СОДЕРЖАНИЕ

1 ЦЕЛИ И ЗАДАЧИ ПРОЕКТА

Цели проекта:

Разработать программный инструмент для автоматического преобразования растровых изображений *PNG* формата в математически представленные кривые с последующей их параметризацией, сплайновой аппроксимацией и подготовкой данных для использования в *LaTeX* документации.

Задачи проекта:

1) Изучить структуру *PNG*-файлов и методы выделения белых областей.

2) Реализовать построение маски и обнаружение границ объектов.

3) Адаптировать и применить алгоритм Рамера-Дугласа-Пекера для сокращения точек.

4) Разработать алгоритм параметризации упрощенных контуров.

5) Реализовать построение монотонных *PCHIP*-сплайнов для координат *X* и *Y*.

6) Создать модуль преобразования сплайнов в кубические полиномы и их экспорт в *LaTeX*-формат.

7) Проанализировать качество аппроксимации на контурах разной сложности.

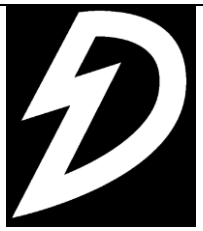
8) Проанализировать сложность алгоритмов и создать модульную архитектуру кода.

9) Проверить работу программы на изображениях с различной геометрией.

2 ОБЛАСТЬ ПРИМЕНЕНИЯ

Параметрическое представление контуров позволяет создавать плавные анимации, так как положение объекта в любой момент времени может быть вычислено аналитически. Упрощенные и параметризованные контуры требуют значительно меньше вычислительных ресурсов для рендеринга по сравнению с исходными растровыми масками или сложными полигональными сетками.

Математически представленные кривые (рисунок 1.4) способны масштабироваться до любого размера без потери качества – принцип векторной графики, что критически важно для полиграфической продукции, наружной рекламы, веб-дизайна и создания фирменного стиля.

	
Рисунок 1.1 - Исходное растровое изображение логотипа	***Рисунок 1.2 - Бинарная маска после пороговой обработки***
	
Рисунок 1.3 - Выделенный контур, построенный алгоритмом Marching Squares	***Рисунок 1.4 – Аппроксимированные PCHIP-сплайны***

3 СЛОВАРЬ ИСПОЛЬЗУЕМЫХ ТЕРМИНОВ

Epsilon (ε) – параметр точности в алгоритме Рамера-Дугласа-Пекера, задающий максимально допустимое отклонение исходных точек контура от аппроксимирующей кривой. Чем больше ε, тем сильнее упрощается контур.

Find_contours – функция библиотеки *scikit-image*, предназначенная для поиска линий уровня (контуров) на бинарном или полутоновом изображении. В проекте используется для извлечения границ белых областей маски.

LaTeX – система компьютерной верстки, широко применяемая для представления математических формул. В проекте используется для экспорта параметрических уравнений кривых в текстовом формате, пригодном для научных отчетов.

Marching squares – алгоритм поиска контуров на двумерной дискретной сетке. Используется внутри функции *find_contours* для построения ломаных линий границ бинарных областей изображения.

Matplotlib – библиотека *Python* для визуализации данных. В проекте применяется для отображения исходных контуров и аппроксимированных кривых в двумерном пространстве.

NumPy – библиотека для численных вычислений в *Python*, обеспечивающая работу с многомерными массивами. Используется для обработки координат точек, вычисления расстояний и параметризации.

PCHIP (Piecewise Cubic Hermite Interpolating Polynomial) – метод кусочно-кубической интерполяции, сохраняющий монотонность данных. В проекте применяется для построения гладких параметрических кривых без самопересечений и осцилляций.

PCHIP-сплайн – вид кусочно-кубической интерполяционной функции, который гарантирует сохранение монотонности исходных данных.

PIL (Python Imaging Library) – библиотека для загрузки и обработки изображений. В проекте используется для открытия *PNG*-файлов и доступа к значениям пикселей в формате *RGBA*.

PNG (Portable Network Graphics) – формат растровых изображений с поддержкой прозрачности и без потерь качества. Используется как входной формат для анализа изображения.

RGBA – модель представления цвета пикселя, включающая красный (*R*), зеленый (*G*), синий (*B*) и альфа-канал (*A*), отвечающий за прозрачность. Используется при анализе пикселей изображения.

Sciikit-image – библиотека *Python* для обработки изображений. В проекте применяется для поиска контуров по бинарной маске.

SciPy – научная библиотека *Python*, расширяющая возможности *NumPy*. В проекте используется модуль *scipy.interpolate* для построения *PCHIP*-сплайнов.

Алгоритм Рамера-Дугласа-Пекера (RDP) – алгоритм упрощения ломаных линий путем удаления точек, не оказывающих существенного влияния на форму контура. Позволяет уменьшить количество точек перед интерполяцией.

Альфа-канал – дополнительный канал в цветовых моделях изображения, который определяет степень прозрачности (или непрозрачности) каждого пикселя.

Аппроксимация – процесс приближенного представления дискретных данных аналитической функцией. В проекте используется для замены дискретного контура гладкой кривой.

Аппроксимированная кривая – кривая, полученная в результате аппроксимации исходного контура с помощью *PCHIP*-сплайнов. Описывается параметрическими уравнениями.

Бинарная маска – частный случай бинарного изображения, который выполняет функцию логического фильтра, указывая, какие области исходного изображения подлежат обработке, а какие игнорируются.

Бинарная область изображения – связная группа пикселей, имеющих одинаковое значение в бинарном изображении, которая представляет собой единый логический объект или часть объекта.

Бинарное изображение – цифровое изображение, каждый пиксель которого может иметь только два возможных значения: 0 (черный, фон, ложь) и 1 (белый, объект, истина).

Векторизация – преобразование растрового изображения или набора дискретных точек в аналитическое или параметрическое представление. В проекте результатом векторизации являются параметрические уравнения кривых.

Двумерное пространство – математическая модель плоскости, в которой каждая точка задается двумя координатами, обычно обозначаемыми как *(x, y)*.

Дискретизация – представление непрерывного объекта в виде набора отдельных точек. Исходный контур изображения является дискретным представлением границы объекта.

Интерполированный контур – контур, восстановленный по набору дискретных точек с помощью интерполяции, так чтобы кривая проходила точно через заданные точки.

Интерполяция – процесс построения функции, проходящей через заданные точки. В проекте используется кубическая интерполяция *PCHIP*.

Исходный контур – набор точек, полученный непосредственно из растрового изображения алгоритмом поиска контуров, до упрощения и аппроксимации.

Контур – ломаная линия, описывающая границу объекта на изображении. Представляется в виде упорядоченного списка точек в декартовой системе координат.

Коэффициенты полинома – числовые параметры, определяющие конкретный вид полинома. В проекте коэффициенты используются для записи уравнений *x(t)* и *y(t)*.

Кубический полином – полином третьей степени. Используется для описания каждого сегмента параметрической кривой.

Кубический сплайн – гладкая функция, область определения которой разбита на конечное число отрезков, на каждом из которых она совпадает с некоторым кубическим многочленом

Кусочно-гладкая функция – функция, состоящая из нескольких гладких участков, соединенных в конечном числе точек. Параметрическая кривая в проекте является кусочно-гладкой.

Кусочно-кубическая интерполяция – метод аппроксимации функции, при котором вся область разбивается на интервалы (сегменты), и на каждом таком интервале функция представляется отдельным кубическим полиномом третьей степени.

Линия уровня – кривая на поверхности или в пространстве, соединяющая точки с одинаковым значением некоторой скалярной функции.

Ложный экстремум – точка локального максимума или минимума интерполирующей функции, которая не соответствует реальному поведению исходных данных, возникает как артефакт метода интерполяции.

Маска изображения – двумерный массив, в котором каждому пикселю сопоставляется бинарное значение, указывающее принадлежность пикселя к объекту. Используется для выделения белых областей изображения.

Многомерный массив – структура данных, организованная в виде таблицы с более чем одним измерением, где каждый элемент идентифицируется набором индексов.

Монотонная интерполяция – тип интерполяции, сохраняющий порядок значений данных и предотвращающий появление ложных экстремумов. Реализуется методом *PCHIP*.

Монотонность данных – математическое свойство последовательности значений, при котором они не меняют направление изменения: либо только возрастают, либо только убывают.

Нормированный параметр t – параметр, изменяющийся в диапазоне от 0 до 1, используемый для параметризации контура по длине дуги.

Осцилляция – нежелательное колебательное поведение интерполирующей функции, проявляющееся в виде «волн» или «зигзагов» между узловыми точками, которых нет в исходных данных.

Параметризации контура по длине дуги – метод задания положения точек на кривой с помощью параметра, который пропорционален пройденному расстоянию вдоль кривой от начальной точки.

Параметризация – задание кривой в виде функций координат от одного параметра. В проекте используется параметризация по нормированной длине контура.

Параметрическое уравнение кривой – представление кривой в виде системы уравнений: $x = x(t), y = y(t)$. Именно в этом виде кривые экспортируются в *LaTeX*.

Пиксель – минимальный элемент растрового изображения, содержащий информацию о цвете и прозрачности.

Полином – математическое выражение, представляющее собой сумму одночленов, где каждый одночлен состоит из константы (коэффициента), умноженной на переменную в целой неотрицательной степени.

Полутоновое изображение – цифровое изображение, в котором каждый пиксель представлен одним числом, кодирующим его яркость в диапазоне от черного до белого.

Пороговая обработка – метод сегментации изображения, при котором пиксели классифицируются на основе сравнения их значений с заданным порогом. Используется для выделения белых областей.

Растровое изображение – изображение, представленное в виде сетки пикселей. Исходные данные проекта являются растровыми.

Самопересечение кривой – ситуация, когда кривая пересекает саму себя, образуя петли, узлы или точки пересечения.

Сегментация – процесс разделения изображения на области с общими свойствами. В проекте сегментация применяется для выделения объекта на фоне.

Скалярная функция – функция, которая каждой точке из области определения ставит в соответствие одно число (скаляр), в отличие от векторной функции, которая возвращает вектор.

Сплайн – гладкая аппроксимирующая кривая, заданная кусочно-полиномиальными функциями, которые стыкуются в узловых точках с определенной степенью гладкости.

Узловая точка – точка стыковки сегментов в кусочно-заданной функции, где происходит соединение отдельных полиномиальных кусков.

Упрощение контура – процесс уменьшения количества точек контура при сохранении его общей формы. Выполняется с помощью алгоритма *RDP*.

4 ФОРМАТ И СТРУКТУРА PNG-ФАЙЛОВ

PNG (Portable Network Graphics) является растровым форматом изображений, который поддерживает сжатие без потерь, а также использование альфа-канала для задания прозрачности пикселей. В рамках проекта *PNG*-файлы используются в качестве входных данных, содержащих графические объекты, которые подлежат дальнейшей обработке и преобразованию в параметрические кривые.

Каждый *PNG*-файл состоит из набора пикселей, каждый из которых представлен в цветовой модели *RGBA*, где компоненты *R*, *G* и *B* определяют цвет, а альфа-канал (*A*) задает степень непрозрачности. Это позволяет работать как с полностью непрозрачными областями, так и с частично прозрачными, что важно при выделении необходимых объектов на фоне.

Структурно *PNG* организован как последовательность блоков, содержащих информацию о размере изображения, глубине цвета, метаданных и непосредственно пиксельных данных. Для обработки в программе используется библиотека *PIL (Python Imaging Library)*, которая обеспечивает загрузку изображения, доступ к массиву пикселей и их значениям в формате, пригодном для дальнейшего анализа.

5 МЕТОДЫ СЕГМЕНТАЦИИ БЕЛЫХ ОБЛАСТЕЙ НА ИЗОБРАЖЕНИИ

Основным методом сегментации является пороговая обработка – процесс классификации пикселей на основе их яркости и прозрачности. В проекте используется критерий, согласно которому пиксель считается принадлежащим белой области, если его красная, зеленая и синяя компоненты превышают значение 240, а альфа-канал – 200. Это позволяет выделять только яркие, малопрозрачные участки изображения, соответствующие целевому объекту.

Результатом пороговой обработки является бинарное изображение – матрица, в которой каждый пиксель имеет значение 0 (черный, фон) или 1 (белый, объект). Эта матрица выполняет функцию бинарной маски, указывающей, какие части исходного изображения следует обрабатывать. Бинарная маска служит основой для построения ломаных линий, описывающих границы выделенных областей.

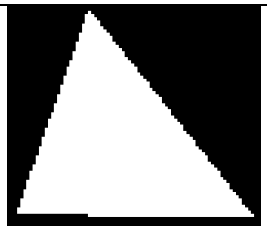
6 АЛГОРИТМ FIND_CONTOURS БИБЛИОТЕКИ SCIKIT-IMAGE

Алгоритм *Marching Squares* последовательно обрабатывает квадратные ячейки размером 2 на 2 пикселя в бинарной маске. Для каждой ячейки на основе значений четырех угловых пикселей определяется конкретный паттерн. В зависимости от паттерна внутри ячейки вычисляются и добавляются точки, через которые должна пройти линия контура. Эти точки затем соединяются, формируя непрерывные ломаные линии – контуры. Результатом работы функции *find_contours* является список контуров, где каждый контур представлен в виде упорядоченного массива точек в двумерном пространстве с координатами *(x, y)*. Эти точки соответствуют пиксельным координатам на исходном изображении и точно описывают границу выделенной белой области.

Важным свойством функции является то, что она возвращает замкнутые контуры для изолированных бинарных областей, что соответствует структуре типичных логотипов или графических объектов.

В качестве примера рассмотрим обработку *PNG*-изображения с белым треугольником на прозрачном фоне (рисунок 2.1). После применения пороговой обработки создается бинарная маска (рисунок 2.2), где область треугольника имеет значение 1, а фон – 0. Функция *find_contours* с использованием алгоритма *Marching Squares* анализирует эту маску и строит замкнутый контур, точно соответствующий границе фигуры. Результатом

является упорядоченный массив точек с координатами *(x, y)*, образующий ломаную линию, которая проходит через вершины треугольника (рисунок 2.3).

		
Рисунок 2.1 - Исходное растровое изображение треугольника	***Рисунок 2.2 - Бинарная маска после пороговой обработки***	***Рисунок 2.3 - Выделенный контур, построенный алгоритмом Marching Squares***

7 ПРЕОБРАЗОВАНИЕ КООРДИНАТ КОНТУРОВ

Координаты точек контура, полученные с помощью функции *find_contours*, изначально заданы в системе координат растрового изображения, где начало отсчета находится в левом верхнем углу, ось *X* направлена вправо, а ось *Y* – вниз. Такое представление является стандартным для компьютерной графики, но не всегда удобно для математической обработки и визуализации, особенно при подготовке данных для построения графиков или использования в *LaTeX*-документации, где традиционно используется декартова система с осью *Y*, направленной вверх.

Поэтому необходимо выполнить преобразование координат. Выполняется инверсия оси *Y*, чтобы привести направление оси к привычному математическому виду. Это достигается вычитанием каждой *y* координаты из высоты изображения.

После преобразования контур сохраняет свою геометрическую форму, но его представление становится более естественным для алгоритмов аппроксимации и визуального сравнения. Входными данными для этого этапа являются массив точек исходного контура, а выходными – массив точек с

преобразованными координатами, готовый для этапа упрощения алгоритмом Рамера-Дугласа-Пекера.

8 РЕАЛИЗАЦИЯ АЛГОРИТМА РАМЕРА-ДУГЛАСА-ПЕКЕРА

Для упрощения контуров, полученных после сегментации изображения и преобразования координат, применяется алгоритм Рамера-Дугласа-Пекера. Этот алгоритм позволяет значительно сократить количество точек, описывающих контур, при этом сохраняя его общую геометрическую форму. Основной принцип работы алгоритма заключается в рекурсивном удалении тех точек, которые оказывают наименьшее влияние на форму ломаной линии.

Алгоритм принимает на вход упорядоченный набор точек исходного контура и параметр *epsilon (ε)*, который задает максимально допустимое расстояние отклонения точек от аппроксимирующей кривой.

Процесс начинается с построения отрезка между начальной и конечной точками контура. Затем для всех промежуточных точек вычисляется перпендикулярное расстояние до этого отрезка. Если найденная точка с максимальным отклонением превышает заданное значение ε, то она считается значимой и разделяет контур на две части. Алгоритм рекурсивно применяется к каждой из полученных частей до тех пор, пока все оставшиеся точки не будут находиться в пределах допуска ε. В результате формируется новый упрощенный контур, состоящий только из ключевых точек, которые наилучшим образом сохраняют исходную геометрию.

Важной особенностью алгоритма является его способность адаптивно упрощать контуры в зависимости от их сложности. На участках с плавными изгибами и малыми отклонениями будет удалено больше точек, тогда как в областях с резкими изменениями направления, например, вершины углов или мелкие детали, точки сохраняются. Это делает алгоритм особенно полезным

для обработки контуров сложных графических объектов, где важно сохранить характерные черты формы.

Параметр ε используется для сохранения баланса между степенью упрощения и точностью аппроксимации. Чем меньше значение ε, тем ближе упрощенный контур к исходному, но тем больше точек остается. И наоборот, увеличение ε приводит к более агрессивному упрощению и уменьшению количества точек, что может быть полезно для снижения вычислительной нагрузки на последующих этапах, например, при построении сплайнов. В проекте значение ε подбирается эмпирически, исходя из требований к точности и сложности обрабатываемых изображений. В проекте значение $\varepsilon = 5$ является оптимальным, так как при большем – более сложные логотипы, в отличие от треугольника, теряют характерные черты формы.

В визуальном представлении можно наблюдать, как исходный контур, состоящий из множества точек, преобразуется в более простую ломаную линию с сохранением основных геометрических признаков. Например, для треугольника (рисунок 2.1) алгоритм оставляет только три вершины (рисунок 3.1), полностью отбрасывая промежуточные точки на прямолинейных сторонах.

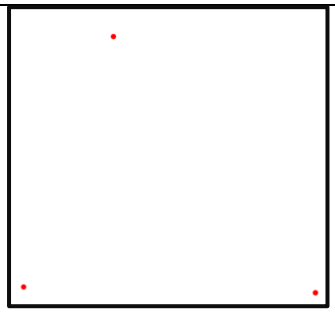
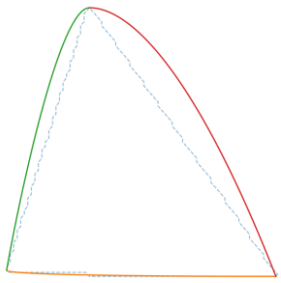
	

Рисунок 3.1 – Характерные точки треугольника после применения алгоритма RDP	***Рисунок 3.2 – Соединение полученных точек полиномами третьей степени***

После применения алгоритма Рамера-Дугласа-Пекера упрощенный контур проходит дополнительную проверку на замкнутость и минимальное

количество точек (В данном проекте этот параметр равен 1). Это необходимо для обеспечения корректной работы последующих этапов параметризации и аппроксимации. В случае если количество точек после упрощения оказывается недостаточным, выполняется дополнительная обработка для гарантии того, что контур сохраняет свою целостность и пригоден для дальнейшего математического описания.

9 ПАРАМЕТРИЗАЦИЯ УПРОЩЕННЫХ КОНТУРОВ

После применения алгоритма Рамера-Дугласа-Пекера мы получаем упрощенный контур, состоящий из характерных точек, которые сохраняют общую форму исходного объекта. Однако для дальнейшей математической обработки, такой как построение гладких кривых или экспорт уравнений кривых в *LaTeX*, необходимо перейти от дискретного набора точек к непрерывному параметрическому представлению. Этот процесс называется параметризацией контура.

Основная идея параметризации заключается в том, чтобы сопоставить каждой точке контура некоторый числовой параметр, обычно обозначаемый как t, который изменяется монотонно вдоль всей кривой. В данном проекте используется параметризация по нормированной длине дуги. Это означает, что параметр t изменяется от 0 до 1, где $t = 0$ соответствует начальной точке контура, а $t = 1$ – конечной, причем изменение t пропорционально пройденному расстоянию вдоль контура. Такой подход обеспечивает равномерное распределение параметра вдоль кривой, что важно для последующей интерполяции.

Процесс параметризации начинается с вычисления кумулятивных длин отрезков между соседними точками упрощенного контура. Для каждой точки определяется накопленное расстояние от начала контура, которое затем делится на общую длину контура. В результате получается массив значений t,

каждое из которых соответствует конкретной точке. Например, для треугольника после алгоритма *RDP* остаются три вершины (рисунок 3.1). Расстояния между ними суммируются, и каждой вершине присваивается нормированный параметр (первой точке присваивается значение 0, так как от нее начинается отсчет расстояния). Таким образом, контур превращается в две параметрические последовательности: *x(t)* и *y(t)*, где *t* пробегает значения от 0 до 1.

Параметризация по длине дуги аналогична движению автомобиля вдоль извилистой дороги: одометр показывает пройденное расстояние (параметр *t*), а координаты автомобиля *(x, y)* в каждый момент времени определяются его положением на карте. Такой способ гарантирует, что параметр *t* меняется плавно и предсказуемо, без скачков, что важно для построения интерполяционных сплайнов.

Важно отметить, что параметризация выполняется после упрощения контура алгоритмом *RDP*, что сокращает количество точек и, как следствие, уменьшает вычислительную сложность последующей интерполяции без потери основных геометрических особенностей объекта.

10 ПРИНЦИП РАБОТЫ PCHIP-ИНТЕРПОЛЯЦИИ

После параметризации контур готов к этапу аппроксимации *PCHIP*-сплайнами. Дискретные последовательности *x(t)* и *y(t)* используются в качестве опорных точек, через которые будет проходить гладкая параметрическая кривая.

PCHIP (Piecewise Cubic Hermite Interpolating Polynomial) – это метод кусочно-кубической интерполяции, который в отличие от классических кубических сплайнов, гарантирует сохранение монотонности исходных данных. Это означает, что если последовательность значений *x* или *y* монотонно возрастает или убывает, то интерполирующая функция также

будет монотонна на соответствующем участке. Такой подход полностью исключает появление нежелательных осцилляций и ложных экстремумов.

Работа *PCHIP*-интерполяции в проекте организована следующим образом: для каждой координатной последовательности (*x(t)* и *y(t)*) строится свой независимый *PCHIP*-сплайн. При этом в узловых точках, соответствующих значениям параметра *t*, сплайн не только проходит точно через заданные координаты, но и имеет определенные производные, которые подбираются специальным образом для обеспечения монотонности. Этот процесс можно сравнить с плавным вождением автомобиля по извилистой дороге: водитель (алгоритм *PCHIP*) не только точно проходит через контрольные точки (города), но и плавно регулирует скорость и направление на поворотах, избегая резких рывков и «виляний» на траектории.

После построения *PCHIP*-сплайнов для *x(t)* и *y(t)* каждый сегмент кривой между соседними узловыми точками представляется в виде кубического полинома. Коэффициенты этих полиномов (для рассматриваемых полиномов – это коэффициенты *A, B, C, D* из многочлена $At^3 + At^2 + At + D$) преобразуются из локальной формы (относительно начала сегмента) в глобальную, что позволяет записать параметрические уравнения для всего контура в виде набора кусочно-заданных функций. Результатом этого этапа является гладкая, лишенная самопересечений кривая, которая точно проходит через характерные точки упрощенного контура и готова к экспорту в формате *LaTeX*.

11 ПРЕОБРАЗОВАНИЕ ЛОКАЛЬНЫХ КОЭФФИЦИЕНТОВ В ГЛОБАЛЬНЫЕ

Процесс преобразования выполняется функцией *expand_shifted_poly(coefs_u, t0)*, где *coefs_u* – массив локальных коэффициентов [*a, b, c, d*] полинома, а *t0* – значение глобального параметра в

начале сегмента. Функция реализует аналитическое преобразование полинома из локальной формы в глобальную.

На вход подаются локальные коэффициенты, извлеченные из *PCHIP*-сплайна (*cs_x.c[:, i]* и *cs_y.c[:, i]*), и значение *t0* – начало текущего сегмента в нормированной параметризации. На выходе получаются массивы глобальных коэффициентов, которые уже соответствуют полиномам от общего параметра *t*.

Результатом этого этапа является полное математическое описание кривой в форме, готовой для экспорта в *LaTeX*, построения графиков или использования в системах компьютерного моделирования.

12 СРАВНЕНИЕ С КУБИЧЕСКИМИ СПЛАЙНАМИ

В проекте для построения гладких параметрических кривых вместо классических кубических сплайнов был выбран метод *PCHIP*, который также относится к семейству кусочно-кубических интерполяционных методов, но обладает важными отличиями, делающими его более предпочтительным для аппроксимации контуров графических объектов.

Классические кубические сплайны стремятся обеспечить максимальную гладкость кривой, требуя непрерывности не только значений функции, но и ее первой и второй производных в узловых точках. Это приводит к созданию плавных, эстетически приятных кривых, которые хорошо подходят для моделирования физических траекторий или построения графиков данных. Однако при обработке контуров, особенно содержащих участки с резкими изменениями направления или близко расположенные характерные точки, кубические сплайны могут проявлять нежелательное поведение – осцилляции. Осцилляции представляют собой ложные волнообразные отклонения кривой между узловыми точками, которые не соответствуют исходной геометрии контура.

Это означает, что если исходные координаты точек вдоль параметра *t* монотонно возрастают или убывают, то интерполирующая функция будет вести себя так же, не создавая ложных экстремумов или волн. Такой подход гарантирует, что аппроксимированная кривая не будет иметь самопересечений и сохранит общую форму контура даже при наличии острых углов или быстрых изменений направления.

На рисунке 3.2 можно наблюдать результат применения *PCHIP*-интерполяции к упрощенному контуру треугольника: кривая плавно соединяет вершины, стараясь следовать ожидаемой геометрии (пунктирная линия), однако на прямых участках это сложно, потому что в проекте используются именно кубические полиномы.

Но, если взять за пример круг (рисунок 4.1), то полученные *PCHIP*-сплайны будут лежать четко на контуре (рисунок 4.2), что является важным аргументом в выборе степени полиномов.

	
Рисунок 4.1 – Исходное растровое изображение круга	***Рисунок 4.2 – Соединение полученных точек полиномами третьей степени***

Таким образом, выбор *PCHIP* вместо классических кубических сплайнов обусловлен требованием к сохранению геометрической целостности и предсказуемости формы аппроксимированной кривой. Это особенно важно в задачах векторизации геометрически сложных объектов. Хотя *PCHIP* обеспечивает несколько меньшую гладкость (разрыв второй производной в

узлах), эта потеря компенсируется повышением надежности и соответствия исходным данным, что является приоритетом в данном проекте.

13 ГЕНЕРАЦИЯ ПАРАМЕТРИЧЕСКИХ УРАВНЕНИЙ

13.1. Формирование кубических полиномов для x(t) и y(t)

Полученные полиномы объединяются в полное параметрическое описание всей кривой, которое затем экспортируется в текстовый файл в формате *LaTeX* (работа с коэффициентами описана в пункте 11). Каждый сегмент записывается с указанием интервала изменения параметра *t* и соответствующих уравнений для *x(t)* и *y(t)*. (Рисунок 5.1, рисунок 5.2).

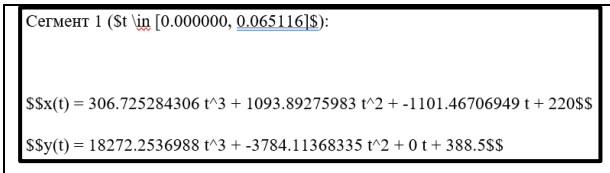


Рисунок 5.1 – Запись уравнений для Сегмента 1 в формате LaTeX

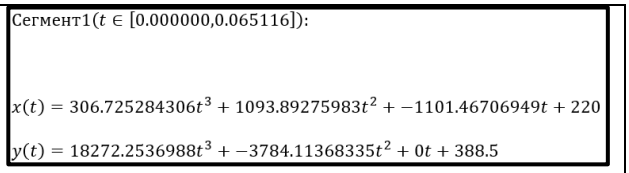


Рисунок 5.2 – Запись уравнений в привычном виде

13.2. Экспорт уравнений в формате LaTeX

После получения параметрических уравнений в виде кубических полиномов для каждого сегмента кривой выполняется их экспорт в текстовый файл, форматированный для использования в системе верстки *LaTeX*. Этот процесс реализован в функции *write_latex(all_segments, filename)*, которая принимает массив сегментов всех кривых и имя выходного файла.

Для каждой кривой в файле создается отдельная секция с заголовком. Внутри секции последовательно перечисляются все ее сегменты с указанием интервала изменения параметра *t* и соответствующих уравнений для координат *x(t)* и *y(t)*. Уравнения записываются в стандартной математической нотации *LaTeX* с использованием окружения «*$$ ... $$*», что обеспечивает их

корректное отображение в виде отдельной формулы. Коэффициенты полиномов выводятся с точностью до 9 знаков после запятой (рисунок 5.2), что гарантирует сохранение деталей геометрии при использовании уравнений в расчетах или визуализации.

14 ПОСТРОЕНИЕ ИСХОДНЫХ И АППРОКСИМИРОВАННЫХ КОНТУРОВ

Визуализация является важным этапом верификации работы разработанного программного инструмента. Для этого используется библиотека *Matplotlib*, позволяющая отображать как исходные контуры (рисунок 2.3), полученные после сегментации изображения, так и аппроксимированные кривые (рисунок 3.2), построенные с помощью *PCHIP*-сплайнов. Такой подход обеспечивает наглядное сравнение точности аппроксимации и сохранения геометрической формы объекта.

Исходный контур, представленный в виде ломаной линии, строится на основе точек, выделенных функцией *find_contours* после применения пороговой обработки и преобразования координат. Этот контур отображается пунктирной линией, что позволяет визуально оценить его дискретную природу и возможные неровности, обусловленные растровым представлением исходного изображения. Например, для логотипа сложной формы (рисунок 1.1) исходный контур будет состоять из множества мелких отрезков, точно следующих границам пикселей.

Аппроксимированный контур строится на основе параметрических уравнений, сгенерированных для каждого сегмента кривой. Для этого вычисляются значения координат $x(t)$ и $y(t)$ с использованием коэффициентов кубических полиномов на равномерной сетке значений параметра t в пределах каждого сегмента. Полученные точки соединяются сплошной линией, формируя гладкую кривую, которая должна максимально точно повторять

форму исходного контура, но без пиксельных шумов и избыточной детализации.

Наложение исходного и аппроксимированного контуров на один график (рисунок 1.4) позволяет оценить точность работы алгоритма. На участках с плавными изменениями кривизны, таких как округлые элементы логотипа, *PCHIP*-сплайн практически точно совпадает с исходными данными. На прямолинейных участках аппроксимированная кривая также стремится к прямой, однако, в силу использования кубических полиномов, могут наблюдаться незначительные отклонения, не влияющие на общее восприятие формы. Визуальный анализ подтверждает, что алгоритм успешно подавляет осцилляции и исключает самопересечения, обеспечивая геометрическую целостность результирующей кривой. Таким образом, построение и сравнение контуров служит не только для демонстрации результатов, но и для валидации выбранного метода интерполяции и параметров упрощения.

15 СТРУКТУРА ПРОГРАММНОГО КОДА

Программный инструмент реализован на языке *Python* и имеет модульную архитектуру, что обеспечивает разделение функциональных блоков и упрощает дальнейшую поддержку и расширение. Код состоит из пяти основных логических модулей: загрузка изображений, построение бинарной маски и контуров, обработка контуров, аппроксимация сплайнами и вывод результатов. Входными данными для программы является *PNG*-файл с изображением, содержащим белые объекты на прозрачном или контрастном фоне. На выходе формируются параметрические уравнения для каждого сегмента кривой в формате *LaTeX*, а также визуализация исходного и аппроксимированного контуров.

15.1. Класс GetPng: загрузка изображений

Класс *GetPng* отвечает за корректную загрузку и первичную обработку растрового изображения в формате *PNG*. Его задача обеспечить доступ к данным пикселей в цветовой модели *RGBA*, которая необходима для последующей сегментации. При инициализации класс принимает путь к файлу, а метод *open()* использует библиотеку *PIL* для открытия изображения и его приведения к формату *RGBA*. Это гарантирует, что каждый пиксель будет представлен четырьмя значениями: красным, зеленым, синим и альфа-каналом, определяющим прозрачность. Важной функцией класса является проверка существования файла, что предотвращает ошибки на раннем этапе выполнения программы. Полученный объект изображения передается далее для построения бинарной маски.

15.2. Класс ChoosePoints: построение маски и контуров

Класс *ChoosePoints* выполняет этап сегментации – преобразование загруженного изображения в бинарную маску и выделение контуров объектов. Работа класса начинается с применения пороговой обработки, реализованной в методе *is_white()*. Критерием принадлежности пикселя к белой области является одновременное превышение пороговых значений для всех цветовых каналов ($R, G, B > 240$) и альфа-канала ($A > 200$). Это позволяет отделить целевые объекты от фона, даже если фон не является абсолютно черным или полностью прозрачным. На основе этого критерия метод *build_mask()* формирует двумерный массив (бинарную маску), где единицы соответствуют объекту, а нули фону. Затем метод *detect_contours()* использует функцию *find_contours* из библиотеки *scikit-image*, которая, применяя алгоритм *Marching Squares*, находит границы белых областей и возвращает список контуров в виде упорядоченных массивов точек. Эти контуры, еще не прошедшие преобразование координат, служат основой для дальнейшей обработки.

15.3. Функции обработки контуров

Данный модуль содержит набор функций, которые подготавливают контуры к аппроксимации. Функция *process_contour()* является «центральной»: она принимает исходный контур, выполняет инверсию оси *Y* для приведения координат к математической системе, где ось *Y* направлена вверх, и обеспечивает замкнутость контура. После этого применяется алгоритм Рамера-Дугласа-Пекера (функция *rdp()*), который упрощает контур, удаляя точки, незначительно влияющие на его форму, в соответствии с параметром *RDP_EPSILON*. Это важно для снижения вычислительной сложности последующей интерполяции без потери ключевых геометрических особенностей. Далее функция выполняет параметризацию упрощенного контура по нормированной длине дуги. Для этого вычисляется кумулятивное расстояние между точками, которое затем делится на общую длину контура, в результате чего каждой точке сопоставляется параметр *t* в диапазоне от 0 до 1. На выходе функция возвращает массив упрощенных точек и соответствующий массив параметров *t*, которые служат входными данными для построения сплайнов.

15.4. Модуль аппроксимации сплайнами

Этот модуль отвечает за построение гладких параметрических кривых на основе упрощенных контуров. Основная функция *fit_piecewise_cubic()* принимает массив точек и соответствующих параметров *t*. Вместо классических кубических сплайнов здесь используется монотонная *PCHIP*-интерполяция (через *PchipInterpolator* из *SciPy*) для координат *X* и *Y* независимо. Для каждого сегмента между узловыми точками извлекаются локальные коэффициенты кубического полинома. Поскольку *PCHIP* хранит коэффициенты относительно начала сегмента ($\tau = t - t0$), вспомогательная функция *expand_shifted_poly()* преобразует их в глобальные коэффициенты для полиномов от общего параметра *t*. В результате для каждого сегмента формируется словарь, содержащий интервал изменения *t* и массивы

коэффициентов для уравнений *x(t)* и *y(t)*. Этот набор сегментов представляет собой полное математическое описание аппроксимированной кривой.

15.5. Модуль вывода результатов

Финальный модуль отвечает за экспорт полученных результатов в удобные форматы. Функция *write_latex()* преобразует массив сегментов всех кривых в текстовый файл, отформатированный для системы верстки *LaTeX*. Для каждой кривой создается отдельная секция, в которой для каждого сегмента указывается интервал параметра *t* и записываются уравнения *x(t)* и *y(t)* в виде кубических полиномов с коэффициентами, выведенными с высокой точностью (рисунок 5.2). Визуализация результатов обеспечивается функциями *plot_splines()* и *plot_rdp_points()*. Первая функция отображает исходные контуры (пунктирные линии) и построенные *PCHIP*-сплайны (сплошные линии) на одном графике, позволяя наглядно оценить качество аппроксимации (рисунок 1.4). Вторая функция визуализирует точки контура после упрощения алгоритмом *RDP*. Дополнительно функция *show_mask()* отображает бинарную маску, что полезно для отладки этапа сегментации. Все графики строятся с помощью библиотеки *Matplotlib*.

16 АНАЛИЗ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ RDP-АЛГОРИТМА

В общем случае вычислительная сложность рекурсивной реализации *RDP*-алгоритма составляет $O(n^2)$ в худшем сценарии, где n – количество точек в исходном контуре. Такой рост обусловлен тем, что на каждом шаге рекурсии выполняется поиск точки с максимальным отклонением от текущего отрезка, что требует вычисления расстояний для всех промежуточных точек. Для контуров с большим числом точек это повлечь снижение производительности, особенно при обработке сложных логотипов или детализированных графических объектов.

Однако на практике средняя сложность алгоритма часто близка к $O(n \log n)$, особенно когда контур содержит участки с плавными изменениями кривизны. Это связано с тем, что алгоритм рекурсивно делит контур на подотрезки, и на каждом уровне рекурсии обрабатывается лишь часть исходных точек. Для контуров, которые после упрощения сохраняют небольшое количество характерных точек (например, треугольник, прямоугольник), алгоритм завершает работу за малое число шагов.

Важным фактором, влияющим на сложность, является параметр точности *epsilon (ε)*, который задает максимально допустимое отклонение точек от упрощенной ломаной. Чем больше значение ε, тем агрессивнее алгоритм удаляет точки, что ускоряет его работу, но может привести к потере деталей формы. В проекте эмпирически выбрано значение $\varepsilon = 5$, которое обеспечивает баланс между степенью упрощения и сохранением геометрической целостности объектов различной сложности.

На рисунке 3.1 показан результат применения алгоритма *RDP* к контуру треугольника: из множества исходных точек остаются только три вершины, что соответствует минимально необходимому количеству для описания данной фигуры. Такой результат достигается за счет того, что все промежуточные точки на прямолинейных участках отклоняются от отрезка, соединяющего вершины, менее чем на ε, и поэтому удаляются.

В контексте проекта *RDP*-алгоритм играет роль фильтра, который подготавливает данные для последующей параметризации и *PCHIP*-интерполяции. Его применение позволяет уменьшить размер входных данных без потери смысловых геометрических особенностей, что положительно сказывается на общей производительности инструмента. Для большинства практических задач, таких как обработка логотипов, производительности реализации *RDP* достаточно, так как количество точек в контурах редко превышает несколько тысяч.

Таким образом, хотя теоретическая сложность *RDP* может достигать $O(n^2)$, на практике его использование оправдано благодаря адаптивному характеру упрощения и возможности регулировки точности через параметр ε. Это делает алгоритм эффективным инструментом предобработки контуров перед их параметризацией и аппроксимацией сплайнами.

17 АНАЛИЗ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ ПОСТРОЕНИЯ PCHIP-СПЛАЙНОВ

Построение *PCHIP*-сплайнов является важным этапом аппроксимации контуров, обеспечивающим гладкое и монотонное параметрическое представление кривых. С точки зрения вычислительной сложности, процесс формирования *PCHIP*-сплайнов является эффективным и предсказуемым, что делает его пригодным для обработки контуров различной сложности. Основная операция заключается в построении независимых интерполяторов для координатных последовательностей *x(t)* и *y(t)* с использованием класса *PchipInterpolator* из библиотеки *SciPy*.

Вычислительная сложность построения *PCHIP*-сплайнов оценивается как $O(n)$, где n – количество точек в упрощенном контуре после применения алгоритма Рамера-Дугласа-Пекера. Это связано с тем, что алгоритм выполняет однократный проход по узловым точкам для вычисления коэффициентов кубических полиномов на каждом сегменте. В отличие от классических кубических сплайнов, требующих решения систем линейных уравнений с матрицами, *PCHIP* использует локальные формулы для определения наклонов в узлах, что исключает необходимость глобальных вычислений.

Важным нюансом является преобразование локальных коэффициентов полиномов, хранящихся в форме относительно начала сегмента $(\tau = t - t0)$, в глобальные коэффициенты для параметра t. Эта операция выполняется функцией *expand_shifted_poly(coefs_u, t0)* и имеет константную сложность

O(1) для каждого сегмента, так как представляет собой аналитическое выражение, не зависящее от количества точек.

Общая сложность этапа аппроксимации сплайнами складывается из *O(n)* на построение интерполяторов и *O(m)* на преобразование коэффициентов, где *m* – количество сегментов (равное *n–1*). В результате общая асимптотика остается линейной – *O(n)*.

На практике это означает, что даже для контуров с большим количеством точек, например, для детализированных логотипов, построение *PCHIP*-сплайнов выполняется быстро и не становится узким местом в работе программы. Это заметно в контексте подготовки данных для *LaTeX*, где требуется генерация параметрических уравнений для множества сегментов кривой. Например, для контура круга (рисунок 4.1), состоящего из некоторых точек после *RDP*, *PCHIP*-аппроксимация позволяет получить гладкую кривую (рисунок 4.2) с минимальными вычислительными затратами, сохраняя при этом монотонность и отсутствие осцилляций.

18 АНАЛИЗ СУММАРНОЙ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ

Общая вычислительная сложность разработанного программного инструмента формируется из последовательного выполнения трех основных этапов: упрощения контура с помощью алгоритма Рамера-Дугласа-Пекера, параметризации упрощенного контура и построения *PCHIP*-сплайнов на основе полиномов третьей степени. Каждый из этих этапов вносит свой вклад в общую асимптотическую сложность, которая в итоге определяется наиболее затратной операцией.

Этап упрощения *RDP* в рекурсивной реализации имеет сложность $O(n^2)$ в худшем случае. Однако на практике, благодаря адаптивному характеру алгоритма и выбору параметра $\varepsilon = 5$, средняя сложность часто приближается к $O(n \log n)$, особенно для контуров с плавными участками.

Этап параметризации включает вычисление кумулятивных длин дуг и нормировку параметра *t*. Этот этап выполняется за *O(k)*, где k – количество точек после *RDP*. Поскольку k существенно меньше n после упрощения, влияние этого этапа на общую сложность незначительно.

Этап построения *PCHIP*-сплайнов, основанных на кубических полиномах, имеет линейную сложность *O(k)*. Таким образом, данный этап является эффективным и не создает вычислительно сложных мест.

Суммарная сложность всего алгоритма в худшем случае оценивается как $O(n^2)$ из-за доминирующего влияния *RDP*-алгоритма. Однако в типичных сценариях работы с графическими объектами, где контуры не являются максимально нерегулярными, общая сложность приближается к $O(n \log n)$. Это позволяет инструменту эффективно обрабатывать изображения с контурами, содержащими до нескольких тысяч точек, без потери производительности.

19 ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

19.1. Входные изображения различной геометрии

Для тестирования программного инструмента использовались растровые изображения в формате *PNG*, содержащие объекты различной геометрической формы. Каждое изображение подвергалось полному циклу обработки: загрузка, построение бинарной маски, выделение контуров, их упрощение алгоритмом Рамера-Дугласа-Пекера, параметризация и аппроксимация с помощью PCHIP-сплайнов.

Рисунок 6.1 - Исходное растровое изображение квадрата	***Рисунок 6.2 - Исходное растровое изображение кольца***
Рисунок 6.3 - Исходное растровое изображение овала	***Рисунок 6.4 – Исходное растровое изображение ромба***

19.2. Полученные изображения по этапам

Первым этапом работы программы является построение бинарной маски (рисунки 7.1-7.4). После применения пороговой обработки создается маска, в которой все пиксели, соответствующие белой области объекта, получают значение 1, а фон – значение 0.

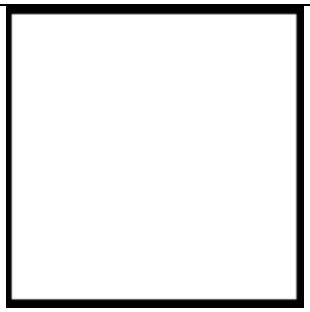
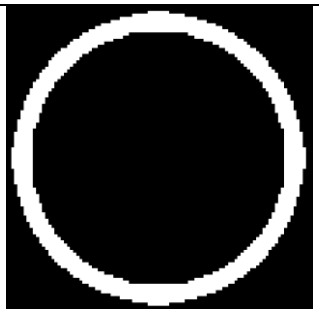
	

Рисунок 7.1 - Бинарная маска после пороговой обработки квадрата	***Рисунок 7.2 - Бинарная маска после пороговой обработки кольца***

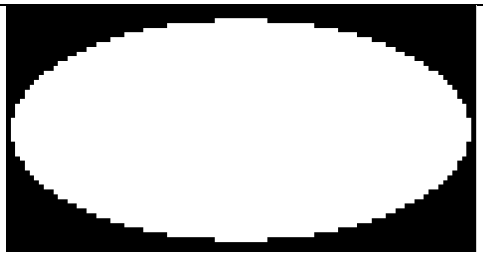


Рисунок 7.3 - Бинарная маска после пороговой обработки овала	***Рисунок 7.4 – Бинарная маска после пороговой обработки ромба***

Далее функция *find_contours* с использованием алгоритма *Marching Squares* выделяет исходный контур (рисунки 8.1-8.4), представляющий собой ломаную линию, проходящую по границам маски. Этот контур состоит из большого количества точек, точно следующих пиксельной сетке, и служит основой для дальнейшего упрощения.

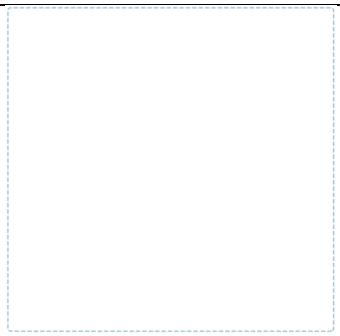
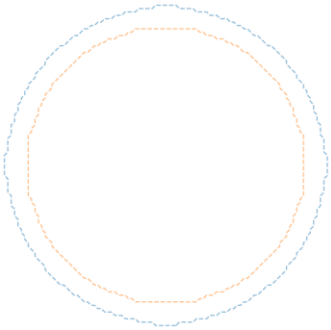
	

Рисунок 8.1 - Выделенный контур квадрата	***Рисунок 8.2 - Выделенный контур кольца***

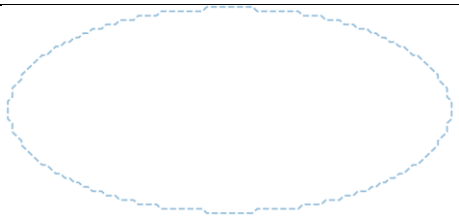
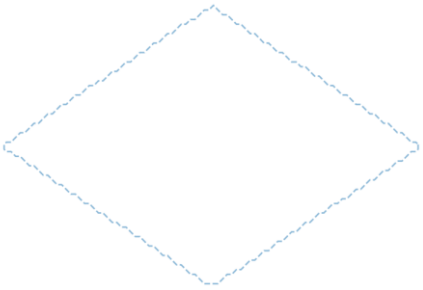

	

Рисунок 8.3 - Выделенный контур овала	***Рисунок 8.4 – Выделенный контур ромба***

Следующий шаг – упрощение контура алгоритмом *RDP*. Алгоритм с параметром $\varepsilon = 5$ оставляет минимальное количество точек (рисунки 9.1-9.4), необходимое для описания фигуры.

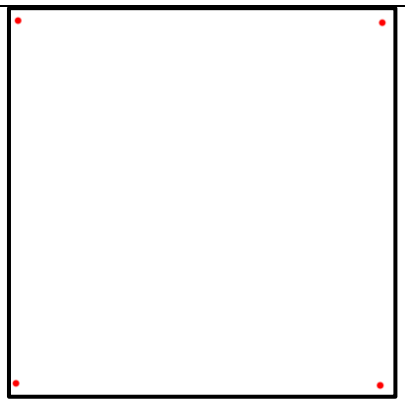


Рисунок 9.1 - Характерные точки квадрата

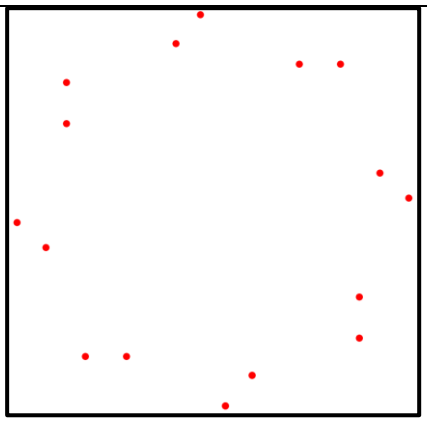


Рисунок 9.2 - Характерные точки кольца

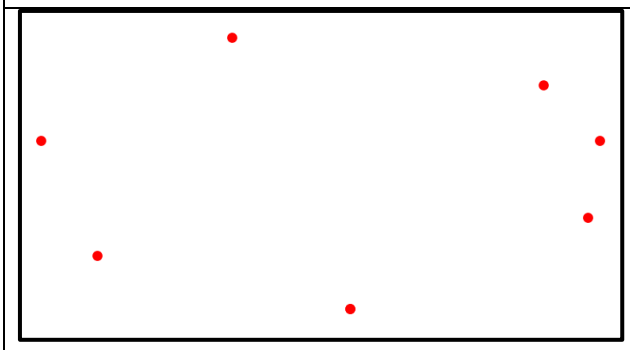


Рисунок 9.3 - Характерные точки овала

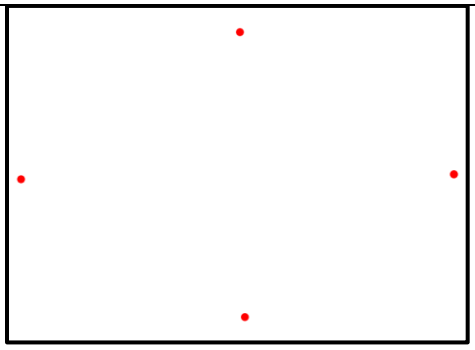


Рисунок 9.4 – Характерные точки ромба

19.3. Качество аппроксимации на различных контурах

Финальный этап – аппроксимация *PCHIP*-сплайнами. На основе параметризованных данных для координат X и Y независимо строятся монотонные кусочно-кубические сплайны (рисунки 10.1-10.4).

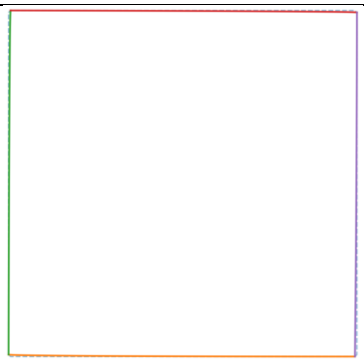


Рисунок 10.1 - Аппроксимированные PCHIP-сплайны квадрата

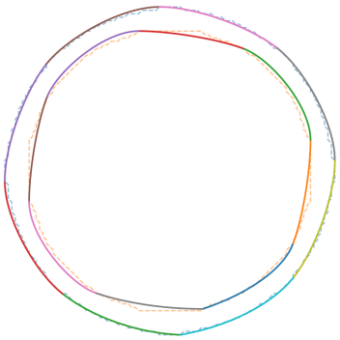


Рисунок 10.2 - Аппроксимированные PCHIP-сплайны кольца

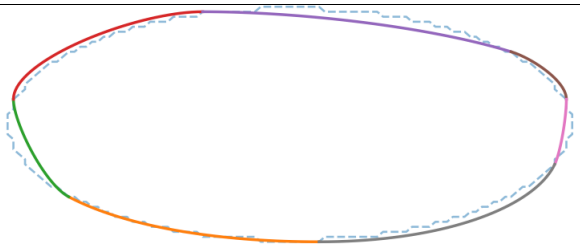


Рисунок 10.3 - Аппроксимированные PCHIP-сплайны овала

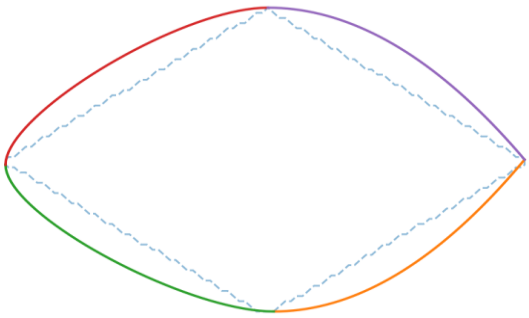


Рисунок 10.4 – Аппроксимированные PCHIP-сплайны ромба

На квадрате (рисунок 10.1) и кольце (рисунок 10.2) полученные *PCHIP*-кривые легли почти идеально на исходные контуры. Это объясняется двумя основными факторами. Во-первых, алгоритм *RDP* для этих фигур оставляет минимально необходимое количество точек: для квадрата – четыре вершины, для кольца – точки, равномерно распределенные по окружности. Во-вторых, *PCHIP*-интерполяция при прохождении через небольшое число корректно расположенных узловых точек успешно воспроизводит как прямолинейные сегменты (квадрат), так и участки постоянной кривизны (кольцо). Монотонность метода гарантирует отсутствие осцилляций и самопересечений, что и приводит к высокой точности аппроксимации.

На овале (рисунок 10.3) наблюдается небольшое, но заметное отклонение *PCHIP*-сплайна от исходного контура, особенно на участках с максимальной кривизной. Это связано с тем, что эллипс не является кривой,

точно описываемой кубическими полиномами. *PCHIP*-интерполяция, будучи методом, который проходит точно через заданные узловые точки, строит гладкую кривую между ними, но эта кривая является кубической, а не эллиптической. В результате на участках между точками, выбранными алгоритмом *RDP*, сплайн может незначительно «спрямлять» исходную дугу, что и приводит к отклонениям. Чем меньше точек остается после упрощения, тем более заметным может быть этот эффект.

Наиболее существенные отклонения наблюдаются на ромбе (рисунок 10.4). Вместо ожидаемых прямолинейных сторон и острых углов аппроксимированная кривая приобретает выпуклую, овально-образную форму. Прямые сегменты становятся дугами, выпуклыми относительно центра фигуры, а углы – сглаженными. Это является прямым следствием основного свойства *PCHIP*-интерполяции и кубических сплайнов в целом: они стремятся обеспечить гладкость (непрерывность первой, а в случае *PCHIP* – разрывной, но управляемой второй производной) в узловых точках. В вершинах ромба производная (направление кривой) резко меняется. Чтобы соединить такие точки гладкой кубической кривой, интерполятор вынужден создавать плавный переход, что неизбежно приводит к скруглению углов и искривлению прямых участков. Фактически, *PCHIP* «не знает», что исходный контур должен состоять из отрезков прямых. Он лишь строит самую гладкую в смысле отсутствия осцилляций кривую, проходящую через заданные вершины. Таким образом, для контуров с острыми углами и строгими прямолинейными участками *PCHIP*-аппроксимация, обеспечивая монотонность и гладкость, приносит в жертву геометрическую точность.

Таким образом, качество аппроксимации *PCHIP*-сплайнами напрямую зависит от соответствия исходной геометрии свойствам кубических полиномов. Метод демонстрирует отличные результаты на контурах, которые либо сами по себе хорошо приближаются кубическими кривыми, например, окружность при достаточном числе точек, либо состоят из небольшого числа

точек, соединение которых гладкой кривой не искажает восприятие формы – квадрат. Однако на фигурах, требующих точного воспроизведения прямых линий и острых углов, или на кривых, не являющихся кубическими, неизбежны отклонения. *PCHIP* жертвует геометрической точностью в углах ради гарантии монотонности и отсутствия осцилляций. Следовательно, выбор данного метода аппроксимации является оптимальным для задач, где приоритетом является получение гладкой, визуально приятной и предсказуемой кривой, но может быть неприменим в случаях, когда необходимо точное сохранение углов и прямолинейных сегментов, например, в техническом черчении или при векторизации архитектурных планов.

20 ДОСТИГНУТЫЕ РЕЗУЛЬТАТЫ

В результате выполнения проекта разработан программный инструмент, который полностью автоматизирует процесс преобразования растровых изображений в формате *PNG* в математически представленные параметрические кривые, готовые для использования в *LaTeX*-документации и системах компьютерного моделирования. Основным достижением является создание целостного конвейера обработки – от загрузки пиксельного изображения до генерации гладких, монотонных кусочно-кубических сплайнов с экспортом их уравнений.

Программа успешно реализует поставленные задачи, включая изучение структуры *PNG*-файлов, сегментацию белых областей с использованием пороговой обработки и построения бинарной маски, а также выделение контуров с помощью алгоритма *Marching Squares* через функцию *find_contours*. Была выполнена адаптация алгоритма Рамера-Дугласа-Пекера для упрощения контуров, что позволило значительно сократить количество точек при сохранении ключевых геометрических особенностей объекта – как показано на примере треугольника, где остаются только три вершины (рисунок 3.1). После упрощения контур параметризуется по нормированной

длине дуги, что аналогично движению автомобиля с одометром: параметр *t* плавно изменяется от 0 до 1, пропорционально пройденному расстоянию вдоль кривой.

Для аппроксимации был выбран метод *PCHIP (Piecewise Cubic Hermite Interpolating Polynomial)*, который, в отличие от классических кубических сплайнов, гарантирует сохранение монотонности исходных данных, исключая появление ложных экстремумов и осцилляций. Этот подход обеспечил построение гладких, лишенных самопересечений кривых, точно проходящих через характерные точки упрощенного контура. Однако, как показал анализ качества аппроксимации на различных геометриях, *PCHIP*-сплайны демонстрируют различную точность в зависимости от формы исходного объекта. На фигурах, хорошо приближаемых кубическими полиномами (например, окружность или квадрат при достаточном числе точек), кривые легли почти идеально (рисунки 10.1, 10.2). В то же время, на контурах с острыми углами и строгими прямолинейными участками, таких как ромб (рисунок 10.4), метод жертвует геометрической точностью ради гладкости, что проявляется в скруглении углов и искривлении сторон.

Архитектура кода построена по модульному принципу, что обеспечивает наглядность, а также возможность масштабирования. Каждый этап обработки помещен в отдельный класс или функцию: от загрузки изображений (*GetPng*) и построения маски (*ChoosePoints*) до обработки контуров, аппроксимации сплайнами и вывода результатов. Такое разделение позволяет легко модифицировать или заменять отдельные компоненты, например, изменить критерий сегментации или метод интерполяции.

Важным практическим результатом является создание модуля экспорта параметрических уравнений в формате *LaTeX*. Программа генерирует текстовый файл, содержащий для каждого сегмента кривой интервал изменения параметра *t* и соответствующие кубические полиномы для координат *x(t)* и *y(t)* с высокой точностью коэффициентов (рисунок 5.2).

Проведенный анализ вычислительной сложности подтвердил эффективность предложенного подхода. Хотя алгоритм *RDP* в худшем случае имеет квадратичную сложность $O(n^2)$, на практике, благодаря адаптивному упрощению и выбору параметра $\varepsilon = 5$, его производительность близка к $O(n \log n)$. Этап построения *PCHIP*-сплайнов обладает линейной сложностью $O(k)$, где k – количество точек после упрощения, что делает его быстрым даже для детализированных контуров. Суммарно это обеспечивает приемлемую производительность инструмента для обработки типичных графических объектов.

Таким образом, проект достиг основной цели – создания работоспособного инструмента для автоматической векторизации растровых изображений с получением математически строгого параметрического описания кривых. Программа прошла проверку на изображениях различной геометрической сложности (квадрат, кольцо, овал, ромб), продемонстрировав как свои сильные стороны – высокую точность на гладких контурах, гарантию монотонности, удобный экспорт, так и ограничения – потерю точности на угловатых фигурах. Результаты работы могут быть использованы в областях, требующих параметрического представления графических объектов: компьютерной графике, веб-дизайне, подготовке полиграфической продукции и создании анимаций, где аналитическое задание траекторий критически важно для плавности и масштабируемости.

21 ВОЗМОЖНЫЕ УЛУЧШЕНИЯ И РАСШИРЕНИЯ ФУНКЦИОНАЛА

Одно из основных направлений – совершенствование этапа аппроксимации. Как показал анализ (рисунки 10.3, 10.4), использование кубических *PCHIP*-сплайнов не всегда обеспечивает точное воспроизведение прямолинейных сегментов и острых углов. Для этого нужна функция,

автоматически определяющая характер участков контура (прямой или криволинейный) и выбирающего соответствующий метод аппроксимации. Альтернативой может служить использование сплайнов более высокой степени с адаптивным выбором узлов, что позволит лучше приближать сложные кривые, такие как эллипсы, ценой увеличения вычислительной сложности, времени выполнения программы и возможного появления осцилляций.

Для повышения удобства использования инструмента целесообразно разработать графический интерфейс пользователя. Он позволил бы интерактивно загружать *PNG*-файлы, визуализировать все этапы обработки (исходное изображение, бинарную маску, исходный и упрощенный контуры, аппроксимированную кривую), а также в реальном времени регулировать ключевые параметры, такие как *epsilon* для *RDP* или пороги сегментации, сразу наблюдая влияние на результат. Такой интерфейс стал бы мощным средством для настройки и отладки, особенно для пользователей, не знакомых с программированием.

Текущая реализация экспортирует параметрические уравнения только в формате *LaTeX*. Расширение списка выходных форматов увеличило бы применимость инструмента в различных областях. Например, экспорт в форматы *SVG (Scalable Vector Graphics)* позволил бы напрямую использовать векторизованные кривые в графических редакторах и веб-разработке. Генерация кода на языках, используемых в мехатронных и робототехнических комплексах с описанием траекторий в виде функций, могла бы напрямую интегрировать результаты работы в системы управления движением.

Наконец, для обработки сложных изображений, содержащих несколько разноцветных объектов или объектов, не являющихся чисто белыми, логичным развитием было бы расширение функционала сегментации с поддержкой выделения областей по произвольному цвету или диапазону цветов в *RGB* или *HSL*-пространстве (*Hue, Saturation, Lightness* – тон,

насыщенность, яркость). Это превратило бы инструмент из специализированного средства обработки белых логотипов в универсальный конвертер растровых изображений в параметрические кривые, пригодный для более широкого круга задач компьютерной графики и автоматизации проектирования.

СПИСОК ИСТОЧНИКОВ

1) PNG (Portable Network Graphics) Specification (Second Edition) [Электронный ресурс]. – Режим доступа: https://www.w3.org/TR/PNG/ (дата обращения: 08.12.2025).

2) SciPy Documentation. PchipInterpolator [Электронный ресурс]. – Режим доступа: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.PchipInterpolator.html (дата обращения: 09.12.2025).

3) scikit-image Documentation. find_contours [Электронный ресурс]. – Режим доступа: https://scikit-image.org/docs/stable/api/skimage.measure.html#skimage.measure.find_contours (дата обращения: 09.12.2025).

4) Matplotlib Documentation [Электронный ресурс]. – Режим доступа: https://matplotlib.org/stable/index.html (дата обращения: 10.12.2025).

5) NumPy Documentation [Электронный ресурс]. – Режим доступа: https://numpy.org/doc/ (дата обращения: 20.05.2025).

6) Pillow (Python Imaging Library) Documentation [Электронный ресурс]. – Режим доступа: https://pillow.readthedocs.io/en/stable/ (дата обращения: 20.05.2025).

7) Хилл К. Научное программирование на Python : пер. с англ. А. В. Снастина. – М. : ДМК Пресс, 2021. – 646 с. : ил.

8) Криволапов С. Я., Хрипунова М. Б. Математика на Python : учебник. – М. : КНОРУС, 2022. – 456 с. – (Бакалавриат и магистратура).