МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

01.03.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

БАКАЛАВРСКАЯ РАБОТА

на тему: **Модели и алгоритмы трансформации изображений для виртуальной примерочной интернет-магазина одежды**

Студентка Н.В. Елюшкина _____

Руководитель Т.Г. Султанов _____

Консультант по аннотации Н.В. Ященко _____

Допустить к защите
Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____

«_____» _____ 20___ г.

Тольятти 2017

Аннотация

Название бакалаврской работы – «Модели и алгоритмы трансформации изображений для виртуальной примерочной интернет-магазина одежды».

Цель работы: разработать алгоритмы, трансформирующий одежду с учетом особенностей фигуры клиента, чтобы снизить риск во время покупки в интернет-магазине.

Объект работы: алгоритмы трансформации изображений в виртуальной примерочной.

Предмет исследования: процесс преобразования изображений в виртуальной примерочной.

В работе рассматривается разработка алгоритма, который строит фигуру, основываясь на ключевых точках, и алгоритма трансформации изображения одежды с учетом особенностей построенной фигуры.

Покупая в интернет-магазинах, нельзя быть уверенным в том, что приедет именно то, что хотелось: купленная вещь может плохо сидеть или не подходить к внешности покупателя. Использование виртуальной примерочной призвано частично решить эту проблему. Разработанные алгоритмы «надевают» виртуальную одежду на виртуальную фигуру клиента, чтобы он мог сделать более взвешенный выбор. Работа этих алгоритмов основана на использовании «ключевых точек», которые указывает клиент.

В первой главе рассматриваются плюсы и минусы интернет-шоппинга, выявляется необходимость создания приложения примерочной, строится математическая модель.

Во второй главе описывается процесс разработки алгоритма.

В третьей главе представляются результаты тестирования разработанных алгоритмов. Они тестируются на трех типах фигуры с разными типами одежды. Сравнение с алгоритмом пропорционального масштабирования показало, что разработанные алгоритмы справились с задачей лучше.

Бакалаврская работа выполнена на 47 страницах, состоит из введения, трёх глав, включающих 28 изображений, 1 таблицу и 13 формул, заключения, списка используемых источников, включающего 36 источников, из них 5 на иностранном языке, и 2-х приложений.

Abstract

The title of the graduation work is *Models and Algorithms of Image Transformation for Virtual Dressing Room of Online Clothing Store*. The object of the graduation work is image transformation process in the virtual dressing room. The subject of the graduation work is creation of algorithms for this transformation. The aim of the work is to develop the algorithms which transform image of clothes according to client's figure. The work touches on the development of the algorithm that builds a figure based on 'key points'. We develop the algorithms that transform image considering the features of the client's figure.

Buying in online stores you can't be sure that it will come exactly what you wanted. The bought item may to be a bad fit or isn't suitable the client's appearance. The virtual dressing room partially solves this problem. The developed algorithms 'put on' virtual clothes on client's virtual figure, so that he can make a deliberate choice. So, we study image transformation algorithms and choose one which will become the basis of the new algorithms. Next we concentrate on formation of a mathematical model of the algorithm of image transform and the algorithm of building a figure. Both algorithms are based on using the "key points" that the client points out.

We first analyze advantages and disadvantages of online shopping and build a mathematical model.

The next part describes the development process of the algorithms.

Finally, we show test results of the algorithms. They were tested on three types of figures with different types of clothes and compared with the proportional scaling algorithm. Testing showed that the developed algorithms coped with the task better than proportional scaling algorithm.

The graduation work consists of an explanatory note on 47 pages, introduction, three parts, including 28 figures, 1 table, 13 formulas, conclusion, the list of 36 references including 5 foreign sources and 2 appendices.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Информационные технологии стремительно развиваются, как и сфера их применения. Трудно перечислить весь спектр их возможного использования. Одной из важнейших областей, связанных с IT, является Всемирная паутина: интернет-технологии уже стали неотъемлемой частью жизни и продолжают развиваться. Интернет предоставляет пользователям почти неограниченные возможности в работе с разнообразнейшей информацией.

Все большую популярность набирает интернет-шоппинг, и это неудивительно: это удобно, занимает в разы меньше времени, позволяет получить доступ к товару, который в магазине не купишь, цены в интернет-магазинах зачастую ниже реальных. Но, к сожалению, у медали есть обратная сторона: известно, что осуществление покупок таким образом сопряжено с риском. Это может быть вина почты, потерявшей посылку; вина продавца, отправившего не то, что ожидалось; вина производителя, выпустившего брак; наконец, ЧП в виде независящих от человека факторов.

Интернет-магазины одежды здесь уязвимы едва ли не больше остальных: помимо уже перечисленных факторов, далеко не каждый может подобрать подходящую вещь, ориентируясь лишь на фото. Часто случается, что риск не оправдывается, и пользователь зарекается пользоваться такими магазинами. Разрабатываемый в данной работе алгоритм призван частично решить эту проблему, предлагая возможность примерить виртуальную одежду перед покупкой.

Актуальность данной работы не вызывает сомнений: проведенное исследование показало, что несмотря на общее развитие интернет-шоппинга, виртуальные примерочные есть далеко не в каждом магазине. В тех же, в которых они есть, они обладают довольно бедным функционалом и спорной возможностью примерки.

Предмет исследования: процесс преобразования изображений в виртуальной примерочной.

Объект исследования: алгоритмы трансформации изображений в виртуальной примерочной.

Цель работы: разработать алгоритм, трансформирующий одежду с учетом особенностей фигуры клиента, чтобы снизить риск во время покупки в интернет-магазине.

Для достижения цели решаются следующие задачи:

- рассмотрение алгоритмов масштабирования и выбор алгоритма для дальнейшей разработки;
- изучение выбранного алгоритма;
- рассмотрение способов построения фигуры клиента и выбор одного из них;
- изучение выбранного способа;
- выбор технологии реализации;
- реализация алгоритмов и демонстрационной программы;
- тестирование разработанных алгоритмов.

Выпускная квалификационная работа состоит из введения, трех глав и заключения.

В первой главе проводится анализ выбранной темы, определяется математическая модель алгоритма. Рассматриваются и выбираются наиболее подходящие для дальнейшей разработки математические алгоритмы.

Во второй главе обосновывается выбор технологии реализации, а также описывается процесс разработки алгоритма.

Третья глава посвящена тестированию: проверяется работа алгоритма с различными фигурами и типами одежды, делаются выводы.

1 РАЗРАБОТКА МАТЕМАТИЧЕСКОЙ МОДЕЛИ ДЛЯ АЛГОРИТМОВ ТРАНСФОРМАЦИИ ИЗОБРАЖЕНИЙ

1.1 Анализ предметной области

Мы живем в век высоких технологий, когда компьютеры прочно вошли в жизнь и стали незаменимыми помощниками в решении разнообразнейших задач. Значительную часть работы, выполняемой раньше человеком, теперь выполняют они, в разы ускоряя ее выполнение и экономя немало средств.

Среди прочего можно выделить работу с изображениями. С ними связан огромный пласт задач, выполняемых компьютером. Трудно найти область, где не используются изображения и какие-либо операции с ними. Медицина, астрономия, радио-, тепло-, гидролокация, телевидение не достигли бы таких высот, как сегодня, без использования компьютерной обработки изображений. Они используются в задачах идентификации личности, распознавании текста и образов, мультипликации, архитектуре и многих других. Машинное зрение широко используется в медицине. УЗИ, рентгенография – известные каждому примеры такого использования. Но это лишь верхушка айсберга. Так, бинаризация (перевод цветного изображения в монохромное) может помочь в обнаружении опухолей и нарушений кровообращения, а вычитание изображений позволит определить наиболее активные клетки и проследить их передвижение. Обработка снимков со спутников позволяет ученым провести необходимые наблюдения и сделать новые открытия. Они широко используются в навигации, обнаружении лесных пожаров, оценки экологического состояния регионов. Даже в повседневной жизни изображения и задачи их обработки встречаются буквально на каждом шагу. Сегодня трудно найти человека, у которого нет камеры и который хотя бы раз не пользовался услугами фоторедактора.

Появление первых персональных компьютеров вывело их применение за рамки необходимости. Рядовой пользователь получил доступ к новым возможностям, которые способны упростить жизнь или просто развлечь.

Появление интернета сделало ПК неиссякаемым источником информации. Каждый пользователь получает доступ к библиотеке, включающей миллионы книг [1], коллекции фильмов из тысяч кинолент [2] и более чем миллиарду сайтов самой разной направленности [3]. Интернет-технологии стремительно развиваются, позволяя простому человеку, сидя вечером перед монитором, читать последние новости, смотреть новую серию любимого сериала, изучать основы квантовой физики и даже путешествовать в любую точку земного шара.

Среди занятий, предлагаемых пользователям сети, можно выделить интернет-шопинг, то есть посещение интернет-магазинов. Это направление получает все большее распространение, и уже сейчас некоторые аналитики высказывают мнение, что виртуальные магазины со временем могут вытеснить реальные. Трудно сказать, насколько перспективна такая точка зрения, но то, что интернет-магазины прочно заняли свою нишу, уже неоспоримый факт. У интернет-шопинга немало плюсов, привлекающих потенциальных клиентов: ленивых или, напротив, занятых людей, искателей специфичного товара.

Сегодня совершать в сети покупки можно, находясь практически в любой стране, при этом продавец может жить на другой стороне земного шара. Если с расширением реального магазина усложняется поиск нужного товара, то на сайте того же магазина его можно найти на несколько секунд. Интернет-магазин открыт в любое удобное время, а сама покупка через сайт займет 10-15 минут. В большинстве интернет-магазинов некоторые товары стоят дешевле, чем в обычном магазине. Существуют интернет-магазины, в которых продаются необычные, редкие вещи, эксклюзив или изделия ручной работы, которые в обычных магазинах найти невозможно.

В то же время покупки в интернете сопряжены с рядом минусов. Долгое ожидание, вплоть до нескольких месяцев, доставка может обойтись дороже, чем сам товар, шанс утери посылки в пути – а вместе с ней и покупки, и денег. Не исключена встреча с недобросовестным продавцом,

который отправит некачественный товар или не отправит вовсе. Наконец, может приехать бракованный товар.

Выбирая в интернет-магазине одежду и обувь, можно ошибиться с размером. Вещь может не понравиться внешне и не подойти к внешности при примерке, хотя на фото магазина смотрелась великолепно. В таких случаях можно потребовать обмена товара или возвращения денег за него, но это не всегда возможно и удобно. Стремление хотя бы частично обезопасить себя от неудачной покупки приводит к необходимости появления виртуальных примерочных.

Виртуальная примерочная – приложение, позволяющее пользователю «примерить» перед покупкой вещь из онлайн-каталога. Несмотря на то, что необходимость наличия такой функции кажется очевидной, многие известные магазины ее лишены. В тех, в которых она присутствует, примерочные совершенно не похожи друг на друга и по-разному выполняют свою задачу. Однако приложения, которое бы позволяло покупателю полноценно оценить вещь на своей собственной фигуре, не существует. Анализ разных примерочных показал, что большинство из них успешно «примеряют» вещь лишь на несколько заданных моделей со своей фигурой и типом внешности. Другие работают с манекеном, параметры которого можно задать. Такое приложение позволит оценить посадку будущей покупки, но нельзя предугадать, как она будет смотреться при типе внешности покупателя. А это является большим недостатком таких примерочных: даже идеально сидящая вещь не будет радовать, если попросту не идет.

Разрабатываемые алгоритмы призваны частично решить эту проблему, так как для «примерки» покупатель будет использовать свое фото. Таким образом будут решаться сразу две задачи: можно будет оценить и посадку, и вид самой вещи.

1.2 Проектирование алгоритмов трансформации изображений

1.2.1 Анализ алгоритмов масштабирования изображений

Работу разрабатываемых алгоритмов трансформации можно условно разделить на два этапа. Первый – построение фигуры клиента. Второй – трансформация изображения одежды с учетом особенностей построенной фигуры. Для разных типов одежды разрабатываемые алгоритмы будут отличаться способом обработки этого изображения.

Таким образом, сначала необходимо спроектировать два алгоритма, которые лягут в основу нового: алгоритм масштабирования для изменения одежды и алгоритм построения кривой для получения фигуры.

Компьютерные изображения делятся на два семейства – растровые и векторные. Растровые изображения представляют собой матрицу (растровую сетку) из точек (пикселей), каждая из которых имеют свой цвет. Так как каждый пиксель очень мал, человеческий глаз видит не отдельные элементы, а целое изображение.

В векторной графике изображения описываются с помощью элементарных геометрических объектов, называемых примитивами [4]. Фигуры можно построить использованием контрольных точек, соединенных кривыми Безье.

Алгоритмы, реализуемые в работе, предназначены для работы с растровыми изображениями (фото).

Существует множество методов обработки изображений. Среди них [5]:

- геометрические преобразования (вращение и масштабирование);
- сравнение изображений;
- редактирование и ретуширование;
- интерполяция и сглаживание;
- цветовая коррекция.

Необходимо, чтобы примеряемая вещь подстраивалась под фигуру потенциального покупателя, растягиваясь в нужных местах – так ведет себя

обычная одежда. Другими словами, в разработке следует использовать расширенное геометрическое преобразование – масштабирование.

Фигура пользователя строится на основе ключевых точек, которые задает он сам. Получив эти точки, алгоритм узнает, где и как именно следует растянуть вещь.

Масштабирование – один из наиболее частых приемов в редактировании изображений. Размер изображения изменяется использованием математических алгоритмов, которые находят цвет нового пикселя, исходя из цвета пикселей оригинала. При этом при уменьшении изображений теряется резкость, а при увеличении — детализация. Это неизбежный эффект, который можно минимизировать использованием более подходящего алгоритма.

Выделяют однокадровые и многокадровые методы [6].

К первым относятся:

1) линейные методы:
   a) метод ближайшего соседа;
   b) билинейная интерполяция;
   c) бикубическая интерполяция;
2) адаптивные нелинейные методы.

Многокадровые методы включают:

– итерационные методы суперразрешения;
– прямые быстрые методы суперразрешения.

Любой линейный метод представляет собой свертку [6]:

$$f(x) = \sum_{i=-\infty}^{+\infty} F(i)\,K(i-x), \qquad (1.1)$$

где $F(i)$ – яркость i-ого пикселя на изображении;

K – маска, накладываемая на изображение;

x – координата искомой точки;

пределы изменения i определяются выбранной формой и размером окрестности пикселя, к которому применяется ядро.

В двумерном случае свертка выглядит следующим образом:

$$f\ x,y\ = \sum_{i,j=-\infty}^{+\infty} F\ i,j\ K\ i-x\ \ K\ j-y\ , \qquad (1.2)$$

где $F(i,j)$ – яркость i,j-ого пикселя на изображении;

K – маска, накладываемая на изображение;

x,y – искомая точка;

пределы изменения i,j определяются выбранной формой и размером окрестности пикселя, к которому применяется ядро.

Результат масштабирования зависит от выбора ядра K (маски).

Масштабирование методом ближайшего соседа (Nearest neighbor) является самым простым и быстрым методом изменения размера изображения. Его суть заключается в том, что для каждого пикселя конечного изображения выбирается один пиксель исходного, наиболее близкий к его положению с учетом масштабирования [7]. Информация о нем используется для создания нового пикселя. При уменьшении изображения таким образом лишние пиксели отбрасываются, при увеличении – дублируются. Из-за того, что этот метод быстр и прост, он обладает малой точностью. При увеличении изображение получается пикселизированным и при уменьшении зернистым. Метод ближайшего соседа не подходит для большей части изображений, содержащих плавные переходы между отдельными участками [8]. Ядро К (маска) данного метода представлено на рисунке 1.1 [6].

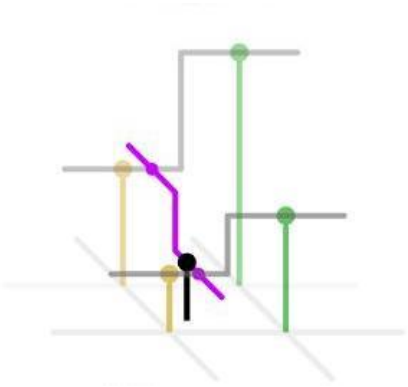


Рисунок 1.1 – Ядро К метода ближайшего соседа

Ядро интерполяции в данном методе для каждого измерения представляет собой [33]:

$$u\ s\ = \begin{matrix} 0, & s\ > 0.5 \\ 1, & s\ < 0.5 \end{matrix} \tag{1.3}$$

где s – расстояние между интерполируемой и рассматриваемой точкой сетки.

Более распространенный метод – использование билинейного или бикубического интерполирования. Он дает заметно лучший по сравнению с методом ближайшего соседа результат. Порядок интерполяции влияет на количество пикселей, задействованных в вычислении нового: в зависимости от сложности могут использоваться от 0 до 256 смежных пикселей. Вместе с тем повышение порядка интерполирования не сможет дать заметно лучший результат при изменении размера в несколько раз.

Интерполирование – это способ нахождения промежуточных значений величины по имеющемуся набору известных значений [9]. Билинейная интерполяция расширяет линейную интерполяцию для функции двух переменных. Таким образом, линейная интерполяция проводится сначала в одном направлении, а затем в перпендикулярном. Значения результирующих пикселей вычисляются как взвешенная сумма ближайших четырёх пикселей исходного изображения (при увеличении) или как взвешенная сумма группы

пикселей (при уменьшении) [10]. Рисунок 1.2 представляет ядро билинейной интерполяции.

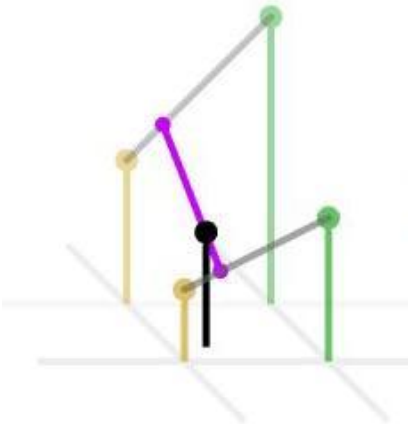


Рисунок 1.2 – Ядро К билинейной интерполяции

Ядро линейной интерполяции имеет вид [33]:

$$u\ s\ = \begin{matrix} 0, & s\ > 1 \\ 1 - s\ , & s\ < 1 \end{matrix} \tag{1.4}$$

где s – расстояние между интерполируемой и рассматриваемой точкой сетки.

Ядро билинейной интерполяции может быть выражено как два независимых ядра линейной интерполяции для двух осей.

Бикубическая интерполяция расширяет кубическую интерполяцию для функции двух переменных, значения которой заданы на двумерной регулярной сетке. Поверхность, полученная в результате бикубической интерполяции, является гладкой функцией [11], что отличает ее от результирующих поверхностей уже рассмотренных методов ближайшего соседа и билинейной интерполяции. Для поиска значения функции в искомой точке используются значения в 16 смежных точках. Рисунок 1.3 демонстрирует ядро бикубической интерполяции.

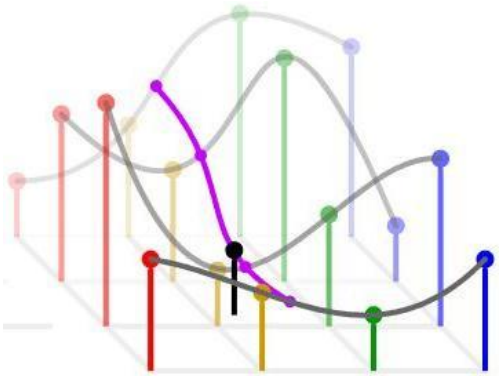


Рисунок 1.3 – Ядро К метода бикубической интерполяции

Ядро свертки для бикубической интерполяции представляет собой:

$$u\ s\ = \begin{cases} a+2\ \ s\ ^3 - \ a+3\ \ s\ ^2 + 1, & s\ \leq 1 \\ a\ s\ ^3 - 5a\ s\ ^2 + 8a\ s\ - 4a, & 1 < \ s\ < 2 \\ 0, & \text{в остальных случаях} \end{cases} \qquad (1.5)$$

где s – расстояние между интерполируемой и рассматриваемой точкой сетки,

a обычно устанавливают равным минус 0,5 или минус 0,75 [34].

Нелинейные методы призваны повышать качество изображений после применения линейных методов. В таких методах гораздо больше математических вычислений. В нелинейных методах для интерполяции вдоль и поперёк границ используются разные ядра. Вдоль границы таким образом используют интерполяцию, а поперёк нее – другие методы для достижения наилучшего результата [6].

В основе алгоритмов для виртуальной примерочной может лежать, в принципе, любой алгоритм масштабирования. Для достижения наилучшего результата следует использовать интерполяцию более высокого порядка, но так как в рамках работы алгоритм масштабирования является вспомогательным, было решено остановиться на более простой билинейной интерполяции.

1.2.2 Масштабирование методом билинейной интерполяции

Формула билинейной интерполяции интерполирует значения функции в произвольном прямоугольнике по четырем её значениям в вершинах прямоугольника и экстраполирует функцию на всю остальную поверхность [12][13]. Графическое изображение билинейной интерполяции показано на рис. 1.4.

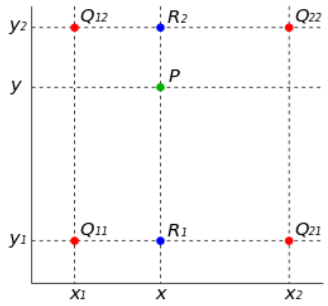


Рисунок 1.4 – Графическое изображение билинейной интерполяции

Чтобы интерполировать значение функции в точке P, сначала определяются значения функции в окружающих точках: Q_{11}, Q_{12}, Q_{21}, Q_{11}.

$$\begin{aligned} Q_{11} &= x_1, y_1 \ , \\ Q_{12} &= x_1, y_2 \ , \\ Q_{21} &= x_2, y_1 \ , \\ Q_{22} &= x_2, y_2 \ , \end{aligned} \tag{1.6}$$

где x_i, y_i – координаты смежных пикселей.

Далее рассматриваются вспомогательные точки R_1 и R_2 с помощью линейной интерполяции. Подставив значения из (1.6) в формулу интерполяции, получим значения функции в этих точках (1.7).

$$\begin{aligned} f\ R_1 &\approx \frac{x_2 - x}{x_2 - x_1} \cdot f\ Q_{11} + \frac{x - x_1}{x_2 - x_1} \cdot f\ Q_{21} \ , \\ f\ R_2 &\approx \frac{x_2 - x}{x_2 - x_1} \cdot f\ Q_{12} + \frac{x - x_1}{x_2 - x_1} \cdot f\ Q_{22} \ , \end{aligned} \tag{1.7}$$

где x_i – координаты абсцисс смежных пикселей;

x – абсцисса искомого пикселя;

$f(Q_{ij})$ – значения функции в окружающих точках.

Проводя линейную интерполяцию между вспомогательными точками R_1 и R_2 можно найти значение функции в искомой точке P (1.8).

$$f\ P\ \approx \frac{y_2 - y}{y_2 - y_1} \cdot f\ R_1\ + \frac{y - y_1}{y_2 - y_1} \cdot f\ R_2 \qquad (1.8)$$

где y_i – координаты ординат смежных пикселей;

y – ордината искомого пикселя;

$f(R_i)$ – значения функции во вспомогательных точках.

Изображение в данном случае рассматривается как поверхность, цвет пикселя – как значение функции. Для цветного изображения интерполяция проводится отдельно для трех цветов.

1.2.3 Способы построения кривой по точкам

Очевидно, что в примерочной нельзя использовать алгоритм масштабирования в чистом виде. Фигура человека имеет определенную форму, и одежда деформируется согласно ее особенностям. Одежда, в свою очередь, тоже растягивается лишь до определенного предела. Необходимо сделать так, чтобы и виртуальная одежда учитывала эти моменты.

Для того, чтобы «объяснить» одежде, как именно следует облегать фигуру, было решено использовать ключевые точки. По умолчанию это точки «углов» фигуры: плечи, бедра и т.д. Для более точного построения количество ключевых точек может быть увеличено. На основании точек строится предполагаемая фигура покупателя. После этого полученные данные обрабатываются для нахождения участков, которые необходимо

деформировать. На рисунке 1.5 представлено возможное расположение ключевых точек.

Предполагается, что пользователь отмечает точки кликами мыши по своему фото. Для того, чтобы получить предполагаемую фигуру, необходимо соединить точки плавной линией. Сделать это можно различными способами.

Рисунок 1.5 – Возможное расположение ключевых точек

Линии, рисуемые компьютером, можно условно разделить на прямые и кривые. И если с рисованием прямой проблем не возникает, достаточно лишь задать первую и последнюю токи, то рисование кривой для компьютера является большей проблемой. Возможности компьютера ограничены тем, что он не может рисовать линию, не имея математической функции, описывающей ее [14]. Существует несколько способов, позволяющих создать красивые кривые на компьютере. Важно выбрать ту, которая наиболее подходит для достижения цели, поэтому начать следует с их рассмотрения.

Кривая второго порядка - это линия на плоскости, задающаяся в декартовой системе координат общим уравнением второй степени [15]:

$$F\ x,y\ = a_{11}x^2 + 2a_{12}xy + a_{22}y^2 + 2a_{10}x + 2a_{20}y + a_{00} = 0, \qquad (1.9)$$

$$a_{11}; a_{12}; a_{22} \neq 0; 0; 0$$

где a_{ij} – коэффициенты;

x,y – координаты точки.

Кривые второго порядка делятся на вырожденные и невырожденные. Примеры вырожденных кривых – вырожденный эллипс, вырожденная гипербола, вырожденная парабола, пара вещественных параллельных прямых, вещественная прямая, пара мнимых параллельных прямых. Невырожденными кривыми являются эллипс, окружность, гипербола и парабола [15]. Кривая третьего порядка отличается от кривых второго порядка возможным наличием точки перегиба, кривая первого порядка является прямой.

Такой способ построения кривой не подходит, так как им невозможно построить произвольную кривую линию по набору точек.

Кривые Безье – способ создания кривых и поверхностей любой формы, предложенный французским инженером Пьером Безье (Pierre Bézier). Кривые Безье используются в компьютерной графике, например, для рисования плавных изгибов, в CSS-анимации. Этот метод основан на использовании опорных точек, их число может быть любым, но, как правило, используют 4 точки. Если точек больше, для каждых четырех точек строят свою кривую и потом соединяют их [14]. Идея кривой Безье заключается в том, что линия проходит только через первую и последнюю точку, а остальные «притягивают» прямую с определенной силой, искривляя ее (рис. 1.6).

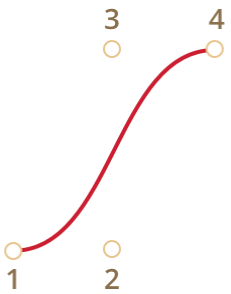


Рисунок 1.6 – Кривая Безье по четырем точкам

Сложность вычисления кривой Безье возрастает с количеством опорных точек. Формула (1.10) демонстрирует кривую Безье по четырем точкам.

$$P = (1-t)^3 P_1 + 3(1-t)^2 t P_2 + 3(1-t)t^2 P_3 + t^3 P_4 \quad (1.10)$$

где P_i – координаты *i*-й опорной точки (x_i, y_i),

t пробегает множество от 0 до 1.

Недостаток такого метода – недостаточная гладкость в местах соединений участков кривой [16]. Кроме того, необходимо, чтобы кривая проходила через каждую ключевую точку, а не только отклонялась в их направлении.

Следующий способ построить кривую – сплайны. Их часто используют в компьютерной графике. Интерполяция сплайнами геометрически представляет собой кривые, соединённые гладким образом (рис. 1.7) [16].

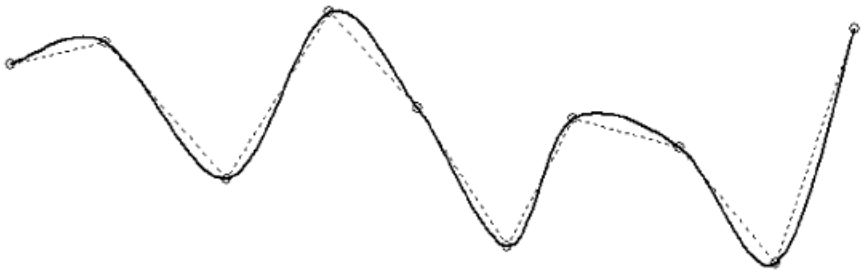


Рисунок 1.7 – Кривая, построенная сплайнами

Этот способ полностью удовлетворяет требованиям: кривая получается гладкой и охватывает любое количество ключевых точек.

1.2.4 Построение фигуры с помощью сплайнов

В общем случае математический сплайн – это кусочный полином степени *K* с непрерывной производной степени *K-1* в точках соединения сегментов, которая на элементарных участках совпадает с более простыми функциями (как правило, многочленами) [17]. Кусочные сплайны из

многочленов невысокого порядка удобны для интерполяции кривых, так как не требуют больших вычислительных затрат и не вызывают численных отклонений, свойственных многочленам высокого порядка [17].

Для этой цели часто используют кубический сплайн. Кубический сплайн – это функция, которая проходит через все заданные точки и на каждом отрезке между соседними точками является кубическим полиномом. Цель интерполяции кубическими сплайнами – получить гладкую интерполирующую функцию, производная первого и второго порядка которой непрерывна как на узлах интерполяции, так и на интервалах [18].

На каждом из отрезков $[x_{k-1}, x_k]$ функция ищется в виде полинома третьей степени:

$$S_k\ x\ = a_k + b_k\ x - x_k\ + c_k\ x - x_k\ ^2 + d_k(x - x_k)^3, \qquad (1.11)$$

где a_k, b_k, c_k, d_k – коэффициенты;

x – искомая точка;

x_k – точка, принадлежащая отрезку.

При этом на каждом из отрезков коэффициенты полинома a_k, b_k, c_k, d_k разные, таким образом, задача построения полинома сводится к поиску этих коэффициентов. Коэффициенты сплайна характеризуют значения его производных в узлах интерполяции.

Полученный таким образом результат называют кубическим сплайном дефекта 1. Дефект сплайна – это разность между степенью многочлена и степенью гладкости сплайна. На практике такой метод чреват появлением ложных экстремумов (рис. 1.7). Причина этого – слишком сильное искривление, обеспечивающее гладкость интерполяционной функции.

Лучшего результата можно достичь, используя сплайн, составленный из кривых Безье. Сплайн в форме Безье – сплайн, соединенный на отрезках кривыми Безье *k*-й степени [19]. В данном случае будет использоваться

классическая кубическая кривая. Такие сплайны обладают свойством локальности влияния контрольных точек на форму кривой.

Как уже упоминалось, недостатком кривых, составленных из кривых Безье, является недостаточная гладкость на местах стыка сегментов. Чтобы добиться непрерывности и гладкости, необходимо, чтобы три точки в местах стыковки лежали на одной прямой. Например, чтобы из шести точек получить гладкую кривую, на местах стыка сегментов кривой нужно вставить дополнительную точку, через которую соединить эти сегменты. Таким образом кривая будет состоять из двух кривых, построенных по четырем точкам [16]. Если кривую нельзя разбить на целое число таких групп, последнюю точку нужно продублировать нужное количество раз.

В проектируемом алгоритме используется схожий способ. Каждый сегмент сплайна строится по четырем точкам (две – ключевые точки и две – опорные), последняя точка одного сегмента является первой точкой следующего. Так из шести точек и одной общей получается гладкая кривая (рис. 1.8). Видно, что опорные точки t_{12}, P_i и t_{21} лежат на одной прямой с заданной точкой интерполируемой последовательности. Если взять касательные к графику интерполируемой функции в точках P_i, можно избавиться от появления ложных экстремумов: кривая Безье ограничена ломаной, построенной на её опорных точках.

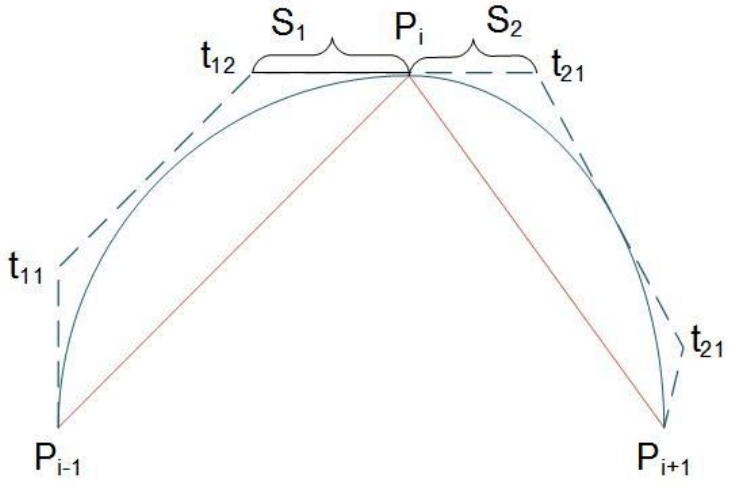


Рисунок 1.8 – Построение сплайна Безье

Задача сводится к нахождению опорных точек для кривой Безье. Это можно сделать, найдя расстояние от точки интерполируемой последовательности до искомой опорной точки (рис. 1.8). Для этого используются формулы:

$$s_1 = \begin{matrix} \frac{P_{i+1} - P_i}{2x_i}, x_i \neq 0, \\ 1, x_{k_i} = 0 \end{matrix}$$
$$s_2 = \begin{matrix} \frac{P_{i+1} - P_i}{2x_{i+1}}, x_{i+1} \neq 0, \\ 1, x_i = 0 \end{matrix} \quad (1.12)$$

где P_i – контрольная точка,

x_i – абсцисса точки,

k_i ищется по формуле (1.11).

$$k_i = \frac{P_{i+1} - P_{i-1}}{P_{i+1} - P_{i-1}} = x_i, y_i \quad (1.13)$$

Чтобы график не искривлялся на плоских отрезках, нужно проверить, совпадают ли точки P_i и P_{i+1} по ординате, и в этом случае постановить $s_1 = s_2 = 0$. Чтобы еще уменьшить кривизну, следует проверить, лежит ли абсцисса точки пересечения касательных в точках $P_i(x_i, y_i)$ и $P_{i+1}(x_{i+1}, y_{i+1})$ в отрезке $[x_i; x_{i+1}]$. В случае, если касательная в точке P_i направлена вверх, максимальное из s_1 и s_2 приравнивается к нулю, если вниз — минимальное.

После того, как сплайн построен и найдены все опорные точки, на каждом сегменте строится кривая Безье третьей степени по формуле (1.10).

Таким образом на основе любого количества ключевых точек можно построить достаточно гладкую кривую. Пример построенной фигуры представлен на рис. 1.9. При построении использовались 7 ключевых точек для каждой линии.

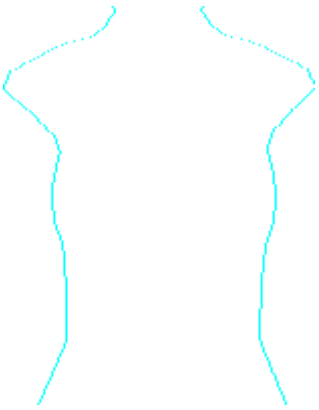


Рисунок 1.9 – Возможная фигура

1.3 Формализация требований к алгоритмам

Целью разработки алгоритмов трансформации изображений для виртуальной примерочной является облегчение выбора одежды в интернет-магазинах. Они позволят, используя свое фото, «примерить» на себя понравившуюся вещь, что частично обезопасит от неудачной покупки.

Какие из существующих приложений схожего назначения выполняют эти функции? Начать стоит с рассмотрения нескольких существующих примерочных.

Примерочная одежды VirtualDress.ru является не магазином, а каталогом вещей других магазинов. Предоставляет возможность узнать информацию о производителе и материале наряда. С помощью этого сервиса можно примерить одежду на модель или свою фотографию. Под фигуру одежда подгоняется простым масштабированием вручную.

Примерочная showroom.onvolga.com – виртуальная витрина, работающая с интернет-магазинами. Позволяет примерить вещь лишь на нескольких виртуальных моделей с модельными параметрами и внешностью, что затрудняет выбор для обладательниц нестандартных фигур. Присутствует возможность загрузить фото с веб-камеры, но преимущество от этого сомнительное: одежда не подстраивается под фигуру, а лишь вручную масштабируется. Выделяется наличием моделей-мужчин.

Примерочная styleclub.com позволяет работать с одной моделью и, соответственно, с одним типом фигуры. Представлен довольно большой каталог одежды разных брендов, позволяет сортировать одежду по бренду, цвету или магазину. После регистрации появляется возможность загрузить фото своей модели, но подогнать одежду под фигуру здесь невозможно.

Примерочная LOOKLET предоставляет возможность одеть одну из 36-ти моделей. Представлено множество одежды различных марок и аксессуаров, возможность настройки заднего фона и эффектов. Широкий выбор моделей позволит выбрать более подходящую по фигуре/внешности, но обладательницам нестандартных данных будет сложно что-то подобрать.

Таблица 1.1 – Сравнительный анализ существующих виртуальных примерочных

Критерий	VirtualDress	showroom.onvolga	styleclub	LOOKLET
Изменение одежды под фигуру	±	±	-	-
Использование манекена/модели	+	+	+	+
Загрузка своего фото	±	±	±	-

Приложения, удовлетворяющего главному требованию – полноценному использованию собственного фото и изменению одежды под фигуру – исследование не обнаружило. Перспективное направление – развитие виртуальных примерочных – разработчики и магазины оставили без должного внимания, поэтому функционал у большинства весьма узок. Все они используют готовые манекены или моделей. В рассмотренных приложениях все модели имеют соответственно модельные фигуры, что не позволяет пользоваться ими обладательницам нестандартных параметров. Кроме того, тип внешности модели может кардинально отличаться от внешности покупательницы, а при покупке одежды то, как она смотрится, тоже немаловажно.

Таким образом, можно сделать вывод о новизне и актуальности разработки новых алгоритмов.

Алгоритмы должны решать следующие задачи:

- получение контрольных точек;
- построение фигуры на их основе;
- сверка с «эталонной» фигурой;
- изменение одежды согласно полученным данным;
- вывод результата на экран или в файл.

2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ АЛГОРИТМОВ ТРАНСФОРМАЦИИ ИЗОБРАЖЕНИЙ

2.1 Анализ и выбор технологии разработки

Выбор технологии, с помощью которой будет вестись разработка, важен с точки зрения как непосредственно разработки, так и дальнейшего внедрения и поддержки программы. Каждый язык и стиль программирования обладает своими плюсами и минусами, что может облегчить, или, напротив, усложнить разработку конкретного проекта.

Язык C++ предлагает средства программирования высокого уровня, переносимость программ между различными платформами и операционными системами [20]. C++, сохраняя совместимость с C, вносит возможности объектно-ориентированного программирования. Считается универсальным языком для любых задач, но его применение может стать неэффективным там, где требуется получить готовый к употреблению результат в кратчайшие сроки, либо там, где невыгоден сам процедурный подход.

C# принадлежит семье языков программирования Microsoft. Содержит много оригинальных идей, при этом часть возможностей заимствованы из других языков и технологий. Является чисто объектно-ориентированным языком и отражает возможности и особенности .NET [21]. В нем доступна библиотека классов .NET Framework Base Class Library. C# - язык компонентного программирования, основной целью которого является разработка программных компонент многократного использования (классов и интерфейсов). Позволяет разрабатывать практически любые приложения, которые связаны с Visual Studio IDE.

Java – полностью объектно-ориентированный язык, все выполняемые программой действия находятся в методах тех или иных классов. Используется для написания апплетов, приложений и серверного программного обеспечения [22]. Преимущество Java заключается в том, что написанные на нем приложения предназначены для работы на различных

платформах и не зависят от конкретного типа процессора и операционной системы. Java является основой практически для всех типов сетевых приложений и стандартом для разработки и распространения встроенных и мобильных приложений, игр, веб-контента и корпоративного программного обеспечения.

PHP – популярный язык для разработки динамических веб-сайтов. Сконструирован специально для ведения веб-разработок и его код может внедряться непосредственно в HTML [23]. PHP является языком сценариев на стороне сервера: код обрабатывается на сервере, а конечный результат пользователь получает в виде HTML. Является открытым языком разработки.

JavaScript является скриптовым языком, предназначенным для создания веб-приложений. Скрипты на этом языке выполняются сразу после загрузки страницы [24]. JavaScript позволяет избежать чрезмерной нагрузки на сервер, выполняя часть команд на компьютере конечного пользователя. Также это положительно влияет на скорость работы приложения.

Так как алгоритм в дальнейшем должен внедряться в веб, следует выбрать язык, имеющий такой инструмент. Также желательно, чтобы язык был прост и удобен для разработки в кратчайшие сроки. Одним из критериев выбора при прочих равных станет собственное владение языком. Недостатком C++ является сложность, кроме того он не предназначен для работы с веб. PHP и JavaScript предназначены для работы непосредственно с сайтами. C# и Java удовлетворяют заявленным требованиям, но уровень владения позволил сделать выбор в пользу Java. Java позволяет создавать как обычные приложения, так и работать с веб. Готовый алгоритм нужно будет просто интегрировать в приложение примерочной.

2.2 Разработка алгоритмов

В результате выполнения выпускной квалификационной работы с использованием языка Java и среды разработки Eclipse были разработаны алгоритмы для виртуальной примерочной, обладающие характеристиками:

– получение ключевых точек и их обработка;

– построение фигуры на основании этих точек;

– сверка фигуры с эталоном и ее обработка;

– масштабирование изображения одежды с учетом особенностей полученной фигуры.

Для каждого типа одежды используется свой алгоритм со своими тонкостями. Например, обтягивающую одежду он утягивает по фигуре, а свободную оставляет свободной. Все эти алгоритмы используют один метод построения фигуры, но различие заключается в способе деформации одежды. Так как в сути своей они схожи, для простоты в дальнейшем будет использоваться слово «алгоритм».

Алгоритм включает в себя ряд классов, выполняющих свои задачи. UML диаграммы классов приведены в приложении А.

Программа состоит из следующих классов:

– Algorithms – класс, реализующий алгоритмы (полный код класса приведен в Приложении Б);

– SplineBezier, который строит сплайн Безье по ключевым точкам;

– DataProcessing обрабатывает полученные данные. В этом классе строится фигура клиента;

– BilinearInterpolation получает данные фигуры, данные об эталоне и обрабатывает одежду с учетом этих данных;

– LinearEquation используется в случае, если часть фигуры необходимо построить обычными прямыми.

– Clothes хранит данные об эталоне.

Программа, демонстрирующая алгоритм, состоит из двух вкладок. Во вкладке Catalogue можно выбрать одежду, которую хочется примерить (рисунок 2.1). В соседней вкладке View можно увидеть результат, эта вкладка показана на рисунке 2.2.

Рисунок 2.1 – Окно вкладки Catalogue

Рисунок 2.2 – Окно вкладки View с возможной загруженной фигурой

Ключевые точки задаются либо кликами мыши, либо предзадаются в виде массивов координат. Метод класса Algorithms figure_building() получает значения ключевых точек и отправляет на обработку (рис. 2.3).

```
//очистить фигуру
DataProcessing.figure_cleaning(coordS,coorRight,coorLeft,hsum);
DataProcessing.figure_updateLine(coordS, temp, averages[2],averages[3]);
coordS[1]=DataProcessing.figure_extract(coordS[1],averages[3]);
//установить значения x
coordS[0]= DataProcessing.set_x(coordS[0]);
//получить ширины фигуры
coordS[3]=DataProcessing.figure_lengths(coordS[1], coordS[2]);
```

Рисунок 2.3 – Фрагмент метода figure_building()

Полученный массив точек X разделяется на правую и левую половины для построения фигуры. Массив точек Y усредняется и обрезается, чтобы получить одинаковое расстояние между парами ключевых точек. После этого массивы передаются классу SplineBezier для построения сплайна. Из-за округления в полученном сплайне могут появиться дубли и пробелы, избавление от которых происходит в методе figure_cleaning() класса DataProcessing. Методом does_figure_cut() проверяется, нужно ли обрезать фигуру, и при необходимости обрезается методом figure_cut(). Метод limit_stretch() сравнивает фигуру с эталоном: если клиент стройный, а одежда свободная, она должна сохранить форму. В случае, если клиент полнее эталона, одежда может растянуться, но лишь до определенного предела. Это также проверяется (рис. 2.4-2.5).

```
for(int i=0;i<ideal_pheisX.length;i++)
{
    //часть фигуры больше предела растяжения? - задать максимальное значение
    if(temp[i]>ideal_pheisX[i]*koeff) temp[i]=ideal_pheisX[i]*koeff;
    //часть фигуры больше эталона, но меньше предела? - оставить как есть
    else if(temp[i]>ideal_pheisX[i]) continue;
    //часть фигуры меньше эталона? - расширить до эталона
    else temp[i]=ideal_pheisX[i];
}
```

Рисунок 2.4 – Фрагмент функции limit_stretch()

Рисунок 2.5 – Возможная фигура до и после сверки с эталоном

Построенная фигура несимметрична: так как ключевые точки отмечаются кликами мыши, такая асимметрия неизбежна. После сравнения с эталоном построенная фигура изменила свою форму.

Иногда бывает необходимо построить часть фигуры прямыми линиями. Чтобы на этих отрезках не проявлялось ненужное искривление, используются методы класса LinearEquation, которые строят прямые, решая уравнения с угловым коэффициентом.

Класс SplineBezier строит сплайн по ключевым точкам. Получив массив точек, строит сплайн, для каждого сегмента которого находит и вычисляет кривую Безье (рис. 2.6-2.7).

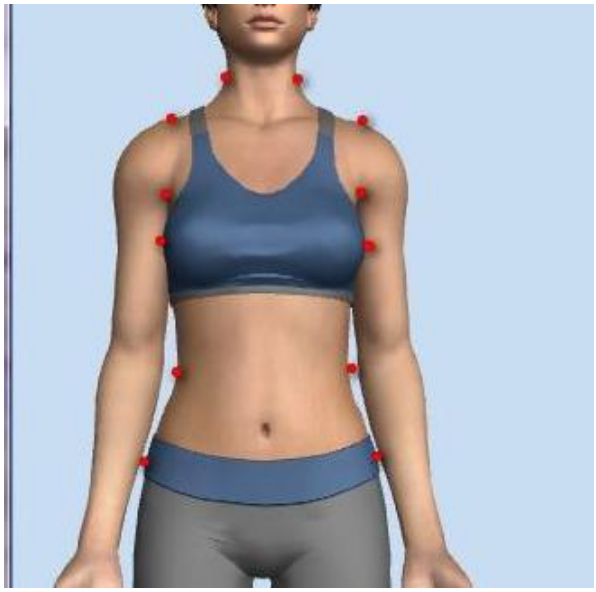


Рисунок 2.6 – Ключевые точки

Рисунок 2.7 – Построение фигуры с помощью сплайна Безье и уравнения с угловым коэффициентом

После того, как фигура построена, в зависимости от типа выбранной одежды используется один из разработанных алгоритмов.

Для обтягивающей одежды построенная фигура и выбранная одежда сразу отправляется классу BilinearInterpolation, где одежда будет обработана по фигуре (рис. 2.8). На каждом участке результирующее изображение имеет свою ширину, зависящую от данных фигуры – «динамическое» масштабирование.

```
//получить значения в окружающих точках
a = pixels[index] ;
b = pixels[index+1] ;
c = pixels[index+w] ;
d = pixels[index+w+1] ;
// blue
blue = (a&0xff)*(1-x_diff)*(1-y_diff) + (b&0xff)*(x_diff)*(1-y_diff) +
       (c&0xff)*(y_diff)*(1-x_diff)   + (d&0xff)*(x_diff*y_diff);
// green
green = ((a>>8)&0xff)*(1-x_diff)*(1-y_diff) + ((b>>8)&0xff)*(x_diff)*(1-y_diff) +
        ((c>>8)&0xff)*(y_diff)*(1-x_diff)   + ((d>>8)&0xff)*(x_diff*y_diff);
// red
red = ((a>>16)&0xff)*(1-x_diff)*(1-y_diff) + ((b>>16)&0xff)*(x_diff)*(1-y_diff) +
      ((c>>16)&0xff)*(y_diff)*(1-x_diff)   + ((d>>16)&0xff)*(x_diff*y_diff);
```

Рисунок 2.8 – Фрагмент функции масштабирования

Для свободной одежды и платьев используется сочетание пропорционального масштабирования и динамического. Сначала трансформируется облегающая часть вещи, а затем – свободная. Во втором

варианте алгоритма порядок трансформации другой. В случае платья отдельно обрабатывается верх и юбка (рис. 2.9). Алгоритм не растягивает искусственно вещь, высота ищется пропорционально.

```
img_result=new BufferedImage(imWeibot,imHeibot+imHeitop, BufferedImage.TYPE_INT_ARGB);
BilinearInterpolation.stretch_bilinear(pix, wei, hei,
        imWeitop, imHeitop,
        coordS[3], coordS[1],
        img_result,
        imWeibot,imHeibot+imHeitop,sdvig);
BilinearInterpolation.biT(pix2, wei2, hei2,
        imWeibot,imHeibot,
        img_result,
        imWeibot,imHeibot+imHeitop,hsum);
```

Рисунок 2.9 – Фрагмент комбинированного алгоритма «динамическое – пропорциональное»

На рисунке 2.10 представлен возможный результат работы: примерка платья. Алгоритм «утянул» верхнюю часть и оставил юбку неизменной: так как она должна быть свободной, ее не нужно дополнительно обрабатывать.

Рисунок 2.10 – Результат работы

Эти алгоритмы можно использовать для трансформации одежды в профиль и со спины: достаточно загрузить фото клиента с нужного ракурса и

с помощью новых ключевых точек «примерить» вещь, которая также должна быть сфотографирована сбоку или сзади. Аналогично, трансформируя части изображения по-разному и сочетая существующие алгоритмы, можно получить результат для любого типа одежды.

3 ТЕСТИРОВАНИЕ И АНАЛИЗ РАЗРАБОТАННЫХ АЛГОРИТМОВ ТРАНСФОРМАЦИИ ИЗОБРАЖЕНИЙ

3.1 Описание технологий тестирования

Тестирование программного обеспечения – важнейший этап в разработке любого ПО. Целью тестирования является проверка возможностей ПО, соответствие ожидаемым результатам. Тестирование должно проводиться на каждом этапе разработки, чтобы избежать появления еще больших ошибок. Стратегии тестирования направлены на устранение багов и ошибок в коде, обеспечение точной и оптимальной производительности программного продукта.

Выделяют две наиболее популярных технологии тестирования [25].

Тестирование методом черного ящика осуществляется без знаний внутренней работы системы. Тестер имеет доступ к ПО только через те же интерфейсы, что и пользователь. Как правило, ведётся с использованием спецификаций или иных документов, описывающих требования к системе.

Тестирование методом белого ящика учитывает внутреннее функционирование и логику работы кода. Для выполнения этого теста тестер должен иметь доступ к исходному коду программы.

Тестирование методом серого ящика – это среднее между тестированием черного и белого ящика. Тестер имеет доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется [25]. Тестирование проводится конечным пользователем.

Тестирование методом белого ящика проводилось во время разработки: проверялась работа с различными входными данными и выявленные ошибки учитывались и исправлялись. После этого нужно протестировать также методом черного ящика. Другими словами, проверить работу алгоритмов на разных типах фигур с разной одеждой.

3.2 Испытание алгоритмов с разными изображениями и фигурами

Для теста использовались 3 типа фигур: обычная и две нестандартных (рис. 3.1). Для простоты в дальнейшем будут использоваться обозначения «обычная», «полная» и «худая» фигура.

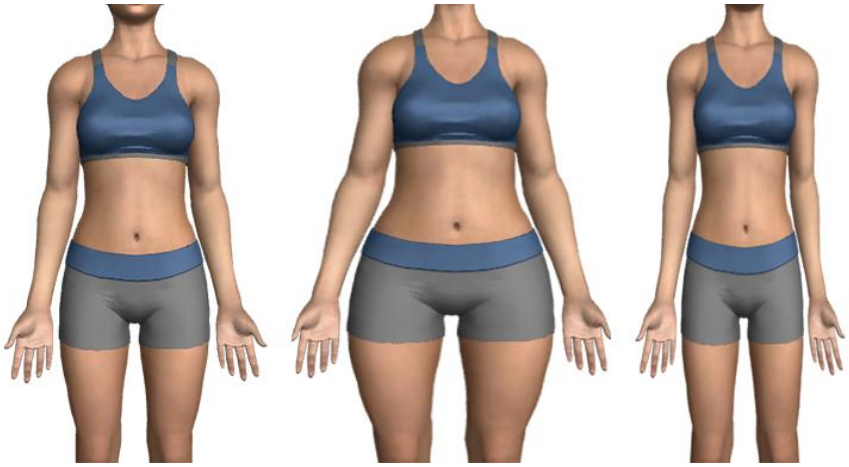


Рисунок 3.1 – Фигуры, использованные в тесте

И три типа одежды: обтягивающая майка, свободная блузка и платье (рис. 3.2). Дальше будут использоваться обозначения «обтягивающая», «свободная» одежда и платье. Тестирование не будет учитывать размеры вещи, они будут подбираться автоматически исходя из посадки на плечах и груди. Несмотря на то, что одна и та же вещь разного размера будет выглядеть немного иначе, в тестировании для наглядности использовалось одно фото для трех случаев.

Рисунок 3.2 – Одежда, использованная в тесте

На рисунке 3.3 представлены три построенные фигуры соответственно.

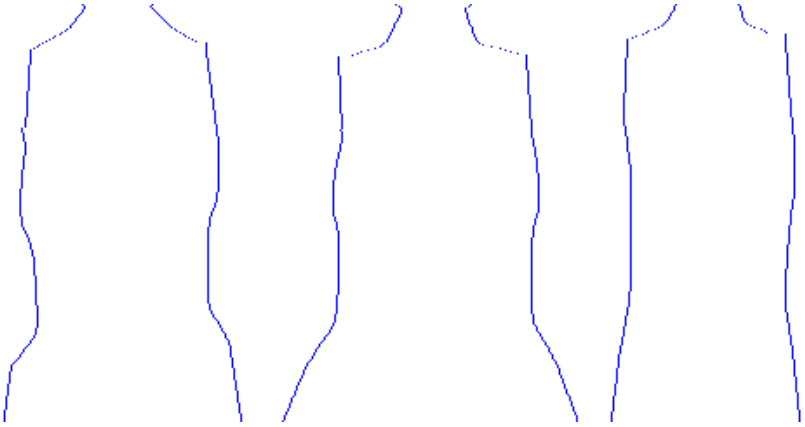


Рисунок 3.3 – Построенные фигуры трех типов

Рисунки 3.4 и 3.5 демонстрируют обтягивающую одежду на этих фигурах.

Рисунок 3.4 – Обтягивающая одежда на обычной, полной и худой фигурах

Для того, чтобы проверить качество полученного результата, было решено сравнить его с пропорциональным масштабированием, которое используется в некоторых виртуальных примерочных. Масштаб выбирался из максимально хорошей посадки на плечах и груди (рис. 3.5).

Рисунок 3.5 – Результат работы пропорционального алгоритма

На рисунке 3.4 одежда приняла форму обычной фигуры и дает достаточно ясное представление о том, что покупательница сможет с комфортом носить ее в жизни. Во втором случае одежда смогла растянуться до размеров фигуры, однако по краям видны зазоры – возможно, лучше выбрать другой размер или модель. На худой фигуре одежда не смогла «утянуться» до такой степени, чтобы облегать, и сидит довольно свободно.

Измененная пропорциональным масштабированием одежда (рис. 3.5) не дает достаточно ясного представления о том, как покупка будет выглядеть в жизни. Удовлетворительный результат такое масштабирование показало лишь на обычной фигуре, из двух других можно сделать вывод лишь о длине вещи. Непонятно, насколько растянется вещь при носке или, напротив, где будет облегать.

Следующий рисунок представляет свободную одежду на трех фигурах.

Рисунок 3.6 – Свободная одежда на обычной, полной и худой фигурах соответственно

На обычной фигуре свободная одежда смотрится так, как и должна. Во втором случае видно, что одежда растянулась до своего максимума, но так и не смогла растянуться до параметров полной фигуры. У худой фигуры узкие плечи и высокий рост, поэтому подходящий в плечах размер сидит свободно, но коротковат.

Результат, полученный после пропорционального масштабирования, показан на рисунке 3.7. Масштаб выбирался так, чтобы одежда максимально хорошо сидела на плечах и груди.

Рисунок 3.7 – Результат работы пропорционального алгоритма

Тут различие заметно меньше: одежда все равно не подходит полной фигуре, а в случае с худой результат получился почти идентичным. Тем не менее, обычной фигуре обработанная разработанным алгоритмом одежда оказалась впору, чего пропорциональное масштабирование не показало.

Следующий рисунок демонстрирует платье.

Рисунок 3.8 – Платье на обычной, полной и худой фигурах

Платье подошло всем фигурам, но на каждой оно выглядит по-своему. Например, на худой фигуре оно кажется коротковатым, хотя длина осталась той же: причина такого результата высокий рост при меньшем размере. На более низкой и полной фигуре оно выглядит немного длиннее, чем на обычной.

На следующем рисунке демонстрируется результат, полученный пропорциональным масштабированием. Как и в прошлых случаях, масштаб подбирался исходя из посадки на плечах и груди.

Рисунок 3.9 – Результат работы пропорционального алгоритма

В данном сравнении на обычной и худой фигуре достигнут достаточно близкий результат, при этом не дающий представления о том, как платье будет сидеть на груди. Для полной фигуры пропорциональное масштабирование не только исказило длину платья, но и дало ложную информацию о том, что оно не подойдет.

Таким образом, разработанные алгоритмы трансформации изображений продемонстрировали свою работоспособность. Они могут помочь предположить, как купленная в интернет-магазине и подходящая по размеру одежда будет смотреться на клиенте в жизни. Испытание на трех типах фигуры показало, что на каждой из них выбранная вещь будет смотреться по-разному: она может сесть идеально, окажется мала или велика, длинна или коротка.

Из недостатков можно выделить не всегда достаточно точное построение фигуры: хотя на трех фигурах во время тестирования критических ошибок не возникло, они могут появиться при построении более нестандартной фигуры.

Также был выявлен ряд нюансов, связанных с использованием фото одежды. Во-первых, фото должно быть в формате png с прозрачным фоном. Во-вторых, ряд эталонных ключевых точек для каждой вещи также необходимо вводить вручную. Кроме того, каждая ткань имеет свой коэффициент растяжения. Для тестирования брались случайные и более

наглядные значения. В-третьих, не учитывался размер выбранной вещи: для разных вещей размер вычислялся, исходя из идеальной посадки на груди и в плечах, но его можно искать по любой части фигуры. Если же предложить пользователю самому выбрать требуемый размер (и иметь фото для каждого), то результат может получиться другим – ведь одна и та же вещь разных размеров будет иметь разные пропорции и нюансы кроя.

В целом поставленную цель можно считать достигнутой: алгоритмы показали свою работоспособность на разных типах фигуры и разных типах одежды. Несмотря на то, что ему есть куда развиваться, он уже может выполнять свою задачу.

ЗАКЛЮЧЕНИЕ

В ходе выполнения бакалаврской работы были спроектированы и реализованы алгоритмы примерки одежды для виртуальной примерочной. В основу легли два других алгоритма: масштабирование билинейной интерполяцией и способ построения фигуры по ключевым точкам при помощи сплайнов в виде кривой Безье.

Для этого были рассмотрены и проанализированы наиболее популярные существующие методы масштабирования: метод ближайшего соседа, билинейной и бикубической интерполяции. Так как алгоритм масштабирования в данном случае является лишь «оберткой», его выбор играет меньшую роль. Поэтому выбор остановился на билинейной интерполяции как простом и дающем относительно неплохой результат алгоритме.

После этого были изучены способы построения кривой по точкам для формирования фигуры клиента: кривые различных порядков, кривые Безье, сплайны. Из рассмотренных удовлетворял требованиям только последний; для более качественного построения сегменты сплайна соединялись кривыми Безье.

На основе выбранных алгоритмов были составлены новые. Работа алгоритмов разделена на две части: сначала строится фигура с помощью сплайнов, уточняется и обрабатывается, а затем данные фигуры передаются алгоритму масштабирования для окончательной обработки изображения.

Тестирование показало, что алгоритмы могут дать более точное представление о том, как купленная вещь будет смотреться на клиенте. Использование алгоритмов, возможно, не даст гарантию удачной покупки, но поможет сделать более взвешенный выбор.

Уже сейчас видно множество путей возможного дальнейшего развития алгоритмов: сокращение количества используемых ключевых точек, усовершенствование алгоритма построения фигуры, использование более

качественных алгоритмов масштабирования, переход к построению трехмерной фигуры.

Сегодня виртуальные примерочные находятся на довольно низкой ступени развития, и отчасти это объясняется сложностью создания такого приложения, которое бы учитывало все нюансы как поведения одежды, так и построения фигуры. Со стороны клиента нежелание использовать такие примерочные продиктовано отсутствием выбора способа посмотреть желаемую вещь: одно дело смотреть на фото модели, и совсем другое – на свое собственное. Разработанные алгоритмы призваны расширить возможности современных примерочных за счет учета особенностей фигуры при примерке.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Google: There Are 129,864,880 Books in the Entire World [Электронный ресурс]. – Режим доступа: http://mashable.com/2010/08/05/number-of-books-in-the-world/#xmHxTHNa8mqx
2. IMDb Database Statistics [Электронный ресурс]. – Режим доступа: http://www.imdb.com/stats
3. Total number of Websites [Электронный ресурс]. – Режим доступа: http://www.internetlivestats.com/total-number-of-websites/
4. Растровая и векторная графика [Электронный ресурс]. – Режим доступа: http://compgraph.tpu.ru/Picture_in_PC.htm
5. Редактирование изображений [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Редактирование_изображений
6. Маркелов, К.С. Модель повышения информативности цифровых изображений на базе метода суперразрешения [Текст] / К.С. Маркелов // Инженерный вестник. – 2013. – №3. – С.525-542.
7. Трубаков, А.О. Сравнение интерполяционных методов масштабирования растровых изображений [Текст] / А.О. Трубаков, М.О. Селейкович // Научно-технический вестник Брянского государственного университета. – 2017. – №1. – С.92.
8. Каменева, А. Е. Некоторые методы интерполяции изображений [Текст] / А. Е. Каменева, А. В. Горбунова // Научное сообщество студентов : материалы X Междунар. студенч. науч.–практ. конф. – Чебоксары: ЦНС «Интерактив плюс», 2016. – С. 121–123.
9. Интерполяция [Электронный ресурс]. – Режим доступа: https://university.prognoz.ru/biu/ru/Интерполяция
10. Попов, М. А. Алгоритм повышения разрешения субпиксельно смещенных изображений [Текст] / М. А. Попов, С. А. Станкевич, С. В Шкляр // Математические машины и системы. – 2013. – №1.

11. Соловьев, Н. В. Улучшение качества сжатых изображений предварительным масштабированием [Текст] / Н. В. Соловьев, Г. В. Шифрис // Информационно-управляющие системы. – 2013. – №3.

12. Билинейная интерполяция [Электронный ресурс]. – Режим доступа: http://expert-kits.ru/PAGE_BiLinearInterpolation.php?lang=RU

13. Ликбез: методы ресайза изображений [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/243285/

14. A Primer on Bézier Curves [Электронный ресурс]. – Режим доступа: https://pomax.github.io/bezierinfo/

15. Додунова, Л.К. Кривые и поверхности второго порядка : учебно-методическое пособие [Текст] / Л.К. Додунова, Т.М. Митрякова. – Нижний Новгород: Нижегородский госуниверситет, 2013. – 38 с.

16. Построение кривых [Электронный ресурс]. – Режим доступа: http://optic.cs.nstu.ru/files/CC/CompGraph/ L5_Кривые Безье.pdf

17. Сеславин, А.И. Сплайны Безье [Текст] / А.И. Сеславин. – М.: МИИТ, 2012. – 28 с.

18. Spline interpolation [Электронный ресурс]. – Режим доступа: http://www.geos.ed.ac.uk/~yliu23/docs/lect_spline.pdf

19. Интерполяция сплайнами [Электронный ресурс]. – Режим доступа: https://fex.gongled.me/Компьютерная графика и геометрия/Лекции/PDF/08 Сплайны.pdf

20. Теория и практические основы программирования [Электронный ресурс]. – Режим доступа: http://bcoreanda.com/ShowArticle.aspx?ID=1511

21. Обзор языков программирования, предоставляемых Visual Studio 2013 [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/studies/courses/13805/1223/lecture/23397

22. Акимов, П.А. Строительная информатика [Текст] : учебное пособие / П.А. Акимов, Т.Б. Кайтуков, М.Л. Мозгалева, В.Н. Сидоров. – М.: АСВ, 2014. – 432 с.

23. Что такое PHP? [Электронный ресурс]. – Режим доступа: http://php.net/manual/ru/intro-whatis.php

24. Введение в JavaScript [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/intro

25. Стратегии тестирования [Электронный ресурс]. – Режим доступа: http://www.4stud.info/software-construction-and-testing/lecture10.html#black-box

26. Тест План (План тестирования) [Электронный ресурс]. – Режим доступа: http://www.protesting.ru/testing/plan.html

27. Digital image interpolation [Электронный ресурс]. – Режим доступа: http://www.cambridgeincolour.com/tutorials/image-interpolation.htm

28. Агафонов, В. Ю. Методы многокадровой обработки для повышения качества цифровых изображений [Текст] / В. Ю. Агафонов // Известия Волгоградского государственного технического университета. – 2015. – №2.

29. Safinaz, S. An Efficient Algorithm for Image Scaling with High Boost Filtering / S. Safinaz // International Journal of Scientific and Research Publications. – Volume 4, Issue 5, 2014.

30. Ethan E. Danahy, Sos S. Agaian, Karen A. Panetta. Algorithms for the resizing of binary and grayscale images using a logical transform [Text] / Ethan E. Danahy // Image Processing: Algorithms and Systems V – Volume 6497, Issue 6, 2014.

31. Грузман, И.С. Цифровая обработка изображений в информационных системах : научное пособие [Текст] / И.С. Грузман, В.С. Киричук, В.П. Косых, Г.И. Перетягин, А.А. Спектр. – Новосибирск: НГТУ, 2013. – 350 с.

32. Schildt, H. Java, A Beginner’s Guide Sixth Edition [Text] / Herbert Schildt. – Oracle Press, 2014. – 729 p.

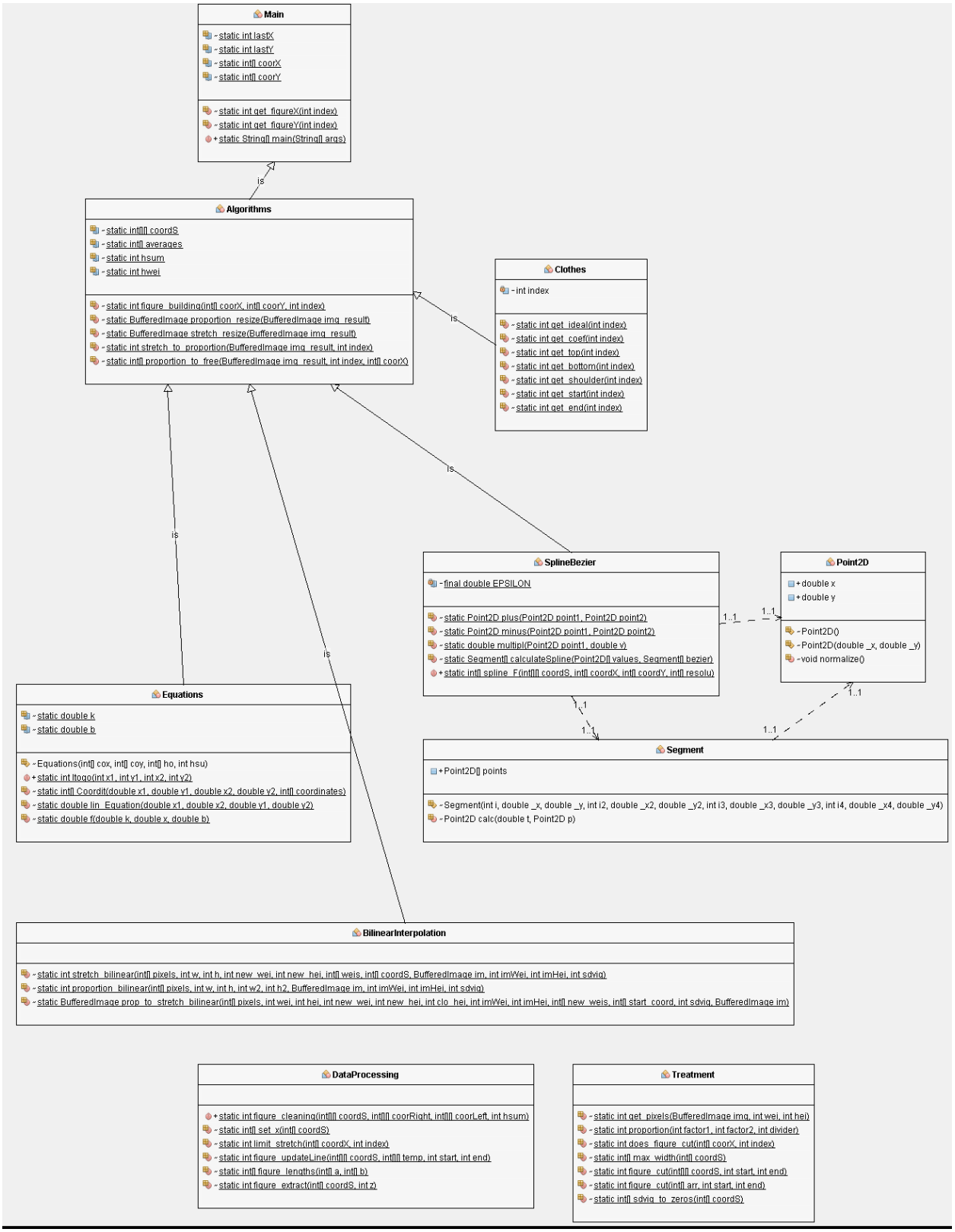
33. Interpolation and Morphing [Электронный ресурс]. – Режим доступа: http://biomachina.org/courses/imageproc/051.pdf

34. Bicubic_interpolation [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Bicubic_interpolation

35. Schumaker, L. Spline Functions: Computational Methods [Text] / Larry L. Schumaker. – Society for Industrial and Applied Mathematics, 2015. – 412 p.
36. Sederberg, T. Computer aided geometric design [Text] / Thomas W. Sederberg. – Brigham Young University, 2016. – 273 p.

Приложение А

UML диаграмма классов



Приложение Б

Код класса Algorithms

```
public class Algorithms {

static int[][]coordS;

static int[] averages;

static int hsum=0, hwei=0;

static void figure_building(int[] coorX, int[] coorY, int index){

coorX=Treatment.does_figure_cut(coorX,index);

coorY=Treatment.does_figure_cut(coorY,index);

Treatment.sdvig_to_zeros(coorX);//сдвигает в 0

Treatment.sdvig_to_zeros(coorY);

if (index!=0)      DataProcessing.limit_stretch(coorX, index);

averages=new int[coorY.length/2];

int[]      coorL=new int[coorX.length/2],

           coorR=new int[coorL.length],

           averages_distanse=new int[averages.length-1];

int        k=0;

for (int i=0; i<coorY.length;i+=2){

averages[k]=(int)(coorY[i]+coorY[i+1])/2;

k++;

}

for (int i=0; i<averages.length-1; i++)

averages_distanse[i]=averages[i+1]-averages[i];

k=0;

for (int i=0;i<coorX.length;i+=2){

coorL[k]=coorX[i];

coorR[k]=coorX[i+1];

k++;}
```

```
hsum=averages[averages.length-1]-averages[0];
int[][]          coorLeft=new int[2][hsum*2],
                 coorRight=new int[2][hsum*2]; //итоговые массивы двух сторон
SplineBezier.spline_F(coorLeft, coorL, averages,averages_distanse);
SplineBezier.spline_F(coorRight, coorR, averages,averages_distanse);
int[][] temp=new int[2][averages[3]-averages[2]];
temp[0]=Equations.Itogo(averages[2],coorX[4],averages[3],coorX[6]);
temp[1]=Equations.Itogo(averages[2],coorX[5],averages[3],coorX[7]);
coordS=new int[4][hsum];
DataProcessing.figure_cleaning(coordS,coorRight,coorLeft,hsum);
DataProcessing.figure_updateLine(coordS, temp, averages[2],averages[3]);
coordS[1]=DataProcessing.figure_extract(coordS[1],averages[3]);
coordS[0]= DataProcessing.set_x(coordS[0]);
coordS[3]=DataProcessing.figure_lengths(coordS[1], coordS[2]);
hwei=Treatment.max_width(coordS[2]);
hsum=coordS[0][coordS[1].length-1];
}
static BufferedImage proportion_resize(BufferedImage img_result) throws IOException{
BufferedImage img_clo=ImageIO.read(new File("src\\catalogue\\dress2.png"));
int wei=img_clo.getWidth(),
hei=img_clo.getHeight(),
imWei=hwei,
imHei=Treatment.proportion(imWei,hei,wei);
int[]pix=Treatment.get_pixels(img_clo, wei, hei);
img_result=new BufferedImage(imWei,imHei, BufferedImage.TYPE_INT_ARGB);
BilinearInterpolation.biT(pix, wei, hei,imWei,imHei,img_result,imWei,imHei,0);
return img_result;
}
static BufferedImage stretch_resize(BufferedImage img_result) throws IOException      {
```

```
BufferedImage img_clo=ImageIO.read(new File("src\\catalogue\\dress3.png"));
int     wei=img_clo.getWidth(),
        hei=img_clo.getHeight(),
        imWei=hwei,
        imHei=Treatment.proportion(imWei,hei,wei);
int[] pix=Treatment.get_pixels(img_clo,wei,hei);
img_result=new BufferedImage(imWei,imHei, BufferedImage.TYPE_INT_ARGB);
BilinearInterpolation.stretch_bilinear(pix, wei, hei, imWei, imHei,coordS[3], coordS[1],img_result,
imWei, imHei,0);
return img_result;
}
static BufferedImage stretch_to_proportion(BufferedImage img_result, int index) throws IOException
{
BufferedImage  imgTop=ImageIO.read(new File("src\\catalogue\\dress6.png")),
imgBott=ImageIO.read(new File("src\\catalogue\\dress7.png"));
int wei=imgTop.getWidth(),
hei=imgTop.getHeight(),
wei2=imgBott.getWidth(),
hei2=imgBott.getHeight(),
imWeitop=hwei,
imHeitop=Treatment.proportion(imWeitop,hei,wei),
imWeibot=Treatment.proportion(coordS[3][hsum-1], wei2, Clothes.get_bottom(index)),
imHeibot=Treatment.proportion(imHeitop, hei2, hei),
sdvig=Math.abs(imWeibot-imWeitop)/2;
int[] pix= Treatment.get_pixels(imgTop, wei, hei),
pix2= Treatment.get_pixels(imgBott, wei2, hei2);
img_result=new BufferedImage(imWeibot,imHeibot+imHeitop, BufferedImage.TYPE_INT_ARGB);
BilinearInterpolation.stretch_bilinear(pix, wei, hei, imWeitop, imHeitop, coordS[3], coordS[1],
img_result, imWeibot,imHeibot+imHeitop,sdvig);
```

```
BilinearInterpolation.biT(pix2, wei2, hei2,imWeibot,imHeibot, img_result,imWeibot,
imHeibot+imHeitop, hsum);

return img_result;

}

static BufferedImage proportion_to_free(BufferedImage img_result, int index, int[]coorX) throws
IOException{

BufferedImage img_clo=ImageIO.read(new File("src\\catalogue\\dress2.png"));

int       wei=img_clo.getWidth(),

hei=img_clo.getHeight(),

top_hei=averages[4],

top_wei=Treatment.proportion(wei,coorX[5]-coorX[4], Clothes.get_shoulder(index)),

imHei=Treatment.proportion(top_wei,hei,wei),

bottom_wei=Treatment.proportion(top_wei,Clothes.get_bottom(index),wei);

int[]pix=Treatment.get_pixels(img_clo, wei, hei);

if(hwei>bottom_wei)

{

int       clo_hei=Clothes.get_top(index),

imWei=0,

delta=0;

if(hwei<bottom_wei*Clothes.get_coef(index))

{

bottom_wei=Treatment.proportion(wei,hwei,Clothes.get_bottom(index));//wei/bott=x/hwei

delta=(int) (Math.round(bottom_wei-top_wei)/2);

}

else

{

double temp=Math.round(top_wei*Clothes.get_coef(index));

bottom_wei=Treatment.proportion(wei,(int)temp,Clothes.get_bottom(index));

delta=(int) (Math.round(top_wei*Clothes.get_coef(index)-top_wei)/2);
```

```
}
int[][] temp=new int[3][imHei-top_hei];
temp[0]=Equations.Itogo(0,delta,imHei,0);
temp[1]=Equations.Itogo(0,top_wei+delta,imHei,top_wei+2*delta);
temp[2]=DataProcessing.figure_lengths(temp[0],temp[1]);
imWei=Treatment.max_width(temp[2]);
img_result=new BufferedImage(imWei,imHei, BufferedImage.TYPE_INT_ARGB);
BilinearInterpolation.prop_to_stretch_Bilin(pix, wei, hei, top_wei,top_hei,clo_hei,imWei, imHei,
temp[2],temp[0],delta,img_result);
}
else
{
BilinearInterpolation.biT(pix, wei, hei, top_wei, imHei, img_result,top_wei,imHei,0);
}
return img_result;
}}
```