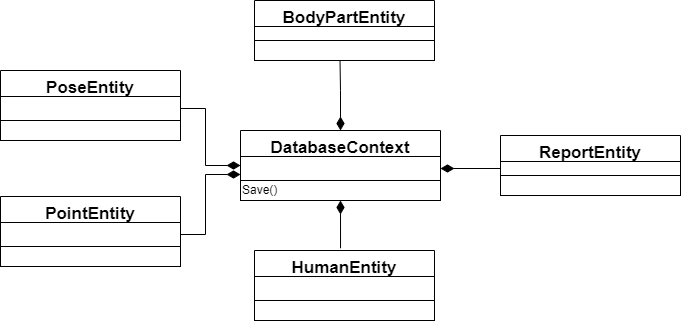


Рис. 8. Уточнённая диаграмма классов пакета «Database Sending»

## 4.1.3 Детальная диаграмма классов

Детальная диаграмма классов пакета «Database Sending» представлена на рис. 9. Описание полей и методов классов пакета «Database Sending» (табл. 3-9).

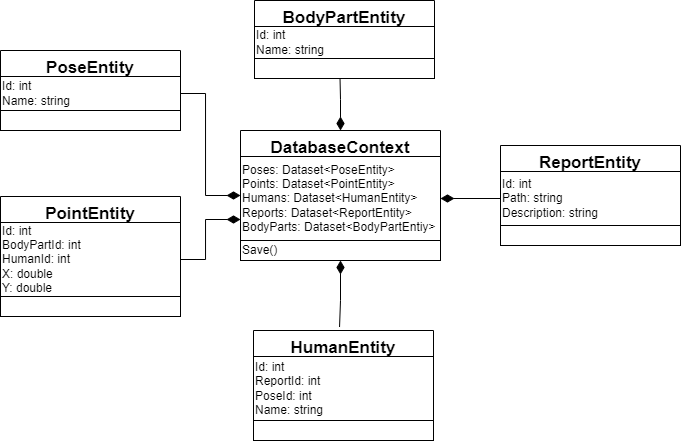


Рис. 9. Детальная диаграмма классов пакета «Database Sending»

Таблица 3

Описание полей класса «DatabaseContext»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Poses | Dataset<PoseEntity> | Репозиторий поз |
| Points | Dataset<PointEntity> | Репозиторий точек |
| Humans | Dataset<HumanEntity> | Репозиторий человек |
| Reports | Dataset<ReportEntity> | Репозиторий отчётов |
| BodyParts | Dataset<BodyPartEntiy> | Репозиторий частей тела |

Таблица 4

Описание методов класса «DatabaseContext»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Возвращаемое значение | Описание |
| Save | **-** | **-** | Метод для сохранения данных в БД |

Таблица 5

Описание полей класса «PoseEntity»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id | int | Идентификатор позы |
| Name | string | Имя позы |

Таблица 6

Описание полей класса «PointEntity»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id | int | Идентификатор точки |
| BodyPartId | int | Идентифкатор части тела |
| HumanId | int | Идентификатор человека |
| X | double | Координата по оси абсцисс |
| Y | double | Координата по оси ординат |

Таблица 7

Описание полей класса «HumanEntity»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id | int | Идентифкатор человека |
| ReportId | int | Идентификатор отчёта |
| PoseId | int | Идентифактор позы |
| Name | string | Отображаемое имя |

Таблица 8

Описание полей класса «ReportEntity»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id | int | Идентифкатор отчёта |
| Path | string | Путь для кадра |
| Description | string | Описание отчёта |

Таблица 9

Описание полей класса «BodyPartEntity»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id | int | Идентификатор части тела |
| Name | string | Имя части тела |

## 4.2 Выбор стратегии тестирования, разработка тестов, программа и методика испытаний

В данном разделе описано и представлено тестирование разработанного программного обеспечения.

## 4.2.1 Объект и цель испытаний

Объектом испытаний является система скелетизации изображений человека для контроля опасных действий.

Цели испытаний:

* нахождение ошибок в программе;
* проверка правильности работы отдельных функций программы;
* проверка соответствия, разрабатываемого ПО требованиям, заявленным в прил. 1.

## 4.2.2 Требования к информационному, аппаратно-программному обеспечению

## 4.2.2.1 Требования к функциональным характеристикам

Программное обеспечение представляет собой десктопное приложение, реализующее следующий функционал:

* отслеживание положение работников;
* отслеживание отдельные части тела работников;
* определение позу работников;
* наличие предусмотренной возможности добавления нескольких зон опасности;
* наличие настройки у каждой зоны обнаружения конкретных частей тела;
* наличие инструмента комбинирования зон для формирования сложной опасной ситуации.

## 4.2.2.2 Требования к надежности

К разрабатываемому программному обеспечению предъявляются следующие требования к надежности:

* система должна иметь защиту от некорректных действий оператора и ошибочных исходных данных;
* система не должна во время работы модифицировать свой код или коды других программ;
* при удалении и передаче зафиксированных данных из базы данных система должна запрашивать подтверждения действия;
* логгирование;
* корректный вывод данных на экран, видео должно иметь качество не менее HD;
* проверка вводимых в настройках пользователем данных на корректность;
* система должна обеспечивать контроль целостности структур баз данных, нарушение которой возможно после аппаратных сбоев.

## 4.2.2.3 Требования к составу и параметрам технических средств

Компьютер, на котором будет установлена система должен обладать следующими требованиями:

* оперативная память – не менее 8 Гб;
* видеокарта – не менее GTX 1660Ti с объемом видеопамяти 6 Гб;
* монитор, клавиатура, мышь;
* подключения к локальной вычислительной сети (ЛВС) через беспроводное подключение (например, Wi-Fi) или через проводное подключение (например, через Ethernet порт).
* процессор – не менее Intel i5 10400 с тактовой частой 3.29 Ггц.

## 4.2.2.4 Требования к программной документации

Программная документация должна включать в себя следующие элементы:

1. Расчетно-пояснительная записка.
2. Техническое задание.
3. Схемы и/или диаграммы.
4. Текст программы.
5. Спецификации.
6. Руководство пользователя.

Документация оформляется на листах формата А4 по действующим стандартам на создание документации к программному обеспечению (ЕСПД).

## 4.2.3 Состав, порядок и методы испытаний

Тестирование разработанного ПО проводилось согласно следующего состава мероприятий по тестированию:

* тестирование разработанных модулей;
* тестирование программы на соответствие функциональных требований, изложенных в прил. 1;
* тестирование надежности программы на соответствие требований к надежности, изложенных в прил. 1;
* проверка программной документации на соответствие требований ЕСПД.

Тестирование ПО проводилось на персональном компьютере, имеющем следующие характеристики:

* процессор Ryzen 7 2700 (8 ядер, 16 потоков, тактовая частота 4.2 ГГц);
* видеокарта Gigabyte GeForce GTX 1660Ti;
* объем оперативной памяти 16 Гб;
* твердотельный накопитель объемом 1024 Гб;
* операционная система Windows 10 Домашняя;
* монитор с разрешением 1920х1080.

Для тестирования программной документации использовался метод ручного контроля. Для тестирования программного обеспечения использовалось два метода: функциональное тестирование и системное тестирование.

Для системного тестирования применялся метод ручного контроля. Ручной контроль обычно используется на ранних этапах разработки, так как с его помощью можно находить от 30 до 70 % ошибок логического проектирования и кодирования. Исходными данными для таких проверок являются: техническое задание, спецификации, структурная и функциональная схемы программного продукта, схемы отдельных компонентов.

Из методов ручного контроля был выбран метод проверки за столом. Этот метод не требует наличия группы специалистов. Проверка исходного текста проводится одним человеком, который читает текст программы и проверяет его на наличие возможных ошибок по списку наиболее часто встречающихся ошибок.

Для тестирования модулей был выбран один из методов функционального тестирования – метод граничных значений. Исходными данными для этого метода являются: программные модули, спецификации на компоненты, правильные и неправильные данные для модулей.

## 4.2.4 Результаты проведения испытаний

В результате проверки программной документации методом ручного контроля были исправлены найденные грамматические и пунктуационные ошибки, было проверено соответствие документации требованиям ЕСПД. Также были проверены построенные схемы и диаграммы на соответствие ГОСТ.

Результаты функционального тестирования компонентов программного обеспечения представлены в табл. 10.

Таблица 10

Результаты тестирования компонентов программного обеспечения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата | Тестируемый модуль или подпрограмма | Кто проводил тестирование | Способ тестирования | Результат тестирования |
| 1 | 2 | 3 | 4 | 5 |
| 10.05.2022 | Query.cs  CheckPerson | Тестировщик | Функциональное | Неудача, люди, не подходящие под запрос, были помечены как нарушители |
| 10.05.2022 | Query.cs  CheckPerson | Тестировщик | Функциональное | Успех |
| 10.05.2022 | Query.cs  AddZone | Тестировщик | Функциональное | Успех |
| 10.05.2022 | Query.cs  RemoveZone | Тестировщик | Функциональное | Неудача, удалена зона не была удалена из запроса |
| 10.05.2022 | Query.cs  RemoveZone | Тестировщик | Функциональное | Успех |
| 10.05.2022 | Query.cs  CreateReport | Тестировщик | Функциональное | Неудача, в отчёте неправильно сформировалось описание ситуации |
| 10.05.2022 | Query.cs  CreateReport | Тестировщик | Функциональное | Успех |

Продолжение таблицы 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 10.05.2022 | Zone.cs  CheckPoint | Тестировщик | Функциональное | Неудача, зона обнаруживает все точки, находящиеся не внутри, а снаружи |
| 10.05.2022 | Zone.cs  CheckPoint | Тестировщик | Функциональное | Успех |
| 10.05.2022 | Zone.cs  AddPart | Тестировщик | Функциональное | Успех |
| 10.05.2022 | Zone.cs  RemovePart | Тестировщик | Функциональное | Успех |
| 10.05.2022 | Zone.cs  SetMaxHumanCount | Тестировщик | Функциональное | Неудача, удалось задать отрицательное количество |
| 11.05.2022 | Zone.cs  SetMaxHumanCount | Тестировщик | Функциональное | Успех |
| 11.05.2022 | Zone.cs  SetDetectionInside | Тестировщик | Функциональное | Неудача, изменение параметра работает инвертировано |
| 11.05.2022 | Zone.cs  SetDetectionInside | Тестировщик | Функциональное | Успех |
| 11.05.2022 | Report.cs  Send | Тестировщик | Функциональное | Неудача, не удалось отправить отчёт в Teams |
| 11.05.2022 | Zone.cs  Send | Тестировщик | Функциональное | Успех |

После функционального тестирования, было проведено тестирование всей системы в целом на выполнение требований ТЗ (см. прил. 1), т.е. было проведено системное тестирование.

Было проведено тестирование разработанного ПО на выполнение требований к функциональным характеристикам. Результаты данного тестирования представлены в табл. 11.

Таблица 11

Тестирование разработанного ПО на выполнение требований к функциональным характеристикам

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата | Тестируемое требование из ТЗ | Кто проводил тестирование | Способ тестирования | Результат |
| 1 | 2 | 3 | 4 | 5 |
| 11.05.2022 | отслеживание положение работников | Тестировщик | Системное тестирование | Успех |
| 11.05.2022 | отслеживание отдельных частей тела работников | Тестировщик | Системное тестирование | Неудача, некорректно преобразуются точки глаз и ушей |
| 12.05.2022 | отслеживание отдельных частей тела работников | Тестировщик | Системное тестирование | Успех |

Продолжение таблицы 11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 12.05.2022 | определять позу работников | Тестировщик | Системное тестирование | Неудача, некорректное определение лежачего положения |
| 12.05.2022 | определять позу работников | Тестировщик | Системное тестирование | Успех |
| 12.05.2022 | добавление  нескольких зон опасности | Тестировщик | Системное тестирование | Успех |
| 12.05.2022 | обнаружение конкретных частей тела | Тестировщик | Системное тестирование | Успех |
| 12.05.2022 | тестирование инструмента для комбинирования зон для формирования сложной опасной ситуации. | Тестировщик | Системное тестирование | Неудача, комбинирование зон работает только для обнаруженных внутри людей, а не снаружи |
| 12.05.2022 | тестирование инструмента для комбинирования зон для формирования сложной опасной ситуации. | Тестировщик | Системное тестирование | Успех |

Следующим было проведено тестирование разработанного ПО на выполнение требований к надежности. Результаты данного тестирования представлены в табл. 12.

Таблица 12

Тестирование разработанного ПО на выполнение требований к надежности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемое требование из ТЗ | Кто проводил тестирование | Способ тестирования | Результаты тестирования |
| 1 | 2 | 3 | 4 | 5 |
| 13.05.2022 | Система должна иметь защиту от некорректных действий оператора и ошибочных исходных данных | Тестировщик | Системное тестирование | Неудача, не был запрещен ввод букв в настройках времени нахождения в зоне |
| 13.05.2022 | Система должна иметь защиту от некорректных действий оператора и ошибочных исходных данных | Тестировщик | Системное тестирование | Успех |
| 13.05.2022 | Система не должна во время работы модифицировать свой код или коды других программ | Тестировщик | Системное тестирование | Успех |

Продолжение таблицы 12

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | | 4 | 5 |
| 13.05.2022 | При удалении и передаче зафиксированных данных из базы данных система должна запрашивать подтверждения действия | Тестировщик | | Системное тестирование | Успех |
| 13.05.2022 | Логгирование | Тестировщик | | Системное тестирование | Неудача, лог-файл сохранялся по неверному пути |
| 13.05.2022 | Логгирование | | Тестировщик | Системное тестирование | Успех |
| 13.05.2022 | Корректный вывод данных на экран, видео должно иметь качество не менее HD | | Тестировщик | Системное тестирование | Успех |
| 13.05.2022 | Система должна обеспечивать контроль целостности структур баз данных, нарушение которой возможно после аппаратных сбоев | | Тестировщик | Системное тестирование | Успех |

Так как тестирование ПО проводилось на персональном компьютере, имеющем следующие характеристики:

* процессор Ryzen 7 2700 (8 ядер, 16 потоков, тактовая частота 4.2 ГГц);
* видеокарта Gigabyte GeForce GTX 1660Ti;
* объем оперативной памяти 16 Гб;
* твердотельный накопитель объемом 512 Гб;
* операционная система Windows 10 Домашняя;
* монитор с разрешением 1920х1080.

данные характеристики соответствуют минимальным системным требованиям, предъявляемым к оборудованию.

В результате функционального тестирования компонентов ПО были выявлены их недостатки и ошибки в работе, которые были устранены в результате доработки и повторного тестирования. В ходе системного тестирования также были устранены недостатки в работе системы. В результате программное обеспечение выполняет все заявленные требования в полном объеме.

## Раздел 5. Технико-экономическое обоснование выполняемой разработки

## 5.1. Расчет себестоимости продукта

Состав разработчиков: программист-дипломник, руководитель ВКР.

Затраты на оплату труда при разработке программного продукта вычисляются по формуле:

|  |  |
| --- | --- |
| , - с окладом | (1) |
|  |  |

где  – общая зарплата работника за час;

Отч – отчисления с зарплаты, %;

 – время написания программы.

Заработная плата программиста за час определяется по следующей формуле:

|  |  |
| --- | --- |
|  | (2) |
|  |  |

где  – ставка программиста;

 – фонд рабочего времени в месяц, ч.

Заработная плата дополнительная определяется по следующей формуле:

|  |  |
| --- | --- |
|  | (3) |
|  |  |

где  – заработная плата программиста;

 – норма отчислений на дополнительную зарплату (10%).

Зарплата общая вычисляется по следующей формуле:

|  |  |
| --- | --- |
|  | (4) |

Отчисления на соцстрах, фонд занятости и пенсионный фонд вычисляются по следующей формуле:

|  |  |
| --- | --- |
|  | (5) |

где  – отчисления на соцстрах (0,5% от );

 – отчисления в фонд занятости (0,5% от );

 – отчисления в пенсионный фонд (2% от ).

Ставка программиста разработчика равна 29000. Ставка руководителя ВКР равна 31700р. По формуле (2) рассчитываются общие заработные платы за час программиста-дипломника и руководителя ВКР:

= 181,25 руб.

= 198,125 руб.

По формуле (3) рассчитываются дополнительные заработные платы:

= 0 руб.

= 0 руб.

Зарплаты общие вычисляются по формуле (4):

Отчисления на соцстрах, фонд занятости и пенсионный фонд вычисляются по формуле (5):

= 870 руб.

= 951 руб.

По формуле (1) рассчитываются затраты на оплату труда при разработке программного продукта:

134415 руб.

146929,5 руб.

Все данные по заработной плате сводятся в табл. 13.

Таблица 13

Данные по заработной плате

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Должность  разработчика | Разряд | Время работы,  мес. | Стпр,  руб. | Зпр,  руб. | Здоп,  руб. | Зобщ,  руб. | Отч,  руб. | Зтр, руб. |
| Программист-дипломник |  | 4,5 | 29000 | 181,25 | 0 | 29000 | 870 | 134415 |
| Руководитель ВКР |  | 4,5 | 31700 | 198,125 | 0 | 31700 | 951 | 146929,5 |

Затраты на использование машинного времени вычисляются по формуле:

|  |  |
| --- | --- |
|  | (6) |

где Зм.вр – затраты на использование машинного времени, руб.;

См.вр – стоимость одного часа машинного времени, руб./ч;

Врв.т – время использования вычислительной техники, ч.

Стоимость одного часа машинного времени рассчитывается по формуле:

где См.вр – стоимость одного часа машинного времени, руб./ч;

Цк – покупная цена компьютера, руб.;

Ссл.к – срок службы компьютера, год;

Кр.д – количество рабочих дней в году;

Врс – время работы компьютера в течение суток, ч;

Стэ – стоимость одного кВт\*ч электроэнергии, руб.;

Мвс – мощность вычислительной системы, кВт.

Время использования вычислительной техники рассчитывается по следующей формуле:

где Врв.т – время использования вычислительной техники, ч;

Кд.р – количество дней разработки ПО.

Затраты на носители информации принимаются в размере 2 % от цены вычислительной техники Зн.и.

Зн.и = 800 руб.

Затраты на текущий и профилактический ремонт принимаются в размере 4 % от цены вычислительной техники Зрем.

Зрем = 1600 руб.

Прочие эксплуатационные расходы включают в себя затраты на освещение, отопление, охрану, уборку и текущий ремонт помещений. Они принимаются в размере 10 % от стоимости помещения (или его аренды), где происходит разработка программного продукта Зпр.

Зпр = 600 руб.

Себестоимость программного продукта рассчитывается по формуле:

По формуле (7) вычисляется стоимость одного часа машинного времени:

Время использования вычислительной техники рассчитывается по формуле (8):

По формуле (6) вычисляются затраты на использование машинного времени:

Себестоимость разработки рассчитывается по формуле (9):

## 5.2. Расчет цены программного продукта

Для определения минимальной цены, ниже которой разработчику будет невыгодно продавать программный продукт, следующая формула:

где Цп.п – цена программного продукта, руб.;

Сп.п – себестоимость программного продукта, руб.;

Нпр – норматив прибыли (20 %, в формуле Нпр = 0,2).

## 5.3. Расчет экономической эффективности

Расходы потребителя, связанные с эксплуатацией программы, определяются по следующей формуле:

где Рэ.п – эксплуатационные расходы потребителя, руб.;

Врп.п – объем машинного времени в течение года, необходимый для решения данной задачи с использованием программы, ч;

См.вр – стоимость одного часа машинного времени, руб./ч;

Цп.п – цена программного продукта, руб.;

Ссл – срок службы программного продукта, год. Обычно составляет 1 – 2 года, затем выпускается новая версия программного продукта.

Капитальные затраты рассчитываются по формуле:

где Ркап – капитальные расходы потребителя, руб.;

Фвр – полезный годовой фонд времени работы вычислительной техники, принимается условно 2000 ч в год;

КЭВМ – капитальные затраты на вычислительную технику, для которой предназначена программа, руб.

Капитальные затраты на вычислительную технику рассчитываются по формуле:

где ЦЭВМ – цена вычислительной техники, руб.;

Рп.п – прочие расходы потребителя, связанные с помещением (отопление, освещение, уборка и т.д.), принимаются в размере 10 % от стоимости помещения потребителя (или его аренды), руб.

Для расчета годовой экономии эксплуатационных расходов потребителя вычисляются эксплуатационные затраты потребителя при решении задачи вручную:

|  |  |
| --- | --- |
|  | (15) |

где Рэ.руч – эксплуатационные расходы потребителя при решении задачи вручную, руб.;

ФЗП – фонд заработной платы персонала, обслуживающего решение задачи вручную, руб.; 12 – количество месяцев в году; 1,21 – поправочный коэффициент.

Тогда годовая экономия эксплуатационных расходов у одного потребителя рассчитывается по формуле:

|  |  |
| --- | --- |
| Э = Рэ.руч  – Рэ.п. | (16) |

Срок окупаемости программного продукта рассчитывается по формуле:

Годовой экономический эффект, получаемый одним пот­ре­бителем, рассчитывается по формуле:

|  |  |
| --- | --- |
| ЭЭ = Э – Ен · Ркап | (18) |

где Ен – нормативный коэффициент эффективности допол­ни­тельных капитальных вложений, равный 0,15.

Стоимость одного часа машинного времени рассчитывается по формуле (7):

= 4,019 руб. час.

Объем машинного времени в течении года, необходимый для решения данной задачи с использованием программы рассчитывается по формуле (8):

=8760 ч.

Расходы потребителя, связанные с эксплуатацией программы рассчитываются по формуле (12):

руб.

Капитальные затраты на вычислительную технику рассчитываются (14):

Капитальные затраты высчитываются по формуле (13):

684 228,12 руб.

Для расчета годовой экономии эксплуатационных расходов потребителя вычисляются эксплуатационные затраты потребителя при решении задачи вручную по формуле (15):

Годовая экономия эксплуатационных расходов у одного потребителя рассчитывается по формуле (16):

Э = 871200 –210880,5= 660 319,5 руб.

Срок окупаемости программного продукта рассчитывается по формуле (17):

= 1,036207 год.

Годовой экономический эффект, получаемый одним потребителем, рассчитывается по формуле (18):

ЭЭ = 660 319,5 – 0,15 · 684 228,12 = 557 685,282 руб.

Заключение

В процессе прохождения производственной практики мною были освоены следующие компетенции:

1. Способностью работать в коллективе, толерантно воспринимая социальные, этнические, конфессиональные и культурные различия;
2. Способность инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем;
3. Способность участвовать в настройке и наладке программно-аппаратных комплексов;
4. Способность разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования;
5. Способность обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности.

По прохождению производственной преддипломной практики были получены и усовершенствованы свои знания и навыки разработки приложения при помощи таких средств как Entity Framework, C# и EmguCV.

В ходе прохождения практики был разработан модуль, отвечающий за работу с базой данных, изучена и использована методика тестирования для соответствующего раздела ВКР, а также был произведен расчёт технико-экономического обоснования разработки.

Список литературы

1. Иванова, Г.С. Технология программирования: Учебник для вузов
2. Ершов Е.В., д-р техн. наук, проф.; Виноградова Л.Н. и др. Методика и организация самостоятельной работы студентов − Коллектив авторов, ФГБОУ ВПО «Череповецкий государственный университет», 2012. −208 с.
3. Буч, Г., Язык UML. Руководство пользователя / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. – М.: ДМК, 2015. – 432 с.
4. Кейлер Адриан, Брадски Гари. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, 2008. – 556 с.
5. Герберт Шилдт. C# 4.0: полное руководство C# 4.0 The Complete Reference. Издательство — «Вильямс», 2010. — С. 1056.
6. Мэтью Мак-Дональд. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов, 4-е издание. Издательство - «Вильямс», 2013. — 1024 с.
7. Барков И.А. Объектно-ориентированное программирование. Лань, 2019 г. 700 с.
8. Human pose estimation using OpenPose with TensorFlow (part 2) [Электронный ресурс]. URL: https://arvrjourney.com/human-pose-estimation-using-openpose-with-tensorflow-part-2-e78ab9104fc8 / (дата обращения: 7.05.2022).
9. Escontrela, A. Convolutional Neural Networks from the ground up, 2018 [Электронный ресурс]. URL: https://towardsdatascience.com/convolutional-neural-networks-from-theground-up-c67bb41454e1 (дата обращения: 7.05.2022).
10. Паттерн MVVM Определение паттерна MVVM [Электронный ресурс] URL: https://metanit.com/sharp/wpf/22.1.php (дата обращения: 8.05.2022)
11. Северсталь [Электронный ресурс]. URL: https://severstal.com/rus (дата обращения: 10.05.2022)

Приложение 1

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

наименование института (факультета)

Математического и программного обеспечения ЭВМ

наименование кафедры

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ

д.т.н., профессор Ершов Е.В.

« » 20 г.

Разработка программного обеспечения скелетизации изображений человека для контроля опасных действий

Техническое задание на выпускную квалификационную работу

Листов 8

Руководитель: Ершов Е.В.

Исполнитель: студент гр. 1ПИб-01-41оп

Богданов А.П.

Череповец, 2022 год

Введение

Производственные процессы в рабочей сфере являются крайне опасными, поэтому к обеспечению безопасности относятся всё серьёзнее. На компании “Северсталь” уже имеется множество различных способов и методик профилактики травматизма среди работников. Для того, чтобы обезопасить работу сотрудников используются видеокамеры, которые при регистрации нарушения прерывают работу агрегата или подают соответствующий сигнал. На данный момент подобные системы используются только на отдельных агрегатах. Предлагаемое решение позволит автоматически регулировать безопасность действий работников, поможет обнаружить и предотвратить деятельность в опасных зонах.

1. Основания для разработки

Основанием для разработки является задание на выпускную квалификационную работу, выданное на кафедре Математического и программного обеспечения ЭВМ Института информационных технологий по запросу заказчика АО "Северсталь менеджмент".

Дата утверждения: 10.02.2021.

Название темы разработки: «Разработка программного обеспечения скелетизации изображений человека для контроля опасных действий»

2. Назначение разработки

Проектируемое программное обеспечение предназначено повышения уровня безопасности на предприятии компании «Северсталь» путём анализа действий работников в реальном времени.

3. Требование к разработке

3.1 Требования к функциональным характеристикам

Необходимо разработать информационную систему контроля безопасности в реальном времени на производстве компании «Северсталь».

Она должна функционировать при помощи архитектуры толстого клиента, в которой приложение напрямую связано с базой данных. Система должна иметь поддержку нескольких типов баз данных. Обработка данных должна происходить с видеопотока в реальном времени, но для тестирования буду предоставлены специальные видеофайлы, следовательно необходимо произвести обработку с разных типов источников. Необходимо предусмотреть обработку нейронной сети для обнаружения частей тела работников с помощью видеокарты для быстроты работы.

Сформированный отчёт в MS Teams должен иметь подробное и понятное описание с прикрепленный изображением для того, чтобы в будущем можно было сделать вывод о корректности выговора.

К функциональным характеристикам разрабатываемого ПО предъявлены следующие краткие требования:

* отслеживать положение работников;
* отслеживать отдельные части тела работников;
* определять позу работников;
* должна быть предусмотрена возможность добавления нескольких зон опасности;
* у каждой зоны должна быть возможность обнаружения конкретных частей тела;
* необходимо придумать инструмент для комбинирования зон для формирования сложной опасной ситуации.

3.2 Требования к надёжности

С целью предотвращения ошибок во время работы информационной системы должны быть предусмотрены следующие обработчики исключительных ситуаций:

1. Система должна иметь проверку от ввода некоренных данных;

2. Система должна иметь проверку на вывод корректных данных, т. е если произошел сбой при взаимодействии с базой данных и данные не могут быть выведены, об этом сообщается пользователю;

3. Все элементы пользовательского интерфейса должны корректно отображаться на экране;

4. Также визуальная часть должна масштабироваться под размеры экрана.

3.3 Условия эксплуатации

Компьютеры и сервер предназначены для работы в закрытом отапливаемом помещении при следующих условиях:

* температура окружающего воздуха от +10°C до +35°C;
* относительная влажность воздуха не более 80%;
* запыленность воздуха не более 0,75 мг/м³;
* атмосферное давление от 630 до 800 мм ртутного столба;
* при работе с монитором расстояние от глаз должно быть 50-75 см;
* уровень шума не должен превышать 50 дБ;
* электропитание оборудования осуществляется от сети переменного тока напряжением 220 В и частотой 50 Гц.

3.4 Требование к составу и параметрам технических средств

Минимальная конфигурация:

* 2.6 GHz или более быстрый процессор;
* наличие 8 Гб ОЗУ или больше;
* объём жёсткого диска 200 Гб;
* видеокарта компании Nvidia серии GTX 1660 или выше для корректной работы нейронной сети;
* видеокарта и монитор, поддерживающие разрешение не менее чем

1270х768 точек и кадровую частоту не ниже 30 кадров в секунду.

Рекомендуемая конфигурация:

* тип процессора Intel с тактовой частотой более 3 ГГц;
* объем ОЗУ – не менее 16 Гб;
* видеокарта компании Nvidia серии GTX 2060 или выше для быстрой
* работы нейронной сети
* объём жёсткого диска не менее 500 Гб;
* видеокарта и монитор, поддерживающие разрешение 1920х1080 точек.

Требования к оборудованию, формирующему видеопоток:

* разрешение видеопотока не должно быть менее 640x480 точек;
* видеокамера должна находиться непосредственно подключенной к
* компьютеру или быть доступна в локальной сети.

3.5 Требования к информационной и программной совместимости

Программное обеспечение должно быть разработано при помощи языка программирования C#, использовать функционал EmguCV и использовать базу данных и канал в MS Teams для фиксации нарушений техники безопасности.

Для стабильного функционирования программного обеспечения необходимо наличие операционной системы Windows, Linux или MacOS и современного Интернет-браузер.

4. Требования к программной документации

Программная документация должна содержать расчётно-пояснительную записку (РПЗ) с содержанием: текст программы (прил. 2).

Документация оформляется на листах формата А4 по действующим стандартам на создание документации к программному обеспечению.

5. Стадии и этапы разработки

Стадии и этапы разработки программного обеспечения представлены в табл. П1.1.

Таблица П1.1

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| 1 | 2 | 3 | 4 |
| Разработка технического задания | 12.09.2021 | Готовое техническое задание |  |
| Изучение предметной области | 18.09.2021 | Предметная область изучена |  |
| Проведение сравнительного анализа аналогов проектируемого ПО | 25.09.2021 | Выявлены преимущества и недостатки аналогов |  |
| Выбор технологии, среды и языка программирования | 05.10.2021 | Выбраны технологии, среда и языки программирования |  |
| Анализ процесса обработки информации, выбор методов и алгоритмов для решения поставленной задачи | 24.10.2021 | Составлен алгоритм решения поставленной задачи |  |

Продолжение таблицы П1.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Разработка спецификаций проектируемого ПО | 20.11.2021 | Разработаны спецификации проектируемого ПО |  |
| Проектирование ПО | 10.01.2022 | Спроектировано ПО |  |
| Организация работ | 03.02.2022 | Выполнена организация работ |  |
| Разработка первой версии ПО | 15.03.2022 | Разработана первая версия ПО |  |
| Выбор методики тестирования и тестирование первой версии ПО | 30.03.2022 | Протестированная первая версия ПО |  |
| Разработка итоговой версии ПО | 25.04.2022 | Разработанное ПО |  |
| Выбор методики тестирования и тестирование ПО | 01.05.2022 | Протестированное ПО |  |
| Оформление документации | 18.05.2022 | Оформлена расчётно-пояснительная записка со всеми приложениями |  |

6. Порядок контроля и приемки

Контроль выполнения работы осуществляется преподавателем в соответствии с графиком, представленным в табл. П1.2.

Таблица П1.2

Порядок контроля и приёмки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  контрольного этапа  выполнения  курсового проекта | Сроки  контроля | Результат выполнения | Отметка о приемке  результата  контрольного этапа |
| Проверка технического задания | 12.09.2021 | Техническое задание утверждено |  |
| Демонстрация спроектированного ПО | 10.01.2022 | Спроектированное ПО согласованно |  |
| Демонстрация финальной версии ПО | 25.04.2022 | Финальная версия ПО утверждена |  |
| Демонстрация стратегии тестирования и проведённых тестов | 25.03.2022 | Стратегии тестирования утверждены, ПО работает исправно |  |
| Подготовка документации | 01.05.2022 | Расчётно-пояснительная записка прошла норм контроль и утверждена |  |
| Защита курсового проекта | 18.05.2022 | Отчёт по практике защищен |  |

Приложение 2

Текст программы

Текст класса MainViewModel представлен на рис. П2.1.

using Emgu.CV;

using Emgu.CV.CvEnum;

using Emgu.CV.Structure;

using MoreLinq;

using MultiPersonEstimators.AwaitCounter;

using MultiPersonEstimators.ExtensionsMethods;

using MultiPersonEstimators.MatExtensions;

using MultiPersonEstimators.Model;

using MultiPersonEstimators.Model.Humans;

using MultiPersonEstimators.Model.Preparing;

using MultiPersonEstimators.Subscribe;

using MultiPersonEstimators.ViewModel.Reports;

using ReactiveUI;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Collections.Specialized;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Net.Mail;

using System.Reactive.Linq;

using System.Windows.Input;

namespace MultiPersonEstimators.ViewModel

{

public class MainViewModel : ReactiveObject

{

private const int ZONE\_ROING\_DELAY = 100;

private const int HUMAN\_DISTRIBUTION\_DELAY = 100;

public IModel Model { get; }

public event Action Started;

public string FileName

{

get => fileName;

set => this.RaiseAndSetIfChanged(ref fileName, value);

}

private string fileName;

public bool CanRead

{

get => canRead;

set => this.RaiseAndSetIfChanged(ref canRead, value);

}

private bool canRead;

public ICommand Start { get; }

public ICommand OpenCamera { get; }

public byte[] FrameAsBytes

{

get => frameAsBytes;

set => this.RaiseAndSetIfChanged(ref frameAsBytes, value);

}

private byte[] frameAsBytes;

#region Zones

public ICommand Pause { get; }

public ICommand AddZone { get; }

public ICommand RemoveZone { get; }

public Zone CurrentZone

{ get => currentZone;

set => this.RaiseAndSetIfChanged(ref currentZone, value);

}

private Zone currentZone;

public ObservableCollection<ZoneGroup> ZoneGroups { get; } = new ObservableCollection<ZoneGroup>();

public ObservableCollection<Zone> Zones { get; } = new ObservableCollection<Zone>();

public bool ShowCurrentZonePanel

{

get => showCurrentZonePanel;

set => this.RaiseAndSetIfChanged(ref showCurrentZonePanel, value);

}

private bool showCurrentZonePanel;

public IEnumerable<System.Windows.Media.Brush> Colors { get; }

#endregion

#region ZonesQueries

public EmailSender EmailSender { get; } = EmailSender.DefaultSender;

public ICommand AddZonesQuery { get; }

public ICommand RemoveZonesQuery { get; }

public ZonesQuery CurrentZonesQuery

{ get => currentZonesQuery;

set => this.RaiseAndSetIfChanged(ref currentZonesQuery, value);

}

public ObservableCollection<ZonesQuery> ZonesQueries { get; } = new ObservableCollection<ZonesQuery>();

private ZonesQuery currentZonesQuery;

#endregion

public IEnumerable<Human> Humans

{

get => humans;

set => this.RaiseAndSetIfChanged(ref humans, value);

}

private IEnumerable<Human> humans = Enumerable.Empty<Human>();

public MainViewModel()

{

Model = new MainModel();

Colors = typeof(System.Windows.Media.Brushes)

.GetProperties()

.Where(x => x.Name.Count(char.IsUpper) == 1)

.Select(x => x.GetValue(null) as System.Windows.Media.Brush);

Start = ReactiveCommand.Create(StartMethod);

OpenCamera = ReactiveCommand.Create(OpenCameraMethod);

AddZone = ReactiveCommand.Create(() => Zones.Add(CreateNewZone()));

Pause = ReactiveCommand.Create(() => { });

RemoveZone = ReactiveCommand.Create(() => Zones.Remove(CurrentZone),

this.ObservableForProperty(x => x.CurrentZone, z => z != null));

Рис. П2.1. Текст класса MainViewModel

AddZonesQuery = ReactiveCommand.Create(AddZonesQueryMethod);

RemoveZonesQuery = ReactiveCommand.Create(RemoveZonesQueryMethod,

this.ObservableForProperty(x => x.CurrentZonesQuery, z => z != null));

Model.ObservableForProperty(mod => mod.DrawedFrame)

.Subscribe(arg => FrameAsBytes = arg.Value.ToBytes());

this.ObservableForProperty(x => x.CurrentZone, x => x != null)

.Subscribe(arg => ShowCurrentZonePanel = arg);

this.ObservableForProperty(x => x.Humans)

.Where(x => x.Value != null)

.Subscribe(\_ => Zones.ForEach(AddHumansToZone));

Zones.ObservableFor(NotifyCollectionChangedAction.Remove)

.Subscribe(zone =>

{

ZoneGroups.Remove(zone.RootZoneGroup);

zone.Dispose();

CurrentZone = null;

});

Zones.ObservableFor(NotifyCollectionChangedAction.Add)

.Subscribe(zone =>

{

foreach (var point in zone.Points)

{

point.WhenAnyValue(p => p.X, p => p.Y)

.Where(\_ => Model.Frame != null)

.Throttle(TimeSpan.FromMilliseconds(ZONE\_ROING\_DELAY))

.Subscribe(\_ => RoiZone(zone));

point.WhenAnyValue(p => p.X, p => p.Y)

.Where(\_ => Humans != null)

.Throttle(TimeSpan.FromMilliseconds(HUMAN\_DISTRIBUTION\_DELAY))

.Subscribe(arg => AddHumansToZone(zone));

}

foreach (var selectedHumanPart in zone.SelectedBodyParts)

{

selectedHumanPart.WhenAnyValue(x => x.Value, x => x.IsSelected, x => x.Location)

.Where(\_ => Humans != null)

.Subscribe(\_ => AddHumansToZone(zone));

}

CurrentZone = zone;

});

Model.ObservableForProperty(x => x.DrawedFrame)

.Where(arg => arg.Value != null)

.Throttle(TimeSpan.FromMilliseconds(ZONE\_ROING\_DELAY))

.Subscribe(\_ => Zones.ForEach(RoiZone));

Model.ObservableForProperty(x => x.HumansPoints)

.Where(arg => arg.Value != null)

.Subscribe(arg => Humans = PrepareHumans(arg.Value));

Model.ObservableForProperty(x => x.MillisecondsForFrame)

П2.1. Продолжение

.Subscribe(arg => Zones.ForEach(z => z.CurrentTicks += arg.Value));

}

private void AddHumansToZone(Zone zone)

{

var humansBodyParts = new List<HumanBodyPartsInZone>();

foreach (var human in humans)

{

var humanPartsInZone = new List<BodyPart>();

foreach (var bodyPartPoint in human.Points.Where(p => p != null))

{

if (zone.CheckBodyPartPoint(bodyPartPoint))

{

humanPartsInZone.Add(bodyPartPoint.BodyPart);

}

}

if (humanPartsInZone.Count > 0)

{

humansBodyParts.Add(new HumanBodyPartsInZone { Human = human, BodyParts = humanPartsInZone });

}

}

zone.HumanBodyParts = humansBodyParts;

}

private void RoiZone(Zone zone)

{

var rect = GetRect(zone.Points);

if (rect.Width > 0 && rect.Height > 0)

{

zone.Roi = new Mat(Model.Frame, rect).ToBytes();

}

}

private int zonesId = 0;

private Zone CreateNewZone()

{

var zone = new Zone(0.1, 0.1, 0.2, 0.2)

{

Name = $"Зона {zonesId++}"

};

zone.RootZoneGroup = new ZoneGroup { Group = $"{zone.Name} группа" };

zone.ZoneGroup = zone.RootZoneGroup;

zone.ZoneGroups = ZoneGroups;

ZoneGroups.Add(zone.RootZoneGroup);

return zone;

}

private IEnumerable<Human> PrepareHumans(Point[,] points)

{

var findedHumans = new List<Human>();

for (int i = 0; i < points.GetLength(0); i++)

{

var humanPoints = points.GetArray(i);

var human = new Human

{

Name = $"Человек {i}",

Points = humanPoints.Select

(

(p, i) => p.IsEmpty() ? null : new BodyPartPoint

{

Point = new Model.Points.Point((double)p.X / Model.Frame.Width, (double)p.Y / Model.Frame.Height),

BodyPart = (BodyPart)i

}

).ToList(),

Poses = Model.PoseDetector.GetPoses(humanPoints)

};

var humanRect = GetRect(human.Points.Where(x => x != null).Select(x => x.Point));

if (humanRect.Width > 0 && humanRect.Height > 0)

{

human.Roi = new Mat(Model.Frame, humanRect).ToBytes();

}

findedHumans.Add(human);

}

return findedHumans;

}

private void AddZonesQueryMethod()

{

var zoneQuery = new ZonesQuery { AllZones = Zones, AllHumans = Humans };

this.ObservableForProperty(x => x.Humans)

.Subscribe(arg => zoneQuery.AllHumans = arg.Value)

.AddSubscribeBy(zoneQuery);

zoneQuery.ObservableForProperty(x => x.WrongHumans)

.Where(h => false)

.Subscribe

(

async arg =>

{

var frame = new Mat();

string text = $"Люди, не удовлетворяющие зонам {(zoneQuery.Inverted ? "инвертируемого (зоны не обнаружили - нарушение)" : "обычного (зоны обнаружили - нарушение)")} запроса:\n";

lock (Model.DrawedFrame)

{

Model.DrawedFrame.CopyTo(frame);

text += string.Concat(arg.Value.Select(x => $"{x.Name} "));

text += "\n\nЛюди, проверяемые запросом:\n";

text += string.Concat(zoneQuery.AllHumans.Select(x => $"{x.Name} "));

text += "\n\nЗоны запроса:\n";

foreach (var zoneGroup in zoneQuery.QueryZones.GroupBy(x => x.ZoneGroup))

{

text += $"{zoneGroup.Key.Group}: {string.Concat(zoneGroup.Select(x => $"{x} "))}\n";

}

foreach (var zone in zoneQuery.QueryZones)

{

(double x, double y) cent = (zone.Points.Min(x => x.X), zone.Points.Average(x => x.Y));

CvInvoke.PutText(frame, zone.Name, new Point((int)(cent.x \* Model.DrawedFrame.Width), (int)(cent.y \* Model.DrawedFrame.Height)),

П2.1. Продолжение

FontFace.HersheyComplex, 1.5, new MCvScalar(0, 0, 255), 2);

CvInvoke.Polylines(frame, zone.Points

.Select(p => new Point((int)(p.X \* Model.DrawedFrame.Width), (int)(p.Y \* Model.DrawedFrame.Height)))

.ToArray(),

true,

new MCvScalar(50, 0, 225),

2);

}

}

using var byteStream = new MemoryStream(frame.ToBytes());

await EmailSender.SendMessageAsync("НАРУШЕНИЕ!", text, new Attachment(byteStream, "warning.png"));

}

);

ZonesQueries.Add(zoneQuery);

}

private void RemoveZonesQueryMethod()

{

CurrentZonesQuery.Unsubscribe();

ZonesQueries.Remove(CurrentZonesQuery);

}

private void StartMethod()

{

CanRead = true;

Started?.Invoke();

if (CanRead)

{

Model.VideoCapture = new VideoCapture(FileName);

}

}

private void OpenCameraMethod()

{

CanRead = true;

if (CanRead)

{

Model.VideoCapture = new VideoCapture(0);

}

}

private Rectangle GetRect(IEnumerable<Model.Points.Point> points)

{

var (startX, startY, endX, endY) = (points.Min(p => p.X), points.Min(p => p.Y), points.Max(p => p.X), points.Max(p => p.Y));

int x = (int)(Model.Frame.Width \* startX);

int y = (int)(Model.Frame.Height \* startY);

int width = (int)(Model.Frame.Width \* endX) - x;

int height = (int)(Model.Frame.Height \* endY) - y;

return new Rectangle(x, y, width, height);

}

}

public class Class1

{

}

}

Текст класса Zone представлен на рисунке П2.2

using MultiPersonEstimators.ExtensionsMethods;

using MultiPersonEstimators.Model.Humans;

using MultiPersonEstimators.Model.Points;

using MultiPersonEstimators.Model.Preparing;

using ReactiveUI;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Diagnostics;

using System.Linq;

using System.Text.Json;

using System.Windows.Media;

namespace MultiPersonEstimators.ViewModel

{

public class Zone : ReactiveObject, IDisposable

{

public event EventHandler Disposed;

public string Name

{

get => name;

set => this.RaiseAndSetIfChanged(ref name, value);

}

private string name;

public byte[] Roi

{

get => roi;

set => this.RaiseAndSetIfChanged(ref roi, value);

}

private byte[] roi;

public ZoneGroup RootZoneGroup { get; set; }

public ZoneGroup ZoneGroup

{

get => zoneGroup;

set => this.RaiseAndSetIfChanged(ref zoneGroup, value);

}

private ZoneGroup zoneGroup;

public IEnumerable<ZoneGroup> ZoneGroups { get; set; }

public IEnumerable<Human> Humans => HumanBodyParts.Select(x => x.Human).Distinct();

public IEnumerable<Point> Points { get; }

public IEnumerable<HumanBodyPartsInZone> HumanBodyParts

{

get => humanBodyParts;

set

{

casheHumanBodyParts = value;

int humansCount = value.Select(x => x.Human).Distinct().Count();

if (humansCount == 0)

{

CurrentTicks = 0;

}

this.RaiseAndSetIfChanged(ref humanBodyParts,

CurrentTicks >= MaxTicks && humansCount > MaxHumansCount ?

value : Enumerable.Empty<HumanBodyPartsInZone>());

}

}

private IEnumerable<HumanBodyPartsInZone> humanBodyParts;

private IEnumerable<HumanBodyPartsInZone> casheHumanBodyParts = Enumerable.Empty<HumanBodyPartsInZone>();

public IEnumerable<Selected<BodyPart>> SelectedBodyParts { get; }

public IEnumerable<Selected<string>> SelectedTexts { get; }

private IReadOnlyDictionary<string, IEnumerable<BodyPart>> textToHumanParts = new Dictionary<string, IEnumerable<BodyPart>>

{

{ "Голова", new [] { BodyPart.Head } },

{ "Руки",

new []

{

BodyPart.RightShoulder, BodyPart.RightElbow, BodyPart.RightWrist,

BodyPart.LeftShoulder, BodyPart.LeftElbow, BodyPart.LeftWrist

}

},

{ "Ноги",

new []

{

BodyPart.RightHip, BodyPart.RightKnee, BodyPart.RightAnkle,

BodyPart.LeftHip, BodyPart.LeftKnee, BodyPart.LeftAnkle

}

}, {

"Тело",

new []

{

BodyPart.Neck, BodyPart.LeftShoulder, BodyPart.RightShoulder,

BodyPart.Hip, BodyPart.LeftHip, BodyPart.RightHip

}

}

};

public Brush Color

{

get => color;

set => this.RaiseAndSetIfChanged(ref color, value);

}

private Brush color = Brushes.Red;

public double Opacity

{

get => opacity;

set => this.RaiseAndSetIfChanged(ref opacity, value);

} private double opacity = 0.5;

public long MaxTick

{

get => maxTicks;

set => this.RaiseAndSetIfChanged(ref maxTicks, value);

} private long maxTicks = 0;

public long CurrentTicks

{ get => currentTicks;

set => this.RaiseAndSetIfChanged(ref currentTicks, value);

}

private long currentTicks = 0;

public bool CanRenameGroup

{ get => canRenameGroup;

set => this.RaiseAndSetIfChanged(ref canRenameGroup, value); } private bool canRenameGroup;

public int MaxHumansCount

{

Рис. П2.2. Текст класса Zone

get => maxHumansCount;

set => this.RaiseAndSetIfChanged(ref maxHumansCount, value);

}

private int maxHumansCount = 0;

public Zone(double left, double top, double right, double bot)

{

Points = new ObservableCollection<Point>

{

new Point(left, top),

new Point(right, top),

new Point(right, bot),

new Point(left, bot)

};

this.WhenAnyValue(x => x.RootZoneGroup, x => x.ZoneGroup)

.Subscribe(\_ => CanRenameGroup = RootZoneGroup == ZoneGroup);

this.ObservableForProperty(x => x.MaxTicks)

.Subscribe(\_ => CurrentTicks = 0);

this.WhenAnyValue(x => x.MaxTicks, x => x.MaxHumansCount)

.Subscribe(arg => HumanBodyParts = casheHumanBodyParts);

foreach (var point in Points)

{

point.WhenAnyValue(p => p.X, p => p.Y)

.Subscribe(\_ => CurrentTicks = 0);

}

SelectedBodyParts = Enum.GetValues(typeof(BodyPart))

.Cast<BodyPart>()

.Select(x => new Selected<BodyPart> { Value = x })

.ToList();

SelectedTexts = textToHumanParts.Keys

.Select(x => new Selected<string> { Value = x })

.ToList();

foreach (var selectedText in SelectedTexts)

{

selectedText.ObservableForProperty(x => x.IsSelected)

.Subscribe(arg =>

{

foreach (var selectedHumanPart in SelectedBodyParts

.Where(x => textToHumanParts[selectedText.Value]

.Contains(x.Value)))

{

selectedHumanPart.IsSelected = arg.Value;

}

});

selectedText.ObservableForProperty(x => x.Location)

.Subscribe(arg =>

{

foreach (var selectedHumanPart in SelectedBodyParts

.Where(x => textToHumanParts[selectedText.Value]

.Contains(x.Value)))

{

selectedHumanPart.Location = arg.Value;

}

}); }

this.ObservableForProperty(x => x.Opacity)

.Subscribe(\_ => Opacity = Math.Round(Opacity, 2));

}

П2.2. Продолжение

public bool ContainsBodyPart(BodyPart bodyPart)

{

return SelectedBodyParts

.Where(p => p.IsSelected)

.Select(p => p.Value)

.Contains(bodyPart);

}

public bool ContainsPoint(Point point)

{

var poly = Points.ToArray();

bool inside = false;

for (int i = 0, j = poly.Length - 1; i < poly.Length; j = i++)

{

if (poly[i].Y > point.Y != poly[j].Y > point.Y

&& point.X < (poly[j].X - poly[i].X) \* (point.Y - poly[i].Y) / (poly[j].Y - poly[i].Y) + poly[i].X)

{

inside = !inside;

}

}

return inside;

}

public bool CheckBodyPartPoint(BodyPartPoint bodyPartPoint)

{

bool containsPoint = ContainsPoint(bodyPartPoint.Point);

foreach (var part in SelectedBodyParts.Where(p => p.IsSelected))

{

if (part.Value == bodyPartPoint.BodyPart)

{

if (part.Location == Location.Inside && containsPoint)

{

return true;

}

if (part.Location == Location.OutSide && !containsPoint)

{

return true;

}

}

}

return false;

}

public void Dispose()

{

Disposed?.Invoke(this, EventArgs.Empty);

Disposed = null;

}

public override string ToString()

{

return $"{Name} [{string.Concat(SelectedTexts.Where(x => x.IsSelected).Select(x => $"({x.Value} {x.Location.ToDescriptionOrString()}) "))}]";

}

}

}

Текст класса Query представлен на рисунке П2.3

using MultiPersonEstimators.Model;

using MultiPersonEstimators.Model.Humans;

using MultiPersonEstimators.Subscribe;

using ReactiveUI;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Windows.Input;

namespace MultiPersonEstimators.ViewModel

{

public class ZonesQuery : ReactiveObject

{

public ObservableCollection<Zone> QueryZones { get; } = new ObservableCollection<Zone>();

public bool Inverted

{

get => inverted;

set => this.RaiseAndSetIfChanged(ref inverted, value);

}

private bool inverted = false;

public Zone AddableZone

{

get => addableZone;

set => this.RaiseAndSetIfChanged(ref addableZone, value);

}

private Zone addableZone;

public Zone RemovableZone

{

get => removableZone;

set => this.RaiseAndSetIfChanged(ref removableZone, value);

}

private Zone removableZone;

public IEnumerable<Zone> AllZones { get; set; }

public IEnumerable<QueryHumans> QuerysHumans { get; }

public IEnumerable<Human> WrongHumans

{

get => wrongHumans;

set => this.RaiseAndSetIfChanged(ref wrongHumans, value);

}

private IEnumerable<Human> wrongHumans;

public IEnumerable<Human> AllHumans

{

get => allHumans;

set => this.RaiseAndSetIfChanged(ref allHumans, value);

}

private IEnumerable<Human> allHumans;

public QueryHumans CurrentQueryHumans

{ get => currentQueryHumans;

set => this.RaiseAndSetIfChanged(ref currentQueryHumans, value);

} private QueryHumans currentQueryHumans = QueryHumans.All;

public bool CurrentQueryHumansIsFromZone

{

get => currentQueryHumansIsFromZone;

set => this.RaiseAndSetIfChanged(ref currentQueryHumansIsFromZone, value);

} private bool currentQueryHumansIsFromZone = false;

public Zone CurrentQueryHumansZone

{

get => currentQueryHumansZone;

set => this.RaiseAndSetIfChanged(ref currentQueryHumansZone, value);

}

private Zone currentQueryHumansZone;

public ICommand AddZone { get; }

public ICommand RemoveZone { get; }

public ZonesQuery()

{

QuerysHumans = Enum.GetValues(typeof(QueryHumans)).Cast<QueryHumans>().ToList();

AddZone = ReactiveCommand.Create(AddZoneMethod,

this.ObservableForProperty(x => x.AddableZone, z => z != null && !QueryZones.Contains(AddableZone)));

RemoveZone = ReactiveCommand.Create(RemoveZoneMethod,

this.ObservableForProperty(x => x.RemovableZone, z => z != null));

this.ObservableForProperty(x => x.CurrentQueryHumans, x => x == QueryHumans.FromZone)

.Subscribe(arg =>

{

CurrentQueryHumansIsFromZone = arg;

if (!arg)

{

CurrentQueryHumansZone = null;

}

});

this.WhenAnyValue(x => x.AllHumans, x => x.CurrentQueryHumans, x => x.CurrentQueryHumansZone, x => x.Inverted)

.Subscribe(\_ => Calculate());

}

public void Calculate()

{

var wrongHumans = new List<Human>();

var queryHumans = CurrentQueryHumans switch

{

QueryHumans.All => AllHumans,

QueryHumans.FromZone => CurrentQueryHumansZone?.Humans ?? Enumerable.Empty<Human>(),

\_ => Enumerable.Empty<Human>(),

};

foreach (var queryZone in QueryZones.GroupBy(x => x.ZoneGroup))

{

var zoneHumans = queryZone.SelectMany(x => x.Humans).ToList();

foreach (var human in queryHumans)

{

bool zoneHumansContains = zoneHumans?.Contains(human) ?? false;

if (zoneHumansContains ^ Inverted)

{

wrongHumans.Add(human);

}

}

}

Рис. П2.3. Текст класса Query

П2.3. Продолжение

WrongHumans = wrongHumans.Distinct().OrderBy(x => x.Name).ToList();

}

private void AddZoneMethod()

{

AddableZone.WhenAnyValue(x => x.HumanBodyParts, x => x.ZoneGroup)

.Subscribe(\_ => Calculate())

.AddSubscribeBy((this, AddableZone));

QueryZones.Add(AddableZone);

var disposingZone = AddableZone;

AddableZone.Disposed += ZoneDisposed;

void ZoneDisposed(object sender, EventArgs e)

{

if (QueryZones.Contains(disposingZone))

{

RemovableZone = disposingZone;

RemoveZoneMethod();

}

}

AddableZone = null;

Calculate();

}

private void RemoveZoneMethod()

{

(this, RemovableZone).Unsubscribe();

QueryZones.Remove(RemovableZone);

Calculate();

}

}

}

Приложение 3

Спецификации

Настоящее приложение содержит спецификации на разработанную программную документацию и компоненты программного обеспечения. Они приведены в табл. П3.1.

Табл. П3.1

Спецификации

|  |  |  |
| --- | --- | --- |
| Обозначение | Описание | Примечание |
| 1 | 2 | 3 |
| Shell.xaml | компонент, характеризующий основное окно приложения |  |
| DetectionControl.xaml | компонент, характеризующий окно детекции |  |
| QueriesControl.xaml | компонент, характеризующий форму запросов |  |
| DetectionZonesControl.xaml | компонент, характеризующий форму зон в окне детекции |  |
| CameraDialogControl.xaml | компонент, характеризующий диалог выбора камеры |  |
| DrawingZonesControl.xaml | компонент, характеризующий форму отображения опасных зон |  |
| ZoneEllipse.cs | компонент, характеризующий визуальную точку зоны |  |
| ZonePolygon.cs | компонент, характеризующий визуальную область зоны |  |
| ZonesControl.xaml | компонент, характеризующий окно зон |  |
| HumansControl.xaml | компонент, характеризующий окно людей |  |
| SettingsControl.xaml | компонент, характеризующий окно настроек |  |

Продолжение таблицы П3.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| ZonesViewModel.cs | класс, характеризующий модель представления зон |  |
| ZoneConsumer.cs | класс, характеризующий модель представления для класса, которому нужно знать о зонах |  |
| DetectionViewModel.cs | класс, характеризующий модель представления детекции |  |
| DetectionZonesViewModel.cs | класс, характеризующий модель представления зон в детекции |  |
| ShellViewModel.cs | класс, характеризующий основную модель представления |  |
| SettingsViewModel.cs | класс, характеризующий модель представления настроек |  |
| OpenCameraDialogViewModel.cs | класс, характеризующий модель представления для диалога выбора камеры |  |
| HumansViewModel.cs | класс, характеризующий модель представления людей |  |
| QueriesViewModel.cs | класс, характеризующий модель представления запросов |  |
| DetectionModel.cs | класс, характеризующий модель детекции |  |
| HumanWithRoi.cs | класс, описывающий человека с его областью |  |
| SettingsModel.cs | класс, характеризующий модель настроек |  |
| QueriesModel.cs | класс, характеризующий модель запросов |  |
| Query.cs | класс, описывающий запрос |  |

Продолжение таблицы П3.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| DetectionZonesModel.cs | класс, характеризующий модель зон в детекции |  |
| Zone.cs | класс, описывающий зону |  |
| ZoneFactory.cs | класс, описывающий логику создания новой зоны |  |
| SelectableBodyPart.cs | класс, характеризующий выбранную часть тела |  |
| HumansCheckResult.cs | класс, описывающий результат проверки человека |  |
| NetOption.cs | класс, описывающий конфигурацию нейронной сети |  |
| Detector.cs | класс, необходимый для детектирования точек частей тела |  |
| Finder.cs | класс, необходимый для детектирования людей |  |
| Drawer.cs | класс, отвечающий за логику отрисовки людей на кадре |  |
| Preparer.cs | класс, отвечающий за логику преобразования найденных точек частей тела |  |
| HumansConverter.cs | класс, отвечающий за логику преобразования найденных точек частей тела в людей |  |
| Human.cs | класс, описывающий обнаруженного человека |  |
| Point.cs | класс, описывающий точку с относительными координатами |  |
| VideoCaptureCameraFactory.cs | класс, описывающий логику открытия видеопотока с камеры |  |

Продолжение таблицы П3.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| VideoCaptureFileFactory.cs | класс, описывающий логику открытия видеопотока из файла |  |
| VideoService.cs | класс, содержащий логику работы с видео |  |
| VideoProcessingHandler.cs | класс, описывающий обработчика нейронной сети |  |
| ReportService.cs | класс, содержащий логику работы с отправкой отчетов |  |
| Report.cs | класс, описывающий отчёт |  |
| DatabaseContext.cs | класс, описывающий схему базы данных |  |
| PoseEntity.cs | класс, описывающий таблицу поз |  |
| PointEntity.cs | класс, описывающий таблицу точек |  |
| HumanEntity.cs | класс, описывающий таблицу людей |  |
| ReportEntity.cs | класс, описывающий таблицу отчётов |  |
| BodyPartEntity.cs | класс, описывающий таблицу частей тела |  |
| TeamsSender.cs | класс, содержащий логику для отправки данных на канал в Teams |  |
| EmailSender.cs | класс, содержащий логику для отправки данных на почтовый ящик |  |
| VideoReader.cs | класс, содержащий логику чтения видеопотока в отдельном потоке |  |
| VideoCaptureFactory.cs | класс, описывающий логику создания видеопотока |  |
| VideoDeviceResolver.cs | класс, содержащий логику получения информации о доступных видеокамерах |  |

Продолжение таблицы П3.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| VideoDeviceInfo.cs | класс, описывающий информацию о видеокамере |  |
| BodyPartsExtensionsMethod.cs | Класс, содержащий методы для работы с частями чела |  |
| ConfigureExtensionsMethod.cs | Класс, содержащий методы для работы с конфигурацией |  |
| MatrixExtensionsMethod.cs | Класс, содержащий методы для работы с матрицами |  |
| PointsExtensionsMethod.cs | Класс, содержащий методы для работы с точками |  |
| BodyPart.cs | класс, описывающий все возможные части тела |  |
| VideoCaptureFactoryType.cs | класс, описывающий все возможные фабрики создания видеопотока |  |
| SendType.cs | класс, описывающий все возможные варианты отправки отчёта |  |
| VideoCaptureFactoryException.cs | класс, описывающий исключение фабрики создания видеопотка |  |
| DatabaseSendException.cs | класс, описывающий исключение отправки данных в базу данных |  |
| TeamsSendException.cs | класс, описывающий исключение отправки данных в Teams |  |