

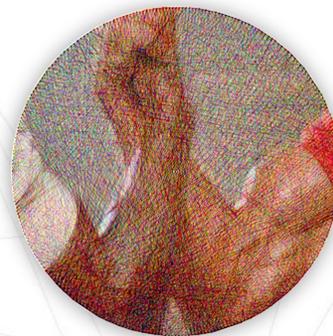


STRING ART

<https://github.com/kaspar98/StringArt>

Kaspar Valk, Kert Tali

Institute of Computer Science



IT Akadeemia



Introduction

You have a photo, a collection of fixed nails and an endlessly long string. In what order do you pull and loop the string around nails so that the end result resembles the initial picture as closely as possible? The form of handicraft creating art with this method is called string art. Creating simplistic images with string is quite easy, but how do you do it when trying to create more complex images with shades, small details, and color mixing?



Implementation Details

Color

The program can be easily made to generate a colored output. Separate your image into color channels and handle each color as a distinct image. Each result can then be pulled with the corresponding color string on the same canvas. Color mixing can be done in several ways: blending, making the top string override previous ones, or combining both ways. Different final goals need different mixing techniques. Whether it is presented digitally or physically — what is the background, does the light shine through the canvas or onto it.

Downscaling

Small resolution fits well for semi-transparent strings on which the algorithm can generalize. The order of string pulls can later be projected on a larger resolution to give a more refined image.

Speed

Figure 1 consists of 282 nails and 3000 string pulls on a 300px diameter circle. Such an example took 0.028 seconds per iteration, totalling 84 seconds. The algorithm can be sped up without much decrease in quality by introducing some randomness when selecting the next nail.

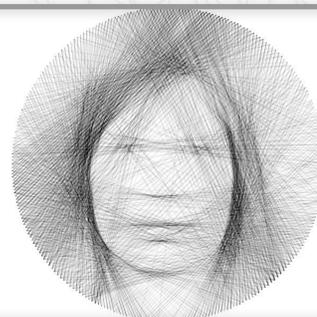
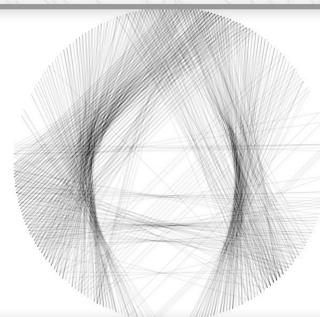
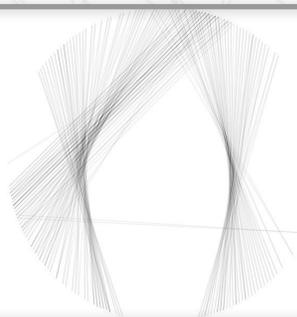


Figure 1. Portrait progression (source, 200, 600, 1200, 3000 iterations)

Algorithm

We tackled the problem with a simple greedy approach

- 1) Evaluate the goodness of every possible string pull from the current nail.*¹
- 2) Pull the string to the best found nail.
- 3) Set the new nail as the current nail.
- 4) Repeat the algorithm.*²

*¹ To evaluate the goodness of a string pull, we evaluated pixel by pixel, how much better or worse does the pull make the picture. Distance of two pixels was square difference. The best line is the one with most cumulative improvement.

*² The number of iterations can be set manually or the algorithm can be set to repeat until it fails to find string pulls that improve the total result.

The algorithm has a time complexity of

$$O(\text{nail_distance} \times \text{nails} \times \text{iterations})$$

