

# Оглавление

<b>Введение</b>	<b>5</b>
<b>1 Аналитический раздел</b>	<b>7</b>
1.1 Анализ предметной области	7
1.1.1 Холст	7
1.1.2 Гвоздики	8
1.1.3 Нить	9
1.1.4 Схема плетения	10
1.2 Обзор существующих методов	11
1.2.1 Петрос Вреллис	11
1.2.2 Ани и Андрей Абакумовы	12
1.2.3 Реализация Бартона Дринга	13
1.2.4 Проект Рафаэля Шаафа	14
1.2.5 Проект shlonkin	16
1.2.6 Реализация в браузере	17
1.2.7 String Art: Towards Computational Fabrication of String Images	20
1.2.8 Сравнительная таблица реализаций	26
1.3 Вывод	26
<b>2 Конструкторский раздел</b>	<b>28</b>
2.1 Пренебрегаемые и дополнительные параметры	28
2.1.1 Толщина гвоздя	28
2.1.2 Минимальная хорда	28
2.1.3 Максимальное количество соединений	28
2.2 Описание алгоритма	29
2.2.1 Инициализация, подготовка к генерации	30
2.2.2 Обработка входного изображения	32
2.2.3 Отрисовка отрезков и расчет ошибки	33
2.2.4 Цикл генерации схемы	41
2.3 Пользовательские возможности	42
<b>3 Технологический раздел</b>	<b>43</b>
3.1 Средства программной реализации	43
3.1.1 Язык программирования	43
3.1.2 Используемые библиотеки	44
3.2 Запуск приложения	46
3.3 Тестовые данные и данные для экспериментов	46
3.4 Проверка обработки изображения	48

3.5 Результат отрисовки соединений	48
3.6 Проверка определения оптимального количества соединений	49
3.7 Пример работы	49
<b>Заключение</b>	<b>51</b>
<b>Источники</b>	<b>52</b>

## **Введение**

Стринг-арт — это вид искусства, который заключается в рисовании нитями — натягивании их на гвоздики, вбитые в твердую поверхность. «String» с английского языка переводится как нить, струна; «art» - искусство. Этот необычный вид рукоделия называют также струнным искусством, нитяной графикой или «изонитью» (от «изображение» + «нить»).

Люди занимаются этим в качестве медитативного хобби и из-за педагогических эффектов изонити. Некоторые занимаются этим профессионально, продавая свои работы и проводя выставки [1]. Объектами стринг-арта украшают помещения, их можно приобрести на некоторых сайтах [2]. Распространенность этого бизнеса и цены говорят о наличии спроса на рынке. Продают также отдельно схемы и заготовки, предоставляя возможность пользователю самостоятельно создать картину [3].

Художники и математики, скульпторы и программисты находят все новые грани струнного искусства. Особенно интересно и ново в этой области - внедрение компьютерных технологий в создание работ. Люди пытаются автоматизировать процесс подготовки, разметки холста, натягивания нитей, стараются придумать более точные и быстрые алгоритмы расчета схем.

Один из видов изонити имеет простую стандартизированную заготовку, но далеко не простую схему натяжения. Здесь канвас имеет форму круга, а гвозди забиваются по окружности на равных расстояниях. Через гвозди нить натягивается прямыми линиями по пути последовательных хорд. Отсутствие нити дает полностью цветовой тон. Тон темнеет по мере увеличения плотности и пересечений нити. Таким образом, возможна полная палитра оттенков. Создание таких картин может занимать недели, расход нити достигает нескольких километров. Генерацию схемы плетения невозможно представить без предварительных программных расчетов — для поиска последовательности натяжений нити используют большое количество компьютерных вычислений, которое позволяет найти наилучшую аппроксимацию изображения.

Алгоритмы создания выдающихся работ данного вида засекречены коммерческой тайной. У большого количества реализаций с открытым кодом отсутствует документация и описание методов, некоторые проекты не дают качественного результата или их не удалось запустить вовсе. Помимо этого, проблема обладает большой вариативностью и множеством влияющих параметров, что является почвой для исследований.

Целью данной работы является создание алгоритма генерации схемы плетения картины описанного вида стринг-арта с понятной документацией и удобным доступным инструментом использования. Для достижения поставленной цели необходимо решить следующие задачи:

- Аналитические
  - Провести аналитическое исследование существующих методов для автоматического создания картин стринг-арта.
  - Обосновать актуальность модификации или создания алгоритма.
- Конструктурские
  - На основе выбранного метода разработать алгоритм автоматической генерации схемы для создания стринг-арта.
  - Графически отобразить декомпозицию задачи и последовательность выполняемых операций на диаграммах.
  - Разработать необходимые для реализации структуры данных.
  - Подготовить тестовые данные для отладки и экспериментов.
  - Разработать интерфейс для ввода необходимых данных и анализа результата работы алгоритма.
- Технологические
  - Реализовать разработанный алгоритм. Протестировать функциональность и отладить созданный программный продукт.
  - Проанализировать качество реализованного алгоритма, его временные и производительные характеристики.
  - На основе подготовленных данных для экспериментов оценить успешность создаваемых с помощью алгоритма картин.

## 1 Аналитический раздел

В данном разделе будут приведены результаты аналитического исследования существующих методов автоматического создания картин стринг-арта. В частности будет рассмотрен и формализован вид нитяной графики, которому свойственны круглое полотно, расположение гвоздей по окружности с одинаковым интервалом, натяжение непрерывной нити вдоль хорд. Будет обоснована актуальность модификации или создания алгоритма.

### 1.1 Анализ предметной области

Предметная область данного исследования — определенный вид струнного искусства, его составляющие, варьируемые параметры и реализации. Ниже на рисунке 1.1 приведены схематическое изображение составляющих данного типа стринг-арта.

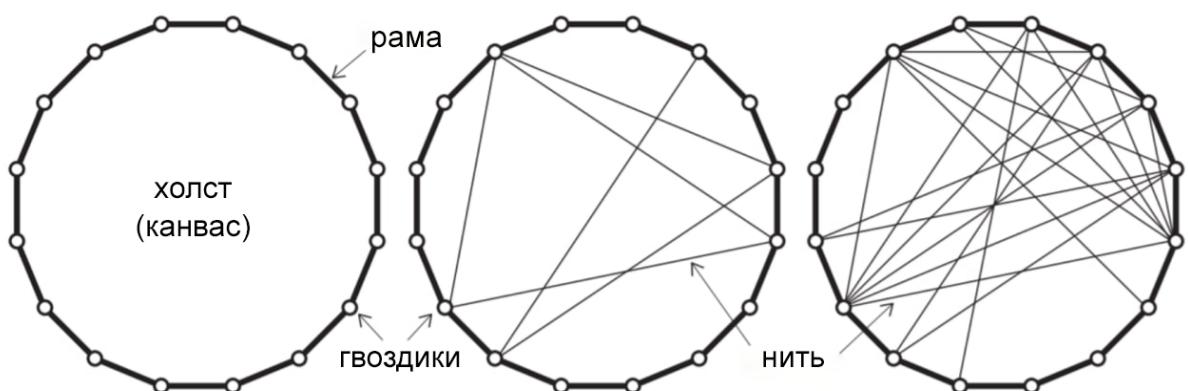


Рисунок 1.1 - Схематическое изображение составляющих стринг-арта.

#### 1.1.1 Холст

Холст имеет форму круга, основа может представлять собой как целиком круг, так и только обруч. Полученное изображение стремится от правильного многогранника к кругу при увеличении количества гвоздей, вбитых по окружности с равным угловым шагом. Однако, их количество ограничено физическими параметрами гвоздя и платформы, на которой они будут закреплены.

Пересечения нити на площади полотна создают пятна, оттенки. Результат плетения можно представить компьютерно как пиксельное изображение совокупности проведенных соединений методами отрисовки отрезков. Этот факт позволяет сравнивать исходное и полученное на выходе изображения, и в итоге принимать решение о проведении соединений.

Холст дискретизируется с помощью регулярной сетки, и предполагается, что физическая толщина нити  $t$  в сочетании с диаметром холста  $d$  определяет его разрешение  $Z$ . То есть, если соединение нарисовано горизонтально или вертикально и проходит через середину пикселя, оно полностью покрывает его.

$$Z = \frac{d}{t}$$

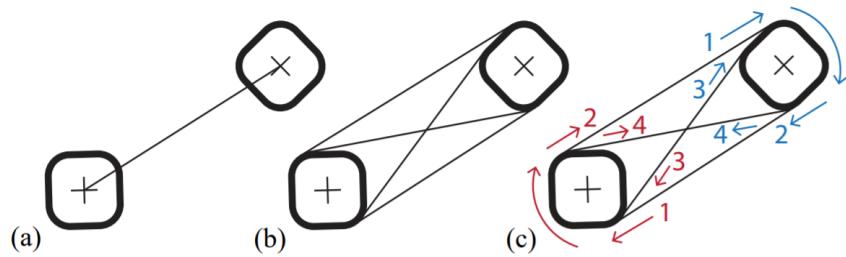
где  $Z$  - разрешение,  $d$  - диаметр холста,  $t$  - толщина нити

Подобрав толщину нити и размер холста, можно определить желаемое разрешение. В соответствии с формулой, чем тоньше нить и больше канвас, тем выше будет разрешение исходной картины, и соответственно детальнее результат.

### 1.1.2 Гвоздики

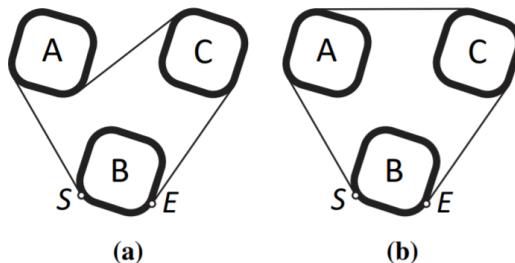
Так называемые гвоздики (pins) - любой вид штыря или иголки, желательно со шляпкой и максимально статический. Для надежной работы он должен быть крепко закреплен или даже являться монолитной частью обруча.

Соединения между двумя штырями на практике может быть выполнено четырьмя разными способами, как показано на рисунке рисунок 1.2 (б). Если размер гвоздика больше толщины нити, диаметр холста мал, то отклонение возможных хорд может привести к значительной ошибке.



*Рисунок 1.2 - Типы соединений между двумя штырями.*

Важно отметить, что приведенные на рисунке виды соединений делятся на группы, вход - выход. Группы следующие (рисунок 1.2 (с)): 1-3 вход и 2-4 выход. Нить должна приходить с одной стороны и покидать гвоздик с другой по или против часовой стрелке. Невозможно физически воспроизвести плетение, где два последовательных соединения принадлежат одной группе (как на рисунке 1.3(а) - вокруг гвоздика A).



*Рисунок 1.3 - Последовательные соединения между штырями.*

### 1.1.3 Нить

На полученное изображение могут влиять толщина, прозрачность, ворсистость и прочность нити. Как было отмечено выше толщина нити определяет разрешение полученного изображения. Прозрачность, цвет и пушистость определяют формирование оттенков картины, нужное количество одинаковых соединений. Так, если нить полностью непрозрачна, то рисование одной и той же хорды несколько раз приводит к одному и тому же результату. Следовательно, вопрос, соединены ли два конкретных штыря нитью, является

бинарным. На практике полиэфирная оверлочная нить очень хорошо соответствует этому утверждению.

Нить должна быть прочной, она не должна рваться. Также натяжение нити имеет значение: она не должна провисать, но и не должна случайно лопнуть. В некоторых случаях порванная нить означает конец картине.

Если нить принципиально непрерывна (свойственно для автоматического плетения) - необходимо генерировать эйлеров путь соединения вершин-гвоздей.

#### 1.1.4 Схема плетения

Помимо физической основы и нити, нужна схема плетения. Данному виду свойственна компьютерная генерация последовательности натяжения нитей из цифрового изображения, переданного на вход алгоритма. На выходе алгоритма получается либо машинные команды для автоматического создания картины, либо удобная для человека инструкция.

Естественно, входное изображение подвергается обработке. Картина в итоге получается круглой, поэтому исходная фотография обрезается по окружности. В случае одноцветного плетения, необходимо представить входное изображение в градациях серого. Серая шкала отражает интенсивность света в каждом пикселе видимой части электромагнитного спектра. На результат может влиять контрастность изображения, его разрешение, а также размеры и четкость деталей.

Схема обычно выглядит как последовательность гвоздей, которых нужно соединить нитью. Гвоздики нумеруются тривиально - натуральными числами по кругу.

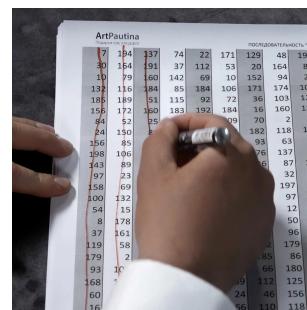


Рисунок 1.4 - Схема плетения (ArtPautina [3]).

## 1.2 Обзор существующих методов

Ниже будут описаны существующие методы решения задачи, а также приведена их сравнительная таблица.

### 1.2.1 Петрос Вреллис

Ярким представителем этого стиля стринга-арта является Петрос Вреллис [4]. В своей статье он рассказывает об изонити, однако, не раскрывает методы генерации схемы плетения и не приводит ссылок на исходный код. В статье говорится, что Петрос использует вычислительную технику для генерации паттернов. В его работе 2016 года шаблон создается на основе специально разработанного алгоритма, закодированного в openframeworks. Для создания каждого шаблона требуется более 2 миллиардов вычислений.

Алгоритм берет на вход цифровую фотографию и выводит схему плетения. Вязание выполняется вручную, по пошаговой инструкции, продиктованной компьютером. Художник использует круглоткацкий станок, который представляет собой алюминиевый обод диаметром 28 дюймов с 200 анкерными штифтами по окружности. Нить проходит от одного анкерного стержня к другому непрерывно 3000–4000 раз, достигая общей длины 1–2 км.

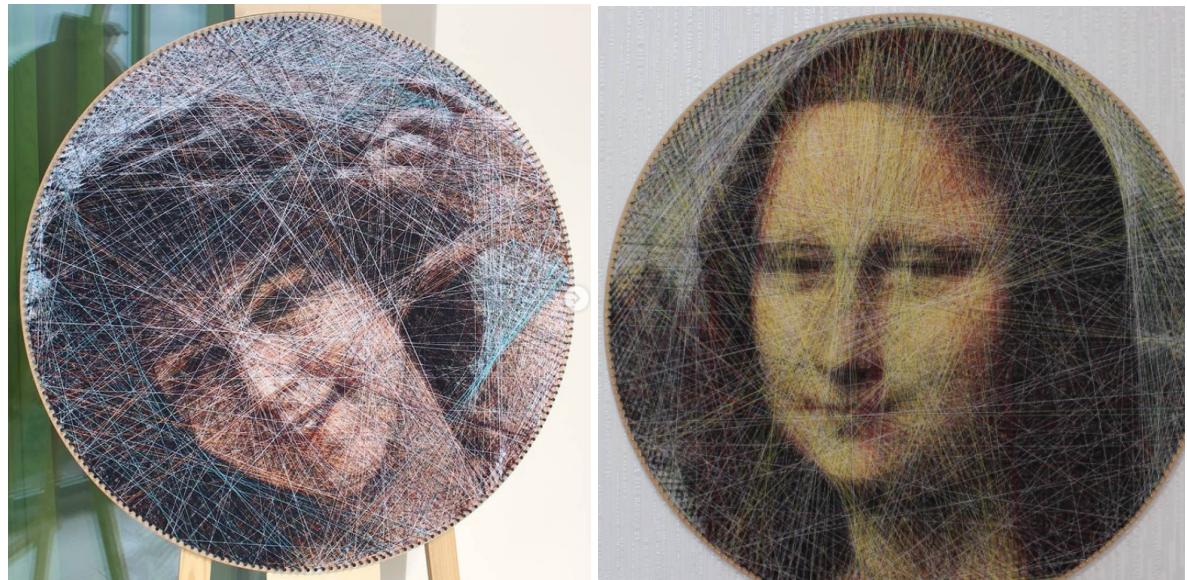


*Рисунок 1.5 - Примеры работ Петроса Вреллиса.*

### 1.2.2 Ани и Андрей Абакумовы

Также можно отметить Ани и Андрея Абакумовых как создателей уникальных работ стринг-арта. Совсем недавно - с 18 сентября по 22 ноября 2020 - в галерее современного искусства ARTSTORY в Москве прошла их выставка - "Ускользающая нить" [2]. Они добились исключительной передачи цвета и оттенков. Богатство тональных переходов зачастую создается ограниченным количеством цветов основы: благодаря наслажданию палитра художника значительно расширяется и усложняется.

Программист Андрей Абакумов разрабатывает математический алгоритм, генерирующий схему рисунка. Андрей также готовит "холст", набивая на него гвозди. И потом художница Ани Абакумова в определенной последовательности, слой за слоем, натягивает на гвозди нити, - так постепенно проступает изображение. На небольшие круглые картины уходит около четырех километров нитей и несколько дней, а на более масштабные произведения - до 20 километров и нескольких недель [5].



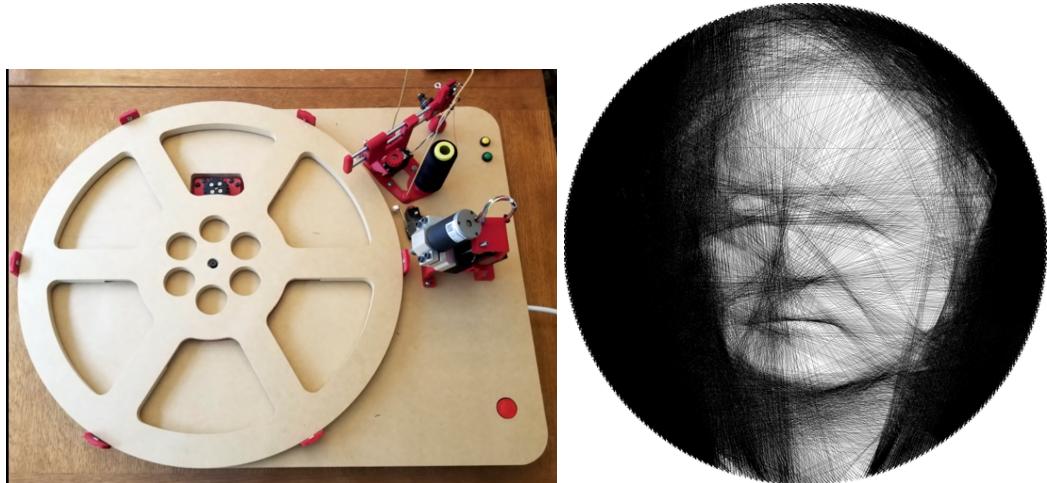
*Рисунок 1.6 - Примеры работ Абакумовых.*

### 1.2.3 Реализация Бартона Дринга

Немало впечатляющего материала можно найти в ютюбе. Например, видео [6] с канала Barton Dring является представителем автоматического плетения.

Для генерации Бартон использует ПО, размещенное на github (в описании к видео приведена ссылка на открытый код [7]). Он утверждает, что его разместили авторы одной из статей, которые его вдохновили.

Генерация достаточно плотного изображения может занимать несколько часов. Автор советует иметь минимум 32 Гигабайта оперативной памяти, иначе генерация займет в несколько раз больше времени. Алгоритм выдает текстовый файл содержащий путь соединения а такжерендированное изображение результата. Скрипт на языке программирования python переводит этот текстовый файл в g-code а также подсчитывает длину нити.



*Рисунок 1.7 - Созданная плетельная машина и результат.*

Но попытка запуска кода, на который ссылается Бартон, на собственном компьютере не увенчалась успехом - произошла ошибка Out of memory. Помимо того, код написан на matlab, следовательно необходимы права для его запуска. Также экспериментально выяснено, что программа требует установить дополнительные пакеты (Image Processing Toolbox).

#### 1.2.4 Проект Рафаэля Шаафа

Проект Рафаэля, статья о котором была найдена на hackaday [8], очевидно реализован с помощью GameMaker Studio. В статье есть ссылка на гугл-диск [9], в котором хранятся исполняемые файлы и множество файлов для интерфейса. Knit\_Generator.exe открывает приложение для ввода исходного изображения, которое выполняет обработку входных данных и непосредственно саму генерацию. По окончании генерации сохраняет инструкцию плетения в виде файла формата ini в папку.



*Рисунок 1.8 - Интерфейс Knit\_Generator.exe и результат работы.*

Knit\_Instructor.exe открывает интерфейс для отображения плетения по сохраненной инструкции. Knit Instructor предоставляет следующий интерфейс для следования схеме плетения.

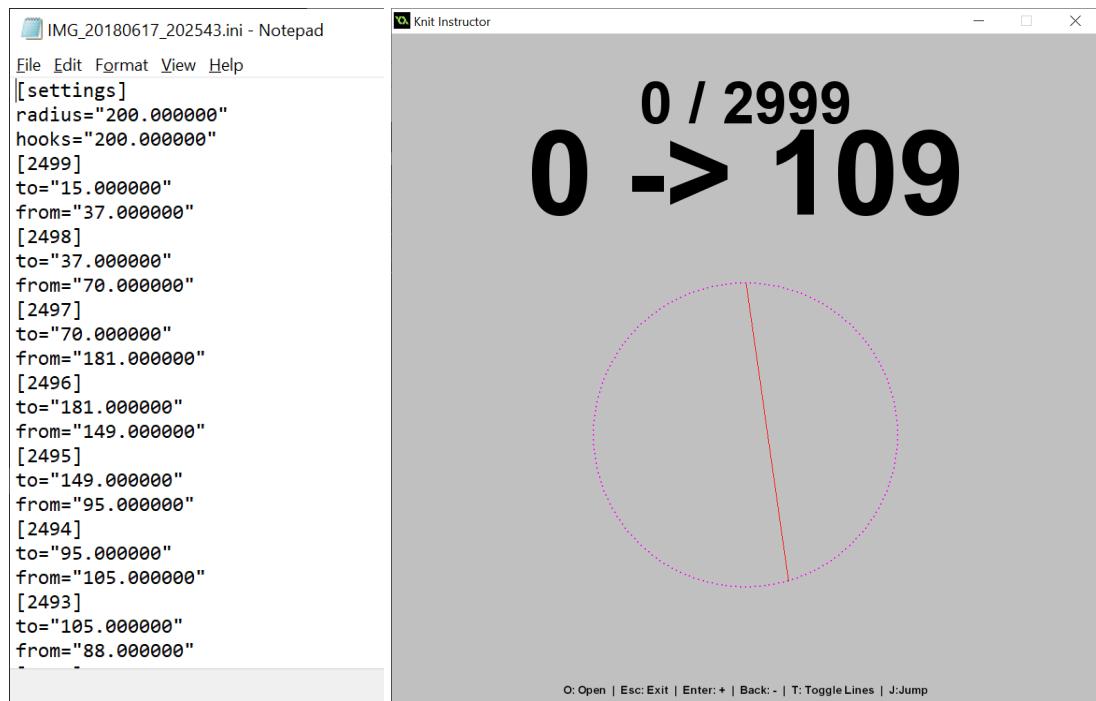


Рисунок 1.9 - Схема плетения и интерфейс Knit\_Instructor.exe.

Радиус круга в пикселях не имеет никакого отношения к физическому размеру портрета, он определяет скорее разрешение картины.

Можно задать количество линий, которые будут проведены между гвоздями. Видимо, достижение этого числа соединений останавливает цикл генерации. Если процесс генерации подходит к концу, и новые линии помещаются вокруг границы кольца, то значение слишком велико. Если они все еще проходят через кольцо, соединений может оказаться недостаточно.

Самый сложный для понимания параметр - контраст. В основном это значение, которое вычитается из каждого пикселя, когда через него проходит линия. Более высокое значение следует использовать для более толстой нити или для портретов меньшего размера. Экспериментально получено, что 20-50 является диапазоном хороших значений.

Приложение открывается на весь экран и не позволяет использовать другие приложения в процессе генерации.

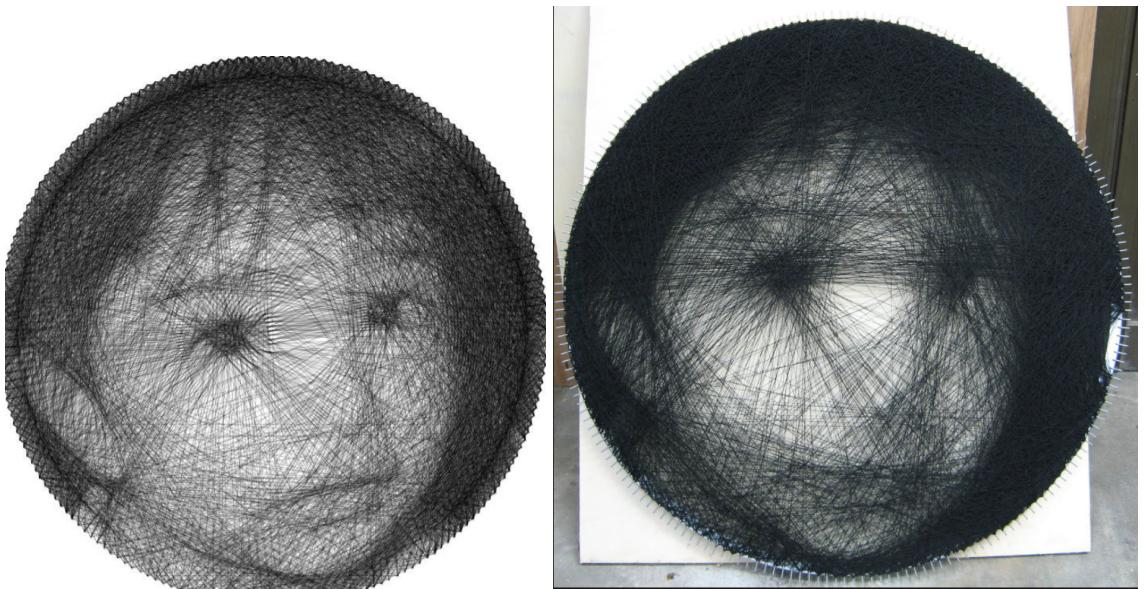
### 1.2.5 Проект shlonkin

Рассмотрим, еще один проект [10], обнаруженный на портале Hackday под авторством shlonkin. Автор предоставляет информацию об автоматическом плетении. К статье он прилагает открытый код [11].

За исключением файлов git, репозиторий содержит два файла: один формата ino, другой pde. Файл ino - это файл исходного кода программы на языке Arduino, который предназначен для программирования плат с процессором и микроконтроллером, используемых для сборки различных устройств. pde - файл исходного кода, написанный на языке программирования Processing.

Сначала изображение преобразуется в оттенки серого, и пользователь выбирает круговую область, которую он хочет использовать. Некоторое количество гвоздей (200 работает хорошо) равномерно распределены по кругу, а соединения между каждой парой отображаются как закрашенные пиксели. Начиная с одного гвоздя, алгоритм усредняет затемнение по каждому пути и выбирает тот, который является самым темным, за исключением некоторого количества гвоздей (20 или около того), окружающих начальный. Затем этот набор пикселей немного освещается. Это повторяется несколько тысяч раз. Все эти параметры, как утверждает автор, настраиваются пользователем для получения наилучшего изображения.

Для автоматического плетения используется велосипедный обод с сотнями проволочных штифтов по краю. Он вращается, и дозатор нити, перемещаемый соленоидом, оборачивает нить вокруг штифтов.



*Рисунок 1.10 - Ожидаемый результат и результат плетения.*

Сам автор отмечает, что результат выглядит не так хорошо, как должен. Параметры для расчета линии нужно немного изменить. Разработчик случайно выполнил расчет для 200 радиальных штифтов вместо 207 на реальном ткацком станке. Поэтому с правой стороны семь пустых гвоздей, а грань заметно вытянута. Вот некоторые вещи, которыми автор захотел поделиться в комментариях к своей работе. Нить оказалась намного толще, чем он предполагал. Картина выглядит темнее, а черные части изображения на самом деле получаются непрозрачными и толстыми. Он говорит, что должен был скорректировать симуляцию, чтобы показать более темные линии и использовать более тонкую нить. А также он мог бы увеличить минимальную длину строки - 25-30 контактов, а не 20 подойдут.

#### 1.2.6 Реализация в браузере

Поиск существующих решений также проводился и просто среди репозиториев на github. Вот хороший результат исследования [12]. Код написан на js, html, css. Интерфейс выглядит следующим образом (рисунок 1.11), что естественно является плюсом данной реализации.

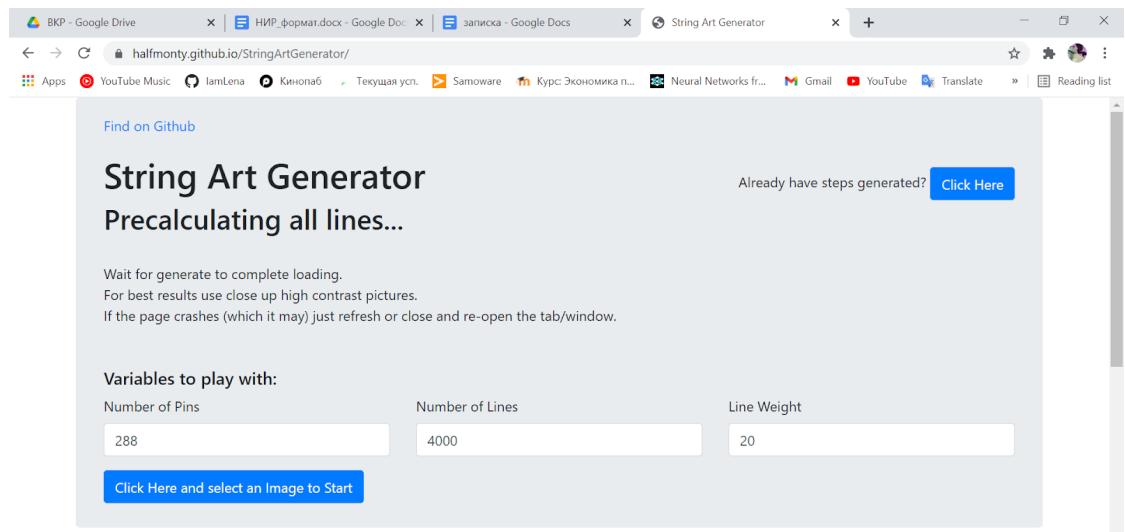


Рисунок 1.11 - Интерфейс.

Документации и описания алгоритма найдено не было, поэтому пришлось восстанавливать метод решения задачи по коду, ниже приведено краткое описание его этап работы.

- Обработка входных данных

Когда изображение загружено, оно обрезается по форме квадрата, центр совпадает с исходным, сторона выбрана как минимальная из ширины и длины. Далее изображение переводится в градации серого методом среднего. Потом оно обрезается по окружности, где центр совпадает с центром изображения, а радиус равен половине стороны.

- Подготовка к генерации схемы

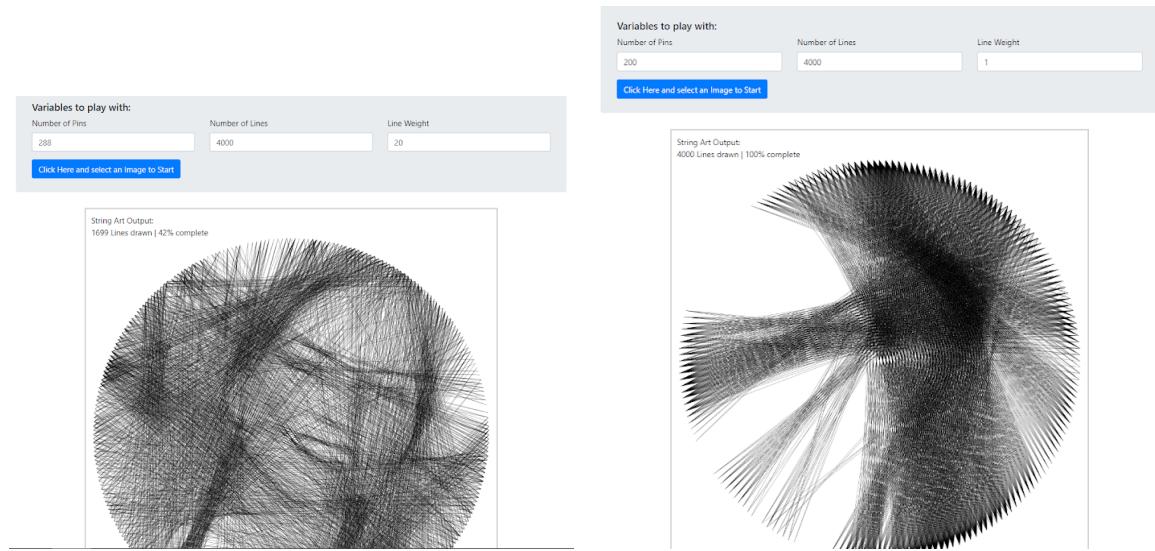
Начинается процесс генерации схемы. Сначала заполняются координаты расположения гвоздей в виде массива массивов [x, y]. Когда все координаты заполнены, начинается предварительный расчет соединений. Рассчитываются длины всех возможных соединений, вес нити инициализируется единицей. Заполняется структура всех возможных соединений (не рассматривает соединения гвоздей которые ближе чем на 20 гвоздей). Длина массива равна квадрату количества гвоздей (все гвоздики соединены со всеми).

- Генерация последовательности

Далее работу выполняет так называемый калькулятор линий. Инициализируется ошибка - отрицательными значениями пикселей исходной картины. Получается, что пока у нас канвас пустой, он не соответствует исходной картине ровно на количество цвета в каждом пикселе. Результат инициализирует тут же как матрица нулей.

Плетение начинается с нулевого гвоздя, он заносится в массив последовательности плетения. Далее в цикле, пока не достигнуто максимальное количество соединений, заданное пользователем, перебираются возможные соединения этого гвоздя с другими. Рассчитывается ошибка и если она больше (ошибка представляет собой отрицательное число при инициализации, то чем ее значение фактически математически больше), чем максимальная, то ее значение и значение лучшего пина обновляются. Лучший гвоздь добавляется в массив последовательной схемы, к массиву ошибок добавляется это соединение. Текущий гвоздь равен концу рассчитанного лучшего соединения, алгоритм переходит на поиск следующего соединения.

Структуры весьма непонятные. Ошибка определяется как абсолютная разница между исходными и сгенерированными пикселями, причем имеет отрицательное значение. Инструмент обрабатывает входное изображение любого цвета и формы. Входные параметры не позволяют определить физические параметры холста и нити. Параметр максимального количества соединений и веса нити не понятны пользователю. Их значение подбирается экспериментально и не гарантирует лучшего результата.



*Рисунок 1.12 - Примеры работы на разных параметрах.*

### 1.2.7 String Art: Towards Computational Fabrication of String Images

Данная статья [13] довольно популярна. В ней формализованным математическим языком повествуется о некоторых реализациях, приведен их сравнительный анализ, описывается метод решения и автоматическая его реализация.

Входные данные формализуются как вектор столбец, представляющий собой последовательность значений пикселей в градации серого. А выходные как бинарный вектор-столбец, размер которого равен количеству всех возможных соединений. Единица обозначает наличие этого соединения в результирующей картине, ноль - отсутствие. Задача ставится как минимизация ошибки между исходным и сгенерированным методом Брезенхема отрисовки отрезков изображениями.

- Определение ошибки методом линейных наименьших квадратов

Метод наименьших квадратов - самый простой способ определения ошибки между сгенерированным и исходным изображением. Задача заключается в нахождении такого  $x$ , чтобы

$$\min \|Ax - y\|_2^2, \text{ где } A : R^n \rightarrow R^m, \text{ где}$$

$n$  - количество всех возможных соединений,

$m$  - количество пикселей,

$x$  - бинарный вектор-столбец выбранных соединений,

$y$  - входное изображение - значения пикселей построчно друг за другом,

$A$  - матрица перевода пространства соединений в пространство пикселей.

Строки соответствуют пикселям, колонны - соединениям.  $A_{ij} = 1$  если пикセル  $i$  покрывается (даже частично) соединением  $j$ , иначе ноль. Матрица заполняется на основе алгоритма Брезенхема отрисовки отрезков. Результирующее изображение получается как произведение  $z = Ax$ , где  $x$  - бинарный вектор-столбец выбранных соединений. На практике используется сглаживающий алгоритм отрисовки и  $A_{ij}$  принимает значения  $[0, 1]$ . Однако  $x$  получается содержит не бинарные значения, а также может быть отрицательным или больше единицы. Это не соответствует физическому смыслу. Простое решение - округлить до 0 или 1. Это округление хоть и решает проблему со значениями, но дает неадекватный результат (рисунок 1.13(с)).

- Учет влияния толщины нити на разрешение

Решение выше не принимает во внимание влияние физической толщины нити на получаемое разрешение картины. Поэтому предлагается использовать оператор масштабирования к входному изображению.  $U : R^m \rightarrow R^{sm}$ . который переводит входное изображение в разрешения картины стрингарта.  $S$  - фактор масштаба равен количеству суперпикселей который приходится на входной пиксель. Он выбирается таким образом, чтобы суперпиксель соответствовал толщине нити.

$$\min \|\tilde{A}x - Uy\|_2^2, \text{ где } \tilde{A} : R^n \rightarrow R^{sm}, \text{ где}$$

$n$  - количество всех возможных соединений,

$m$  - количество пикселей,

$x$  - бинарный вектор-столбец выбранных соединений,

$y$  - значения пикселей входного изображения построчно друг за другом,

$A$  - матрица перевода пространства соединений в пространство пикселей.

$U$  - оператор масштабирования,

$s$  - фактор масштаба

Однако  $x$  все еще остается не бинарным - что не соответствует понятию произведено это соединение или нет. И округление также приводит к некорректному результату (рисунок 1.13 (e)).

В общем, критический минус использования метода наименьших квадратов, это его не бинарный выход.

- Итеративный “жадный” алгоритм

Процесс начинается с пустого холста и поэтому инициализируем выходной вектор  $x = 0$ . Затем мы последовательно добавляем соединений, по одному, и обновляем вектор  $x$  соответственно. Чтобы отслеживать прогресс, после каждого обновления вектора  $x$  мы вычисляем текущий результат реконструкции и соответствующую норму. На каждой итерации мы выбираем соединение, которое позволяет максимально уменьшить норму, и мы останавливаемся, когда дальнейшее добавление вызовет увеличение ошибки. Это соединение отмечается в массиве единицей.

Рисунок 1.13 (f) и (g) соответствуют бинарному массиву  $x$ . Второй результат с учетом разрешения и толщины. Авторы также отмечают, что генерация второго изображения заняла 14.7 часов.

- Выразительный диапазон на пиксель

Было замечено, что изображения получаются резко контрастными, содержащими либо сильно темные, либо сильно светлые пятна. Авторы утверждают, что причина в том, что выходному изображению не хватает выразительного диапазона на пиксель (expressive range per pixel). Другими словами, каждый пиксель это линейная комбинация вклада каждого соединения, что в результате почти всегда дает значение больше единицы.

Суть их решения состоит в том, чтобы рисовать соединения в высоком разрешении (и, следовательно, также для выполнения требования о разрешении канваса определяемом физическими параметрами - диаметром и толщиной нити), но для вычисления нормы использовать меньшее разрешение и в то же время увеличить количество пикселей на суперпиксель (в описанном примере  $s = 8 \times 8$ ). Используя этот метод, удалось расширить выразительность выходного изображения во время оптимизации и улучшить визуальные результаты значительно, как показано на рисунке. Кроме того, поскольку оценка выполняется в пространстве с низким разрешением, значительно уменьшается вычислительная нагрузка, результат на рисунке 1.13 (j) занял 2,3 часа.

- Абсолютная ошибка и иерархический подход к разрешению

Авторы в начале своей работы также пробовали расчитать ошибку как абсолютную разницу между значениями пикселей ( $L^1$ -норма). Для решения использовалась библиотека GUROBI. Но оказалось, что задача слишком большая (трудная с точки зрения количества вычислений) для обычного компьютера.

Но, так как они нашли способ упростить вычисления (описан ранее), они вернулись к методу абсолютного значения ошибки и GUROBI. Чтобы решить эту проблему с помощью GUROBI, они разработали иерархический подход с несколькими разрешениями, который работает только с подмножеством возможных краев при каждом разрешении. Точнее, вычисления происходят с изображения  $512 \times 512$  до разрешения  $32 \times 32$  пикселей. Затем повторно решается уравнение, начиная с самого низкого разрешение ( $32 \times 32$ ) со всеми 130 560 краями в матрице  $\tilde{A}$ . На каждой итерации увеличивается разрешение

изображения на основе изображения на следующем верхнем уровне расчетов, в то же время уменьшается количество доступных ребер (равное количеству столбцов  $\tilde{A}$ ). Чтобы добиться согласованности на всех уровнях пирамиды, используется результат каждой итерации для определения столбцов  $\tilde{A}$  для следующей итерации.

Авторы утверждают, что ненулевые элементы  $x$  каждой итерации обеспечивают хорошее предположение об оптимуме, поэтому они всегда используют соответствующие столбцы в  $\tilde{A}$  для следующей итерации. Поскольку ограничение только на ребра, соответствующие  $x$ , сильно ограничили бы разнообразие возможных результатов, они дополнительно добавляют ребра к  $\tilde{A}$  до максимальной квоты. Эмпирически определили уменьшение этой квоты как масштабированную и сдвинутую  $e^{-x}$  функцию, обеспечивающую приемлемое время выполнения с заметным повышением качества результата на каждой итерации. В результат изображен на рисунке 1.13 (h).

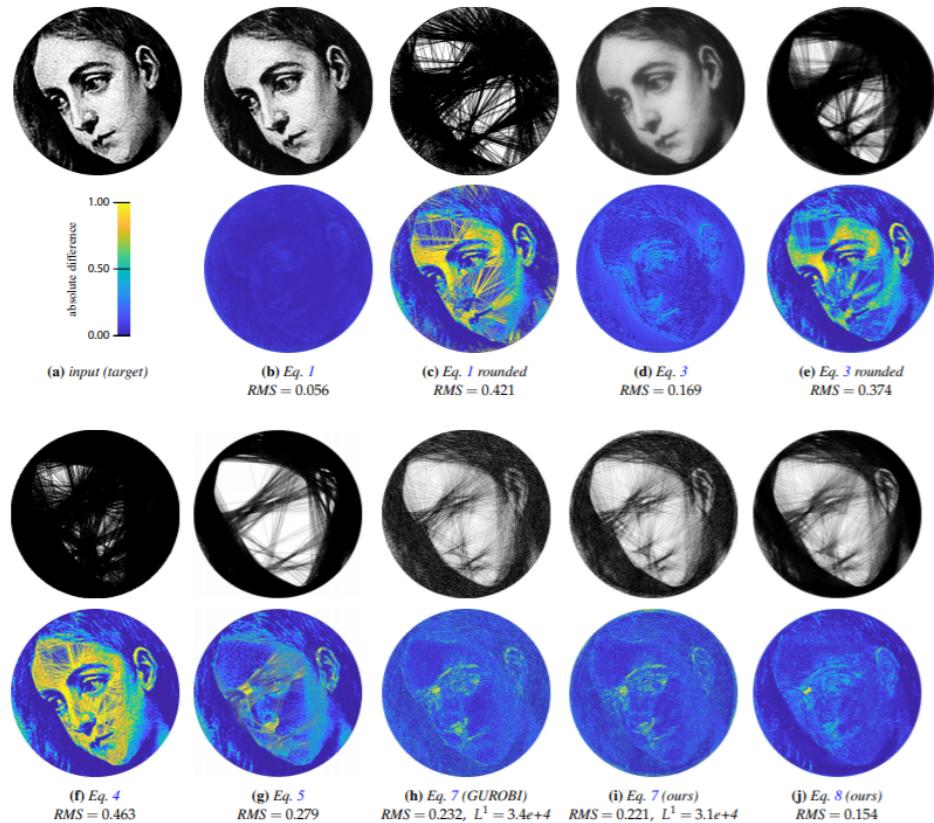


Рисунок 1.13 - Примеры работ, описанные в статье.

- Карта значимости

Также в этой статье они предлагают наложение карты значимости на изображение (importance mapping). Чтобы ввести дополнительный контроль над результатами, они используем карту значимости, которая позволяет повысить важность отдельных регионов исходного изображения вручную. Это позволяет снизить важность определенных пикселей, например фонового шума или другие менее выразительные черты лица.

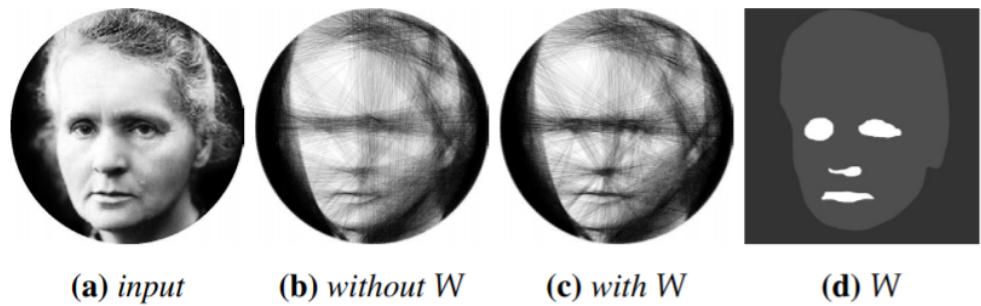


Рисунок 1.14 - карта значимости.

Таким образом задача выглядят так:

$$\min \| WDC(\tilde{A}x) - Wy \|_2^2, \quad x \in B^n, \text{ где}$$

$W$  - карта значимости

$D$  - оператор понижающий дискретизации, который фильтрует изображение обратно до желаемого разрешения

$\tilde{A}$  - матрица перевода пространство соединений в пространство пикселей с учетом разрешения

$x$  - бинарный вектор-столбец размером равным количеству возможных соединений

$y$  - исходное изображение

$n$  - количество всех возможных соединений

$\| . \|_2^2$  -  $L^2$  - норма.

## 1.2.8 Сравнительная таблица реализаций

Таблица 1 - Сравнение описанных реализаций и методов решения.

	Доступность кода	Удалось запустить	Документация	Качество изображения
Петрос Врелис	-	-	Личная веб-страница и интервью.	хорошее
Абакумовы	-	-	Инстаграм аккаунт с результатами и процессом, интервью.	хорошее
Бартон с youtube	+(matlab + python)	-	Видео об автоматическом плетении. Документации и пояснений к коду нет.	хорошее
Рафаэль с hackaday	+(исполняемые exe файлы)	+	Комментарии к статье.	Хорошее при подобранных параметрах
Шлонкин с hackaday	+(processing + arduino)	-	Описывающая скорее интерфейс и результат статья. Документации нет.	Сгенерированное изображение хорошее, но реальная картина плохого качества
Веб с github	+(html, css, js)	+	Описание метода и алгоритма нет.	Хорошее при подобранных параметрах
Статья “Искусство струн”	-	-	Описание метода.	хорошее

## 1.3 Вывод

Таким образом, для создания данного вида стринг-арта, необходимо определить упомянутые выше параметры или же признать их незначимыми для конкретного случая. Сформировать некий интерфейс для ввода пользователем физически осмысленных реальных параметров своей картины. Стоит отметить, что эти факторы влияют не только на вид изображения, но и на количество вычислений при генерации схемы.

Задача, решаемая алгоритмом, - нахождение наилучшего способа соединения гвоздей, который бы минимизировал отличие (так называемую

ошибку) полученной картины от исходного изображения. Следует определить метод отрисовки отрезка и определения значения ошибки, описать процесс поиска лучшего соединения.

Помимо физических параметров будущей картины стринг-арта, входными данными является цифровое изображение (в лучшем случае любого разрешения, размера и цвета). Реализуемое ПО должно выполнять такую обработку исходного изображения, как обрезание по форме круга, перевод в градации серого, возможную коррекцию интенсивности.

Выходные данные - схема одноцветного плетения непрерывной нитью. Также для анализа алгоритм должен сохранять обработанное исходное изображение, превью картины (изображение, обрисованное методами компьютерной графики по сгенерированном схеме), конфигурационную информацию и посчитанные данные о времени генерации, количестве соединений, длине нити и ошибке полученного рисунка.

Среди ограничений можно отметить следующее:

- Картина выполняется черной полностью непрозрачной непрерывной нитью.
- Пользователю требуется ввести физически осмыслиенные параметры - радиус холста в сантиметрах, количество гвоздей и толщину нити в сантиметрах.
- Пользователь несет ответственность за введенные параметры, качественное изображение не гарантируется в случаях малого разрешения или недостаточного количества гвоздей.
- Результаты сохраняются в виде файлов в папку.

## **2 Конструкторский раздел**

### **2.1 Пренебрегаемые и дополнительные параметры**

#### **2.1.1 Толщина гвоздя**

Толщиной гвоздя в данном алгоритме пренебрегается. Для лучшего изображения физические параметры таковы, что за счет дискретности, большого разрешения канваса и малого размера самого гвоздика, он будет вырождаться в точку. В итоге, рассматривать гвоздь как окружность и находить внешние и внутренние касательные теряет смысл. Этот трудоемкий процесс не стоит затраченных вычислительных мощностей и памяти. Данный параметр придется признать неважным. Все соединения описанные в алгоритме будут однотипные (рисунок 1.2(а)).

#### **2.1.2 Минимальная хорда**

Параметр минимальной хорды, присутствующий в некоторых реализациях, является неочевидным, но полезным. Его значение обозначает минимальное количество гвоздей между соединяемыми гвоздями. Например значение, равное 4, означает, что гвоздь с индексом 0 может быть соединен с 4, но не с 1, 2, 3. А также 0 может быть соединен с  $m-4$ , но не  $m-1$  или  $m-2$  или  $m-3$  (где  $m$  - общее количество гвоздей).

Данный параметр не определяется понятными физическими параметрами результативной картины. Но он ускоряет работу всего алгоритма, так как рассматривается меньшее количество соединений, и незначительно влияет на результат, так как обычно это является областью фона / шума.

По умолчанию значение устанавливается в единицу, таким образом рассмотрены будут всевозможные соединения. Однако занесем в алгоритм переменную минимальную хорду для исследований.

#### **2.1.3 Максимальное количество соединений**

Многие существующие решения задачи требуют от пользователя максимальное количество соединений. Реализуемый цикл поиска лучшей хорды

заканчивается, когда достигается максимальное число соединений, заданное пользователем. Данный параметр трудно выставить правильно даже для опытного художника, не говоря уже о новичках в струнном искусстве.

Оптимальное количество соединений зависит от картины. Например, если на входе белый квадрат, ни единого соединения не должно быть проведено, а если черный - то могут быть проведены все соединения (или их большая значительная доля - половина дает уже достаточную плотность изображения). Это значение придется подбирать экспериментально отдельно для каждого изображения, что неудовлетворительно.

В данном алгоритме будут рассматривать иные средства остановки цикла, такие как поиск оптимального количества соединений с помощью вычисления общей ошибки получающейся картины.

## 2.2 Описание алгоритма

Ниже приведено описание этапов разрабатываемого алгоритма, а также введены используемые далее обозначения.

1. Инициализация пользователем определяющих параметров
  - 1.1. Радиус канваса ( $R_c$ )
  - 1.2. Толщина нити ( $t$ )
  - 1.3. Количество гвоздей ( $m$ )
  - 1.4. Минимальная хорда ( $l_{min}$ )
2. Определение следующих констант
  - 2.1. Разрешение - количество пикселей в диаметре канваса ( $Z$ )
  - 2.2. Количество всех возможных соединений ( $N$ )
3. Инициализация структур данных (подготовка к генерации)
  - 3.1. Результат отрисовки ( $res[Z][Z]$ )
  - 3.2. Координаты гвоздей ( $pins[m][2]$ )
  - 3.3. Общая ошибка ( $whole\_error$ )
  - 3.4. Схема ( $schema$  “ ”)
  - 3.5. Длина нити ( $length$ )
4. Обработка входных данных

- 4.1. Загрузка фотографии
- 4.2. Обрезание по форме квадрата
- 4.3. Изменение разрешения
- 4.4. Обрезание по форме круга
- 4.5. Перевод в градации серого
- 4.6. Нормализация, изменение контрастности и яркости
- 4.7. Представление в виде двумерного массива значений интенсивности каждого пикселя ( $img[Z][Z]$ )
- 5. Генерация изображения
  - 5.1. Блок отрисовки соединений
  - 5.2. Блок расчета ошибки
  - 5.3. Цикл генерации схемы
- 6. Завершающий этап
  - 6.1. Сохранение изображения
  - 6.2. Сохранение схемы
  - 6.3. Оценка длины нити
  - 6.4. Оценка количества необходимых соединений
  - 6.5. Оценка времени генерации схемы
  - 6.6. Оценка качества полученного изображения

### 2.2.1 Инициализация, подготовка к генерации

Описание параметров и их влияние на изображение стринг-арта описано в аналитической части. Здесь будут отмечены ограничения, накладываемые на эти параметры.

$$R_c > 0 ; \quad t > 0 ; \quad R \gg t \quad m > 0 ; \quad 0 < l_{min} \leq m/2 ;$$

Все значения положительные. Верхняя граница радиуса основы, толщины нити и количества гвоздей определяется физическими возможностями. Радиус

холста и толщина нити - вещественные переменные, измеряемые в сантиметрах. Количество гвоздей и значение минимальной хорды - натуральные.

Формулы для определения констант из пункта 2 следующие:

$$Z = 2 \times R_c / t$$

$$N = m \times (m - 1 - 2 \times (l_{min} - 1)) / 2$$

Результат отрисовки (*res[Z][Z]*) - двумерный массив, размер которого соответствует матрице обработанного входного изображения. Значения инициализируются 255, что обозначает белый (пустой) холст. Данная структура обусловлена удобством отрисовки соединений, высвечиваемый пиксель индексируется элементарно как *res[y][x]*.

Координаты гвоздей (*pins[m][2]*) - двумерный массив размерности равной количеству гвоздей на два. Координаты - значения i, j на матрице результата, где располагается гвоздик. Соответственно индексы в этом массиве представляют собой нумерацию гвоздей по кругу, четко идентифицируя каждый (с 0 до m-1). Так как они представляют собой индексы и соответствуют пикселям на дискретном полотне - значения координат целочисленные.

Листинг 1 - Инициализация координат гвоздей.

```
center = Z / 2
radius = Z / 2
pin_coords = []
a_step = 360 / m
angle = 0
for i in range(0, m, 1):
    pin_coords.add([floor(center + radius * cos(angle)),
                    floor(center + radius * sin(angle))])
    angle += a_step
```

Ошибка (*whole\_error*) - вещественное целое число, принимающее значение от 0 до 255. Несет в себе среднее значение разности интенсивности в результате отрисовки соединения и желаемой интенсивности. Соединение дающее минимальное значение выбирается как самое лучшее соединение, так как дает лучшую аппроксимацию. Общая ошибка всей картины находится по формуле.

$$whole\_error = \sum_i^Z \sum_j^Z img[i][j] - res[i][j] / Z^2$$

Схема (*schema*) инициализируется пустой строкой. Индекс очередного гвоздя, который нужно соединить с предыдущим, заносится в строку. Периодически содержимое этой строки будет сохраняться в файл, а сама строка обнуляется.

Длина нити (*length*) представляет собой общую длину нити, которая потребуется для создания генерируемой картины. Она инициализируется нулем и на каждой итерации к ней добавляется масштабированная длина найденного соединения.

$$L = \sum_{i=0}^n \sqrt{(x_{pin1i} - x_{pin2i})^2 + (y_{pin1i} - y_{pin2i})^2} \times t$$

### 2.2.2 Обработка входного изображения

Исходное изображение будет загружено по некоторому пути до файла. Происходит обрезание изображение по форме квадрата, а далее по форме круга. Центром выбирается центр заданного изображения, а стороной или диаметром - меньшая из сторон входного изображения. После чего изменяется разрешение фотографии до разрешения картины стринг-арта  $Z$ , которое обусловлено физическими параметрами. Изображение переводиться в градации серого.

Далее изображение представляется в виде двумерного массива  $img[Z][Z]$ , где элемент обозначает интенсивность высвечивания пикселя. Получаем 256 уровней интенсивности света.

### 2.2.3 Отрисовка отрезков и расчет ошибки

В данной реализации будет использовано два метода отрисовки отрезков.

Первый - не сглаживающий целочисленный алгоритм Брезенхема разложения отрезка в растр. Его преимущество в скорости (рисунок 2.1). Два уровня интенсивности достаточно, чтобы определить, какое из соединений лучше. Более того, по факту требуемая бинарность соединений не обеспечивается сглаживающим алгоритмом и проведение повторного соединения дает меньшую погрешность ввиду своего меньшего влияния, чем проведение другого соединения.

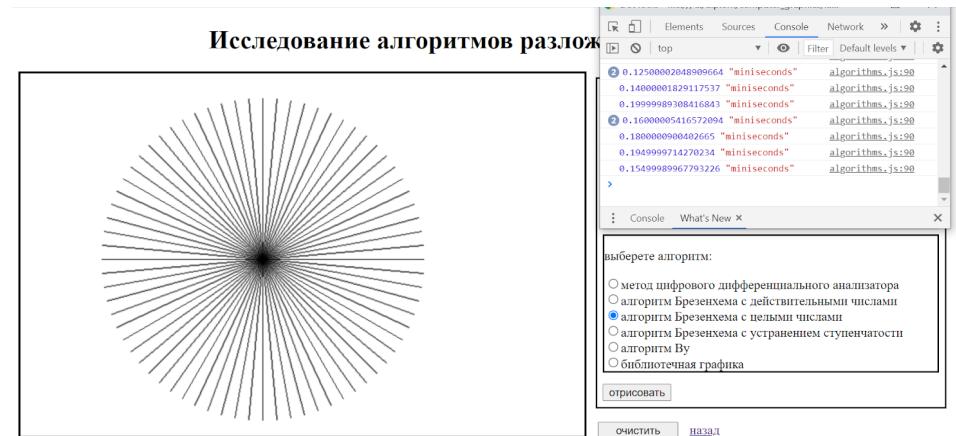
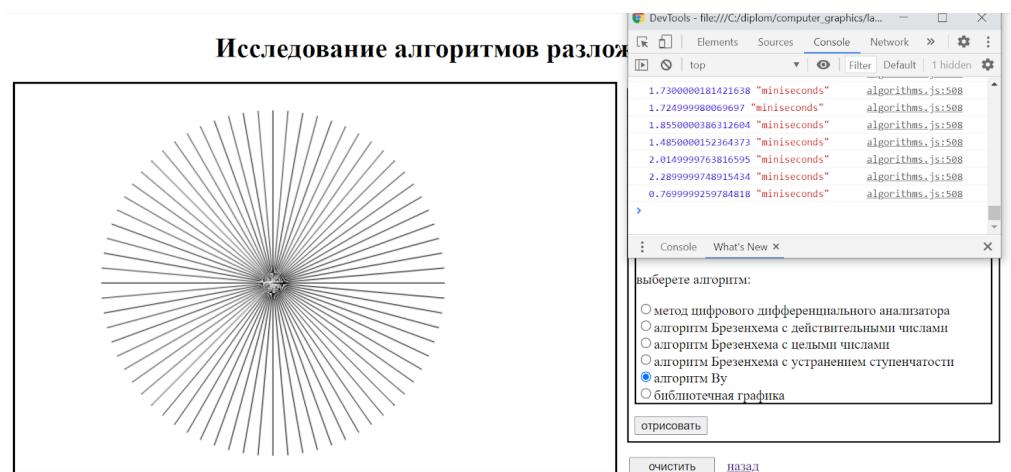


Рисунок 2.1 - Разложение отрезка в растр алгоритмом Брезенхема.

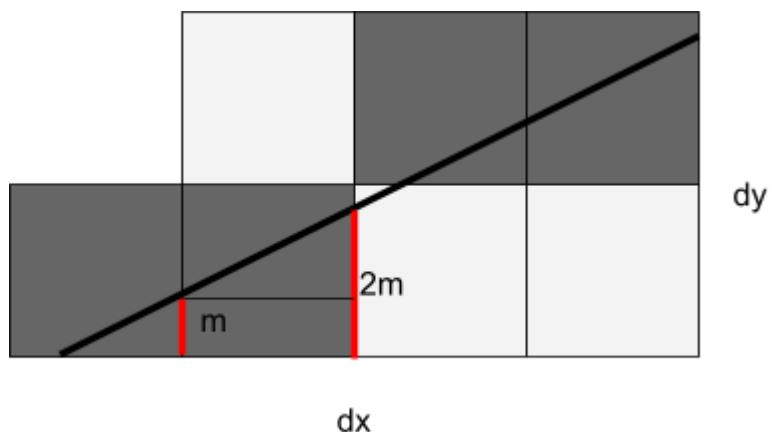
Второй алгоритм - сглаживающий с 256 уровнями интенсивности. Он используется для превью получаемого изображения. Сглаженный отрезок соответствует нити. А также для расчета общей ошибки, на основании которой останавливается цикл итераций. Результат его работы и замеры времени отрисовки отображены на рисунке 2.2.



*Рисунок 2.2 - Разложение отрезка в растр алгоритмом By.*

### Алгоритм Брезенхема

На вход алгоритм принимает координаты начальной точки  $x_n$ ,  $y_n$  и координаты конечной точки  $x_k$ ,  $y_k$  отрезка, который нужно отрисовать. Первом делом находятся длины проекций этого отрезка на оси  $dx$  и  $dy$ , а также направление распространения  $s_x$ ,  $s_y$ . Если  $dx < dy$  (то есть угол наклона прямой к горизонтали меньше 45 градусов) итерации происходят от  $x_n$  до  $x_k$  с единичным шагом в найденном направлении  $s_x$ . Закрашиваемый пиксель по другой координате  $y$  выбирается за счет ошибки. Ошибкой обозначается разница между целочисленным значением пикселя и реальным отрезком. Оно инициализируется значением  $dx / dy$  и на каждом шаге увеличивается на это же значение. Если значение больше 0.5 значение  $y$  изменяется на единицу в направлении  $s_y$ , значение ошибки уменьшается на 1. Если угол наклона отрезка больше 45 градусов, с помощью зеркального отображение, его можно привести к описанному виду.



*Рисунок 2.3 - Иллюстрация отрезка, проведенного алгоритмом Брезенхема.*

Листинг 2 - алгоритм Брезенхема.

```

ввод: xn, yn, xk, yk
x = xn; y = xn
sx = sign(xk - xn)
sy = sign(yk - yn)
dx = |xk - xn|
dy = |yk - yn|
ЕСЛИ (dx < dy), ТО { swap(dx, dy), обмен = true }
ИНАЧЕ обмен = false
m = dy / dx
e = m
ЦИКЛ (для i от 1 до dx + 1) {
    высвечивание (x, y)
    ЕСЛИ ( e >= 0.5 ) {
        ЕСЛИ (обмен), ТО {x += sx}
        ИНАЧЕ { y += sx }
        e -= 1
    }
    ЕСЛИ (обмен), ТО {y += sx}
    ИНАЧЕ {x += sx}
    e += m
}

```

Однако операции с вещественными числами трудоемки, можно преобразовать значение ошибки следующим образом. Во-первых, сравнивать удобнее с нулем, смещение добавлено в инициализацию:  $err = m - 0.5$ .

А также будет исключено деление.

Инициализация  $err = dy/dx - 1/2 \Rightarrow err = dy - dx/2 \Rightarrow err = 2dy - dx$ .

Приращение  $err += m \Rightarrow err += dy/dx \Rightarrow err += 2dy$ .

Адаптируем алгоритм под данную задачу (схема на рисунка 2.4 - 2.5). На вход функция отрисовки соединения принимает `img` - исходное изображение. Каждое отрисованное соединение вычитается из исходного изображения, таким образом решение о проведении нового соединения учитывает вклад уже отрисованных хорд. Также это обеспечивает максимальную ошибку = 255 при попытке отрисовать то же самое соединение. Алгоритм нахождение промежуточной ошибки при выборе соединения аналогичен. Ошибка равна сумме значений пикселей исходного изображения под проводимым отрезком, деленная на количество пикселей в этом отрезке для усреднения. Оба действия объединены в одну функцию, которая имеет булевый параметр разделяющий задачи.

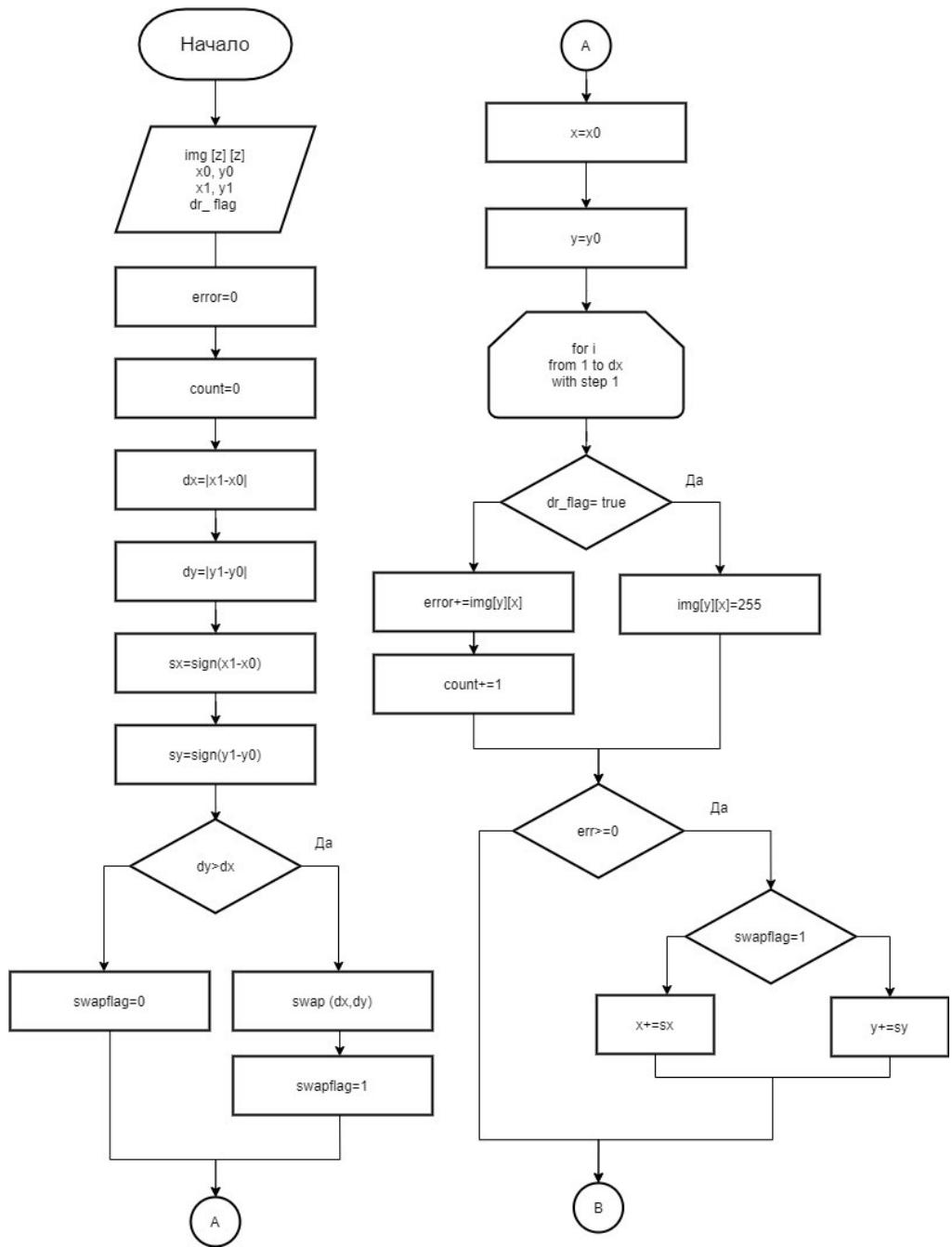
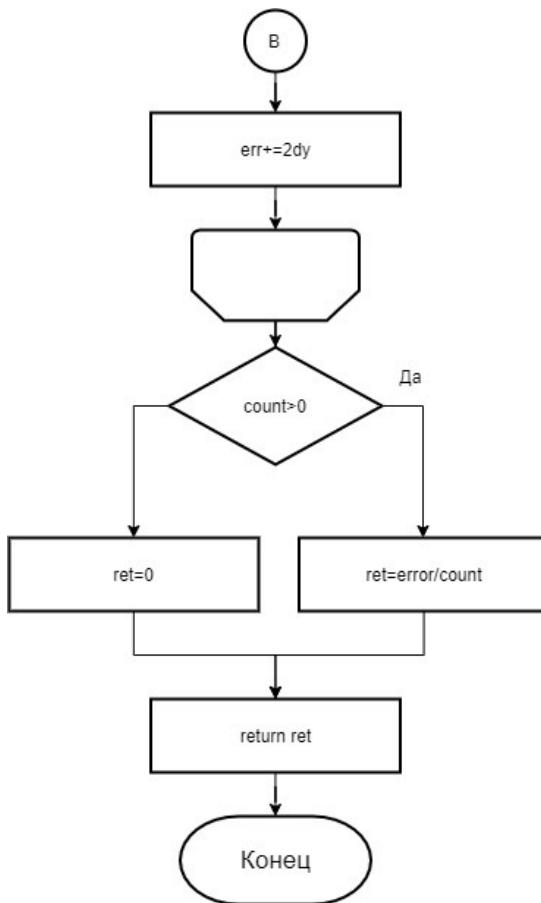


Рисунок 2.4 - Схема адаптивного алгоритма Брезенхема часть 1.



*Рисунок 2.5 - Схема адаптивного алгоритма Брезенхема часть 2.*

### Алгоритм Ву

Алгоритм Ву работает следующим образом (схема на рисунке 2.6 - 2.7).

Высвечиваются первая и последняя точка отрезка. Если отрезок горизонтальный или вертикальный, он заполняется тривиально: одна координата фиксируется, другая изменяется на единицу или минус единицу. Далее рассматриваются наклонные отрезки. Первое - угол наклона меньше 45. Если необходимо, начало и конец отрезка меняются, чтобы обеспечить положительный прирост по x. С шагом в один пиксель по оси x, увеличивается у на значение  $m = dx/dy$ . Высвечивается пиксель [x\_текущий; y\_текущий, округленный до меньшего целого] интенсивностью, пропорциональной площади под отрезком. Если интенсивность не максимальная, до дополняющей интенсивностью высвечивается пиксель сверху. В случае более вертикального наклона отрезка выполняются аналогичные действия по оси y.

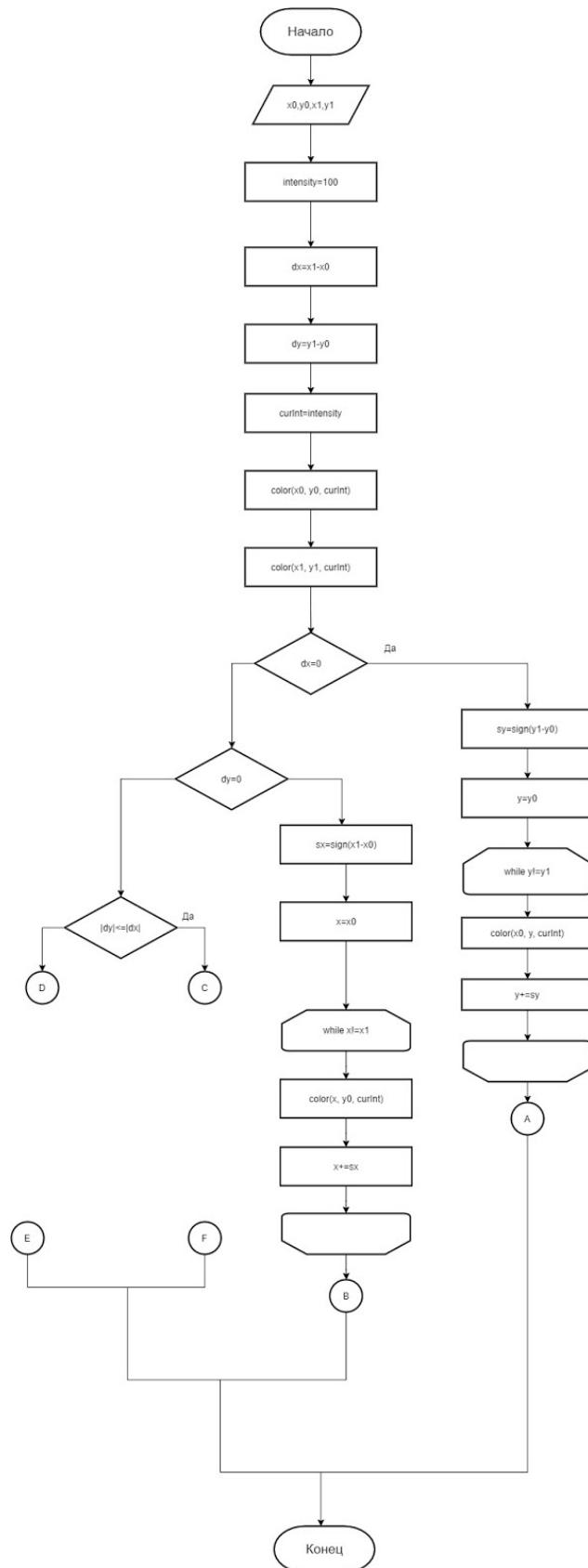


Рисунок 2.6 - Схема алгоритма Ву часть 1.

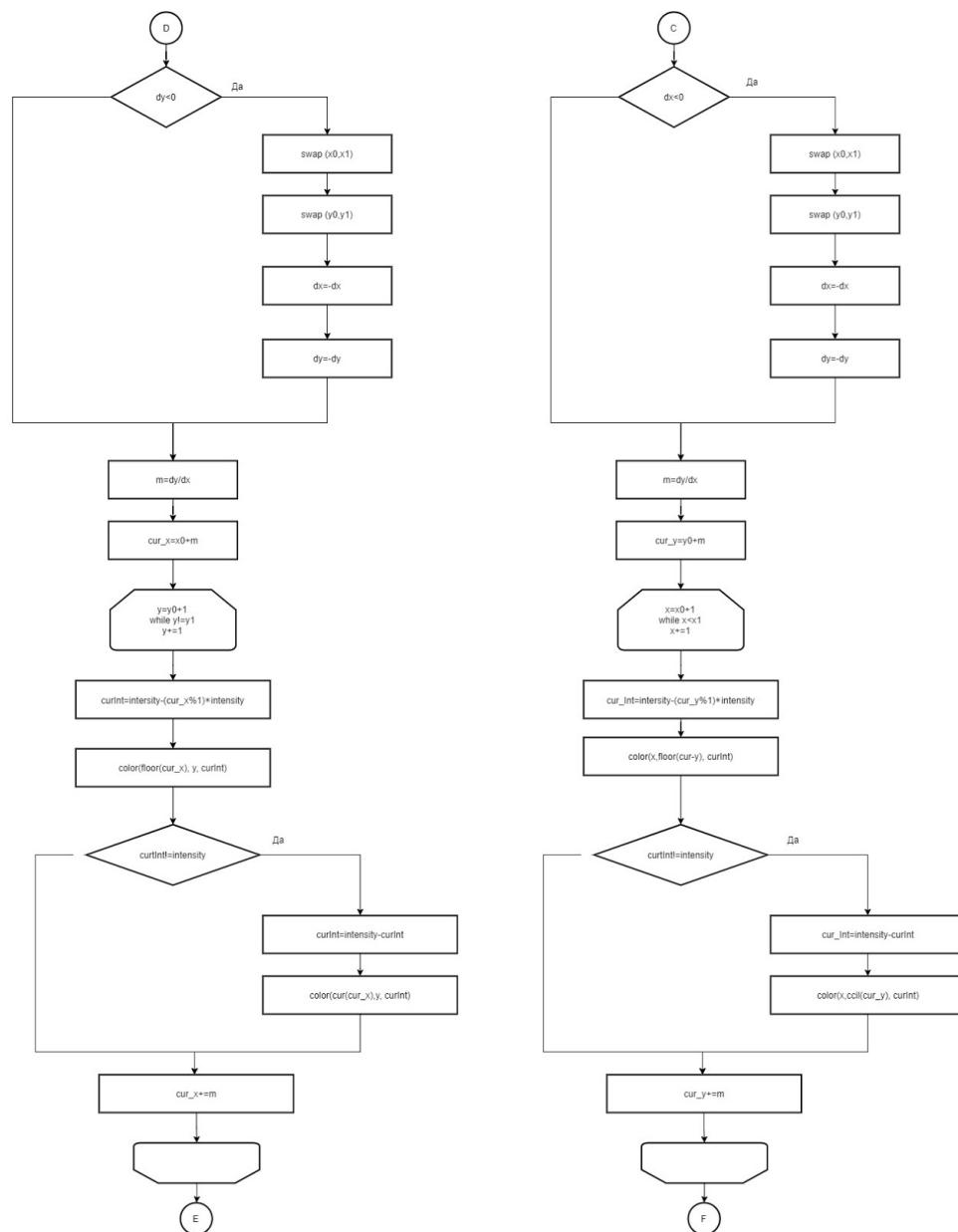


Рисунок 2.7 - Схема алгоритма Ву часть 2.

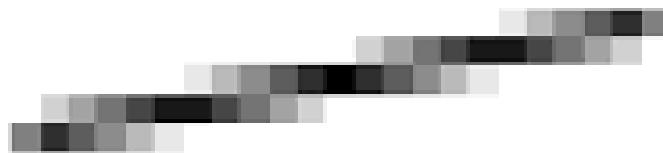


Рисунок 2.8 - Иллюстрация отрезка, проведенного алгоритмом Ву.

Адаптация под данную задачу:

1. Входные данные
  - a. Координаты гвоздей - 4 параметра
  - b. Res - результат, эта структура является полотном, на котором будет осуществляться отрисовка, полотно
2. Интенсивность
  - a. Нужно 256 оттенков. То есть максимальное значение интенсивности равно 255
  - b. При отрисовки интенсивность пикселя вычитается из имеющегося в res значения. Если оно становится отрицательным, то приравниваем его нулю, получая черный цвет. (Это же избавляет от ярко выраженного муарового узора - за счет того, что пиксель окрашивается в зависимости от произведенных рядом отрезков, отсутствует резкий перепад интенсивностей)
3. Отрисовка на канвасе - матрице
  - a. Индексация производится наоборот - сначала координата у являющаяся итератором строки в матрице, потом x - столбец

#### 2.2.4 Цикл генерации схемы

Данный алгоритм итерационный и построен на выборе лучшего соединения посредством сравнения пикселей исходного и получающегося изображения. То есть на каждой итерации из всех возможных соединений нужно выбрать то, что соответствует исходному изображению больше всех, и его отрисовать, занести в матрицу связности и учесть в формировании схеме.

Первое, что сильно ускоряет процесс генерации это перебор не всех возможных соединений, а только тех, что выходят из последнего гвоздя. Так как соединение из него в любом случае будет произведено ввиду непрерывности нити, можно ограничиться перебором соединений только из последней точки, а не всех возможных. Также этот вариант исключает последующее преобразование проведенных соединений в схему и поиск эйлерового пути.

Теперь необходимо выбрать начальный гвоздь. В некоторых реализациях берется просто первый или случайны. В таком случае всё плетение сходится к локальному минимуму. Первый гвоздь очевидно является одним из концов лучшего соединения из всех возможных. Так что однажды все-таки придется перебрать все соединения.

Когда такая начальная хорда найдена, нужно определить который из ее концов будет первым, а какой вторым, так как эйлеров цикл не гарантируется. Выбор делается следующим образом - из обеих точек ищется лучшее соединение (помимо уже найденного) и выбирается то, что дает меньшую ошибку. Таким образом определяются первые три пункта схемы плетения.

Далее в цикле ищется и проводится следующее лучшее соединение, идентификатор гвоздика заносится в файл.

Каждые несколько соединений происходит расчет общей ошибки - сглаживающим адаптированным алгоритмом отрисовываются все произведенные соединения и полученное изображение вычитается из исходного. Берется сумма модулей значений разницы каждого пикселя и делится на количество всех пикселей. Если это значение больше предыдущего *whole\_error* цикл завершается. Оптимальное количество проводимых соединений найдено, дальнейшее добавление хорд ведет к увеличению ошибки.

После цикла происходит формирование и сохранение всех результатов работы алгоритма. Сглаживающим алгоритмом рисуется картина и сохраняется в файл. Сохраняется схема, а в ее конце добавляется аналитическая и конфигурационная идентифицирующая картину информация (название файла, радиус полотна, количество гвоздей, толщина нити, минимальная хорда, разрешение, количество всех возможных соединений, количество проведенных соединений, общая длины нити, общая ошибка, время генерации).

### 2.3 Пользовательские возможности

Чтобы включить / выключить логирование пользователь может задать параметр *if\_log*.

Пользователь может включить опцию просмотра процесса генерации if\_show. Тогда при расчете общей ошибки, текущее состояние картины будет выводиться на экран.

Все параметры задаются в конфигурационном файле вида:

Листинг 3 - пример конфигурационного файла.

```
Rc = 30
t = 0.05
m = 300
skip = 1
if_log = 1
if_show = 0
img_path = "image.png"
```

## 3 Технологический раздел

В данном разделе будут описаны средства программной реализации. Их достоинства и конкретные применения. Описан способ сборки и запуска приложения. А также приведен пример работы программы и результаты тестирования отдельных этапов.

### 3.1 Средства программной реализации

#### 3.1.1 Язык программирования

Для тестирования и анализа работы алгоритма был выбран язык программирования python [15]. Python - гибкий, расширяемый многими модулями язык программирования. Существуют библиотеки и фреймворки под любой тип задач и надобностей. Подходит для обработки больших данных и простого отображения картинок. Язык обладает простым понятным синтаксисом, что делает закодированный алгоритм представительным. Python является интерпретируемым языком, несложен в установке и запуске.

### 3.1.2 Используемые библиотеки

- Numpy [16]

Для работы с матрицами, которые в задаче представляют холст, используется numpy. Numpy — это расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокогоуровневых математических функций для операций с этими массивами.

```
pip install numpy      - для установки  
import numpy          - для подключения в код
```

Numpy массивы - это плотно упакованные в памяти специализированные массивы однородного фиксированного типа. Python списки, напротив, являются массивами указателей на объекты, что занимает гораздо больше памяти и обрабатываются дольше.

Кроме того, многие операции Numpy реализованы в C, что позволяет избежать общих затрат на циклы в Python, косвенное обращение указателя и динамическую проверку типов по элементам. Numpy также является быстрым инструментом, потому что он использует преимущества параллелизма, в то время как традиционный for loop не может его использовать.

Для инициализации структуры pins используется метод linespace. Матрица результирующего изображения инициализируется с помощью numpy.full. Для значения пикселя используется однобайтовый тип numpy.uint8. Структуры res и img таким образом занимают ровно ZxZ Байт. Для поиска ошибки используются numpy.sum, numpy.abs.

- PIL [17]

Для обработки изображений используется библиотека PIL. Не нуждается в дополнительной установке. Поддерживает взаимодействие с numpy. Подключается и используется она следующим образом.

#### Листинг 4 - функция обработки изображения.

```
from PIL import Image, ImageDraw, ImageTk

def parse_image(image_path_in, Z):
    # load
    image = Image.open(image_path_in)

    # square crop
    if image.size[0] < image.size[1]:
        side = image.size[0]
    else:
        side = image.size[1]
    left = (image.size[0] - side) / 2
    top = (image.size[1] - side) / 2
    right = (image.size[0] + side) / 2
    bottom = (image.size[1] + side) / 2
    image = image.crop((left, top, right, bottom))

    # greyscale
    image = image.convert('L')

    # restore matching resolution
    image = image.resize((Z, Z), Image.ANTIALIAS)

    # circle crop
    circlecrop = Image.new('L', [Z, Z], 255)
    circle = ImageDraw.Draw(circlecrop)
    circle.pieslice([1, 1, Z - 1, Z - 1], 0, 360, fill=0)
    npcrop = np.array(circlecrop)
    img = np.asarray(image).copy().astype(np.uint8)
    img[npcrop == 255] = 255

    return img
```

Сохраняется и отображается картинка просто с помощью двух методов `save` и `show`.

#### Листинг 5 - функция сохранения и отображения изображения.

```
def save_image(filename, img):
    Image.fromarray(img).save(filename)

def show_image(img):
    Image.fromarray(img).show()
```

- Tkinter [18]

Для отображения промежуточных результатов генерации используется tkinter. Tkinter – это кроссплатформенная библиотека для разработки графического интерфейса на языке Python (начиная с Python 3.0 переименована в tkinter со строчной буквы). Tkinter входит в стандартный дистрибутив Python.

Подключается и используется следующим образом совместно с Numpy и PIL.

Листинг 5 - отображение промежуточного результата.

```
from tkinter import Tk, Canvas

def show_tk_img(root, canvas, nparr):
    w = len(nparr) / 2
    image = ImageTk.PhotoImage(Image.fromarray(nparr))
    imagesprite = canvas.create_image(w, w, image=image)
    canvas.pack()
    root.update()

root = Tk()
canvas = Canvas(root, width=Z, height=Z)
canvas.pack()

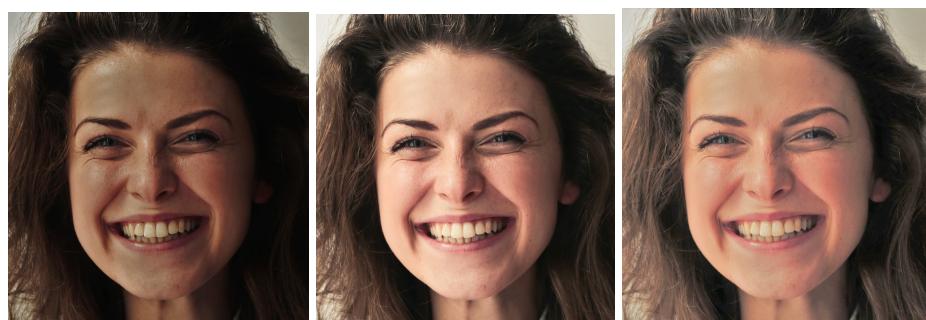
... in the main loop ...
show_tk_image(root, canvas, res)
...
root.mainloop()
```

### 3.2 Запуск приложения

Конфигурационный файл config.txt должен быть в папке. Команда python stringart.py начнет генерацию схемы.

### 3.3 Тестовые данные и данные для экспериментов

Разный цвет, яркость и контрастность исходного изображения [19].





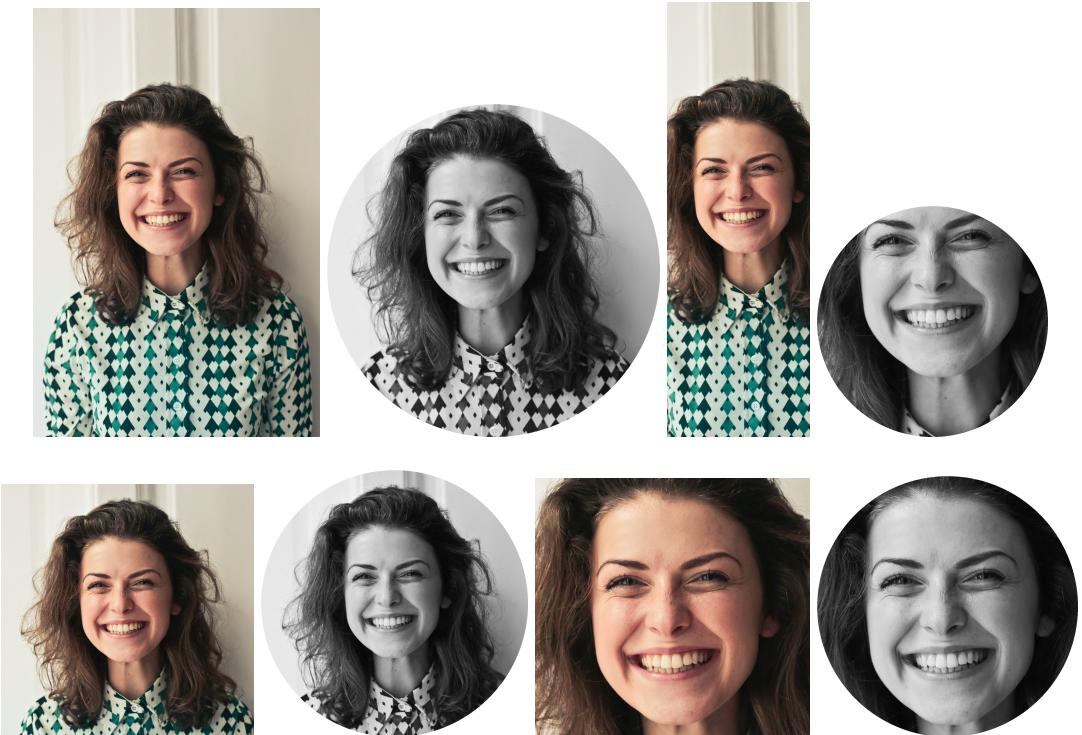
*Рисунок 3.1 - Примеры сходных данных.*

Таблица 2 - Параметры и диапазоны для экспериментов и анализа их влияния.

	Радиус холста	Толщина нити	Кол-во гвоздей	Мин. хорда
Изменения диаметра холста	10 - 40	0.1	300	30
Изменение толщины нити	25	0.02 - 0.3	300	30
Изменение количества гвоздей	25	0.1	50-400	5
Изменение значения минимальной хорды	25	0.1	300	1-50

### 3.4 Проверка обработки изображения

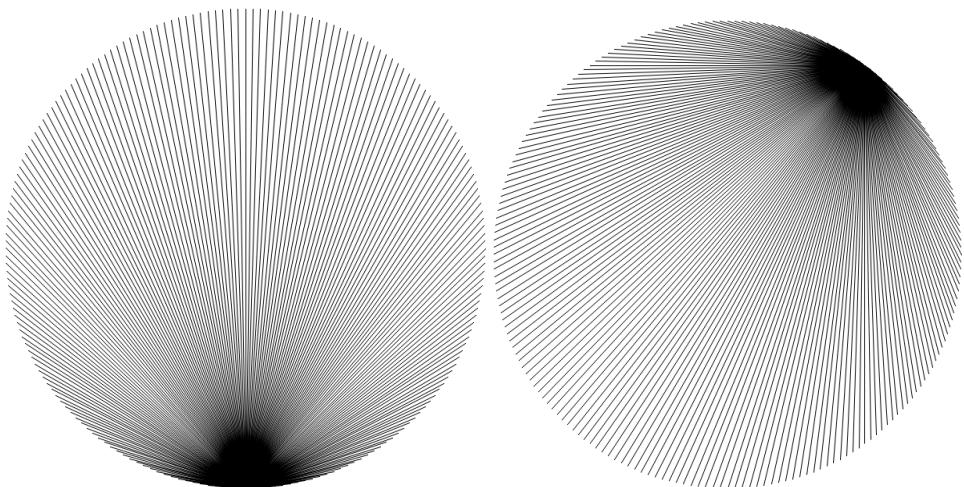
Ожидается, что любое входное изображение будет обрезано по кругу, где центр совпадает с исходным, а диаметр равен меньшей из сторон. Углы изображения должны быть белыми. Изображение переведено в градацию серого, неискаженное и имеет в результате правильное разрешение.



*Рисунок 3.2 - Примеры обработки входных данных.*

### 3.5 Результат отрисовки соединений

На рисунке 3.3 приведен результат отрисовки соединений одного гвоздя со всеми другими адаптированным методом Ву, в соответствии с описанным выше пунктом 2.2.3 “Отрисовка отрезков и расчет ошибки”.



*Рисунок 3.3 - Результат отрисовки соединений одного гвоздя со всеми другими адаптированным методом By.*

### **3.6 Проверка определения оптимального количества соединений**

Белый квадрат - ожидается 0 соединений, ошибка 0.0.

Черный квадрат - ожидается много соединений, ошибка меньше 1.5.

Черные полосы на белом фоне - ожидается, что будет проведена достаточно точная полоса в правильной ориентации.

Сpirаль предположительно будет работать плохо, так как является невозможной для описанной аппроксимации изображением. Любое соединение, проходящее по спирали, одновременно проходит через самый темный и самый светлый участок.

### **3.7 Пример работы**

Радиус холста - 30 см, количество гвоздей - 30, толщина нити - 0.05.

Параметр минимальной хорды для ускорения генерации выбран равным 15.

Разрешение получается 1200x1200 пикселей, количество всех возможных соединений равно 40650.



*Рисунок 3.4 - Входное изображение, обработанное, превью картины.*

```
sch_image1.txt - Notepad
File Edit Format View Help
191 93 21 218 135 214 129 217 291 233 192 280 198 101 201 109 78 145 82 171 84 105 189 103 201 100 32 47 231 44 224 35 ^ 246
200 connections made
29 226 39 246 178 86 104 195 117 194 280 67 2 58 289 67 287 198 113 207 294 215 130 212 292 223 46 239 294 59 1 209
298 74 152 78 142 80 151 79 163 83 106 186 112 45 221 35 228 291 235 12 64 271 63 273 66 296 209 132 80 145 73 0 207 3
56 8 65 267 68 0 232 17 57 40 219 127 215 132 84 163 78 164 82 141 221 37 228 20 240 176 248 172 249 14 216 286 60 274
300 connections made
59 279 55 236 25 232 30 224 39 220 30 49 117 78 155 73 152 75 294 211 131 85 160 238 177 239 21 247 24 240 7 232 34
229 50 21 232 288 71 153 82 135 211 120 194 109 198 100 199 103 87 18 216 62 218 128 81 146 76 291 6 67 273 66 1 60 8
218 41 56 209 135 78 146 77 152 72 5 232 39 215 134 83 140 215 35 222 291 227 26 232 12 58 215 63 224 27 215 30 246
400 connections made
67 267 64 238 216 126 85 116 197 111 189 97 198 107 205 2 232 32 102 189 109 186 117 197 109 42 215 292 233 296 74 299
163 80 149 290 235 190 281 54 20 251 171 250 172 247 20 51 227 62 267 69 146 67 293 76 142 82 118 186 114 189 110 41
223 65 18 98 199 101 189 93 15 243 58 276 59 213 63 255 176 244 7 290 196 280 199 99 201 105 89 14 57 20 242 11 207
280 58 224
500 connections made
38 106 198 111 187 237 54 28 224 31 48 221 60 270 63 217 42 221 135 215 56 25 239 35 227 13 130 87 169 82 140 225 22
247 176 241 295 153 294 103 192 281 92 22 1 56 217 237 9 243 34 224 37 214 141 80 144 78 151 75 146 70 246 55 232 10
72 154 75 157 225 24 228 32 221 67 224 36 232 16 31 246 49 221 39 213 142 79 113 86 137 82 133 222 140 218 61 274 39
223
600 connections made
59 211 63 214 148 226 152 295 238 47 223 40 145 280 195 102 202 50 116 186 120 185 122 213 56 13 224 25 247 19 232 297
60 12 28 44 215 129 58 0 108 189 117 83 131 78 165 79 121 185 45 219 59 246 53 203 108 82 139 80 146 68 148 82 138 31
230 289 8 232 14 251 19 95 275 0 162 298 76 151 229 39 218 50 231 193 280 144 82 163 81 120 215 65 243 29 225 186 115
```

*Рисунок 3.5 - Результатом работы алгоритма - схема плетения.*

```
sch_image1.txt - Notepad
File Edit Format View Help
1400 connections made
121 182 132 26 230 22 196 274 62 295 224 239 33 227 27 57 24 250 100 203 99 251 168 24 243 223 54 2 101 50 219 60 211
61 170 82 241 90 219 47 229 35 53 216 56 207 65 143 289 240 29 221 41 227 47 220 59 217 91 207 63 203 87 265 21 37 227
50 187 109 50 289 60 282 93 125 72 87 213 64 201 104 197 107 199 112 187 113 194 134 184 127 279 129 220 53 217 61 270
285
1500 connections made
87 180 47 6 277 12 252 18 243 74 202 65 270 35 235 24 294 43 224 45 7 138 81 207 84 293 58 223 191 115 94 191 108 193
267 85 200 67 211 92 201 95 216 63 261 78 138 185 116 268

image path: image1.png
canvas radius (cm): 30
numer of pins: 300
thread thickness (cm): 0.05
pins to skip in minimum connection: 15
logging: 1
showing process: 0
resolution: 1200
number of all possible connections: 40650
1550 connections made out of 40650
672.7712679991622 m of thread needed
error is 56.5808875
time: 534.8148124217987
```

*Рисунок 3.6 - Конфигурационные и аналитические данные в конце схемы.*

Сделано было 1550 соединений из 40650. Для плетения этой картины потребуется 672.7 метров нити одноцветной непрозрачной нити.

Общая ошибка полученного изображения равно 56.5808. Генерация, логирование и отрисовка заняли 534.814 секунд - 8.9 минут.

## **Заключение**

Было проведено аналитическое исследование существующих методов для автоматического создания картин стринг-арта. Изучены составляющие и параметры вида нитяной графики, которому свойственны круглое полотно, расположение гвоздей по окружности с одинаковым интервалом. Обоснована актуальность создания алгоритма.

Разработан алгоритм автоматической генерации схемы для создания стринг-арта. Алгоритм получает на вход цифровую картинку, а возвращает последовательность натяжений нити. В работе описаны используемые структуры данных и методы.

В работе представлена реализация алгоритма на языке программирования python. Были подготовлены тестовые и экспериментальные данные. Функциональность программного продукта была протестирована.

## **Источники**

1. Анонс выставки работ Ани и Андрея Абакумовых в галерее ArtStory. - Текст: электронный // art-story : [сайт]. - 2021. - URL: <https://www.art-story.com/events/items/ani-i-andrey-abakumovyi-uskolzayushaya-nit/> (дата обращения: 28.05.21)
2. Студия LAARCO. - Текст: электронный // LAARCO : [сайт]. - 2021. - URL: <https://laarco.com/> (дата обращения: 28.05.21)
3. Компания Art Pautina. - Текст: электронный // ArtPautina: [сайт]. - 2021. - URL: <https://artpautina.ru/> (дата обращения: 28.05.21)
4. Статья Петроса Вреллиса о своей работе. - Текст: электронный // Статья Петроса Вреллиса о своей работе: [сайт]. - 2021. - URL: <http://artof01.com/vrellis/works/knit.html> (дата обращения: 28.05.21)
5. Статья о творчестве Ани и Андрея абакумовых. - Текст: электронный // awesomebyte : [сайт]. - 2021. - URL: <https://awesomebyte.com/string-art-designs-ideas-russian-artist/> (дата обращения: 28.05.21)
6. Пример автоматического создания картины в технике стринг арт youtube-канале Бартона Дринга. - Текст: электронный // Youtube : [сайт]. - 2021. - URL: <https://www.youtube.com/watch?v=M1gXuKFspgY> (дата обращения: 28.05.21)
7. StringArt. - Текст: электронный // Github : [сайт]. - 2021. - URL: <https://github.com/bdring/StringArt> (дата обращения: 28.05.21)
8. Проект Рафаэль Шаафа. - Текст: электронный // Hackaday : [сайт]. - 2021. - URL: <https://hackaday.io/project/130951-diy-knit-portrait> (дата обращения: 28.05.21)
9. Исполняемые файлы работы Рафаэль Шаафа. - Текст: электронный // Drive.google : [сайт]. - 2021. - URL: [https://drive.google.com/drive/folders/1nbvir\\_xkV5CKTdpKAWuU1EC1cfDbtfxq](https://drive.google.com/drive/folders/1nbvir_xkV5CKTdpKAWuU1EC1cfDbtfxq) (дата обращения: 28.05.21)

- 10.Проект shlonkin. - Текст: электронный // Hackaday : [сайт]. - 2021. - URL: <https://hackaday.io/project/13047-automate-the-art> (дата обращения: 28.05.21)
- 11.Исходный код проекта shlonkin. - Текст: электронный // Github: [сайт]. - 2021. - URL: <https://github.com/ericheisler/AutomatedArt> (дата обращения: 28.05.21)
- 12.Исходный код реализации в вебе. - Текст: электронный // Github: [сайт]. - 2021. - URL: <https://github.com/halfmonty/StringArtGenerator> (дата обращения: 28.05.21)
- 13.Исследовательская работа “String Art: Towards Computational Fabrication of String Images” - EUROGRAPHICS 2018 / D. Gutierrez and A. Sheffer - 12c.
14. Курс компьютерной графики МГТУ им. Баумана - Куров А.В.
- 15.Python. - Текст: электронный // Python : [сайт]. - 2021. - URL: <https://www.python.org/> (дата обращения: 28.05.21)
16. Numpy. - Текст: электронный // Numpy : [сайт]. - 2021. - URL: <https://numpy.org/> (дата обращения: 28.05.21)
- 17.Pillow (PIL Fork) 8.2.0 documentation - Текст: электронный // Pillow : [сайт]. - 2021. - URL: <https://pillow.readthedocs.io/en/stable/> (дата обращения: 28.05.21)
- 18.Tkinter. - Текст: электронный // Tkinter: [сайт]. - 2021. - URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения: 28.05.21)
- 19.Стоковое фото. Pexels: [сайт] - 2021. - URL: <https://www.pexels.com/ru-ru/photo/774909/> (дата обращения: 28.05.21)