Research Skills and Methodologies
Path Optimization of 3D Printer

Michał Kowalski, 195447
Piotr Lechowicz, 195651

1 Introduction

Nowadays 3D printing is one of the fastest developing technologies. It allows us to convert digital 3D models into solid 3 dimensional objects. It is commonly used in prototyping because it allows to make 3D object without a need to use forms. The main advantages of 3D printing technology are: low cost of the printer, low cost of printing object, variety of materials and variety of technologies. To use this technology, we need to:

1. create a 3D model of our object,
2. "cut" (convert) it into set of very thin layers,
3. choose path for printing tool.

Our path should consist of points from layer currently being printed. Tool have to visit each point exactly once and can move without printing. Task is to minimize "cost" of moving between all succeding points in our path. We can calculate it for each pair in (at least) three ways:

1. distance between points,
2. time needed for move from one to next,
3. energy needed for that move.

We are assuming that our printing tool is a point moving on two perpendicular axes.

Another task was to write an environment, which will deliver layers for algorithms, provide methods commonly used in finding path, and will save and show results in human-friendly way, including graphical representation of found path. It should also be open to addition of new algorithms.

2 Mathematic description of problem

Given:

- printing layer as an array (size n x m) of binary points, where

$$X_{i,j} = \begin{cases} 1 & \text{if printing point} \\ 0 & \text{otherwise} \end{cases} \quad \text{and} i \in n, j \in m \tag{1}$$

- cost for move between two points calculated in one of three possible ways:
  - for minimum distance

$$L_{P_{a,b},P_{c,d}} = \sqrt{(a-c)^2 + (b-d)^2} \tag{2}$$

  - for minimum time (only axis which need to move more)

$$L_{P_{a,b},P_{c,d}} = max(|a-c|, |b-d|) \tag{3}$$

  - for minimum energy (both axes has separate engines)

$$L_{P_{a,b},P_{c,d}} = |a-c| + |b-d| \tag{4}$$

  where $a, c \in n$ and $b, d \in m$.

Find:

- order of visiting points $V = [X_1, X_2, ..., X_p]$ where p is count of points in layer,
- total cost of path consisting of visiting points $L = \sum_{k=2}^{p} (L_{X_{k-1},X_k})$,
- time of calculations for each considered algorithm,

such that length of path and time of calculations are minimized.

3 Algorithms

Following algorithm were implemented:

- Left-To-Right algorithm
- Snake algorithm
- Edge Following algorithm

All implementations of algorithms are based on superclass, which makes all necessary communication between environment and algorithm, like:

- setting up an algorithm, e.g. getting a layer to print,
- sending results of algorithm to rest of application,
- measuring time of calculations.

Therefore algorithms only have to do all calculations in method returning a path, and optionally set up all variables in separate method. It also simplifies adding new algorithms, because all they have to do is implement this method, and they have to be added to list of algorithms.

3.1 Left-To-Right algorithm

This algorithm was created mostly for testing purposes. It had to be as simple as possible, and still find correct route, so it allows to test how environment works with algorithms in general.

Algorithm starts in top-left corner and goes line-by-line from left to right, adding to route all points on its way. And because only point to print are on found route, printer will go straight from last point in one line, to first in next one containing any points. Behaviour of this algorithm is similar to traditional paper printes.

3.2 Snake algorithm

This algorithm is an improvement of Left-To-Right algorithm. Instead of going always from left to right, it finds a closer end of next line containing points, and then goes to another end of it. Differences between them are show on figure 1

3.3 Edge Following algorithm

This algorithm starts from edge of a shape on a layer, and tries to get to the middle of it, and later choose next shape.

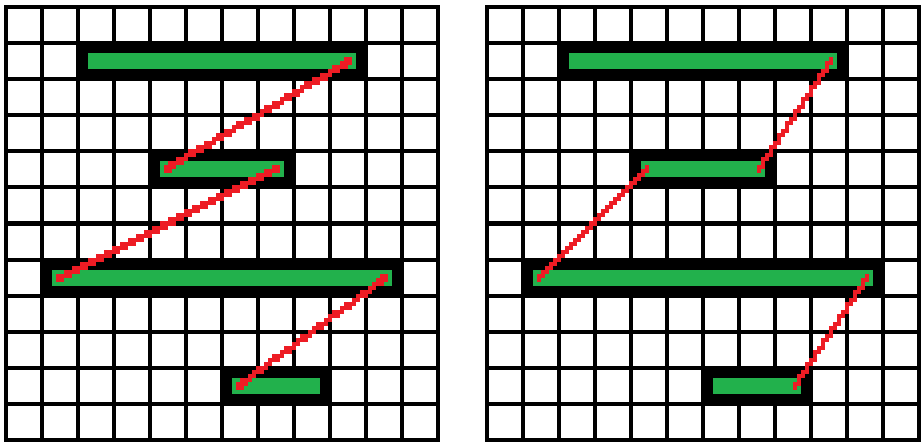

Figure 1: Left-To-Right algorithm on the left, Snake algorithm on the right

4 Experiments

4.1 Environment

4.2 Test layers

Images shown on figures 2 - 10 were used as test layers.

5 Results

For very short calculations, measured time can have very big differences between one measurement and another one with the same algorithm/params.

6 Conclusion

As we can see on figure 11, when algorithm is seeking for closest point, it finds different ones, depending on cost function type. Therefore, if algorithm makes use of this, the results will vary.

Snake algorithm usually provides almost 2 times better results than Left-to-Right, in similar time. For some layers it also provides optimal (or close to it) paths, whereas for other it can give results multiple times worse than optimal.

Not only count of points, but also size of whole layer (even if points are only in the middle) can have influence on results or time of calculations, e.g. when algorithm is searching for closest point.

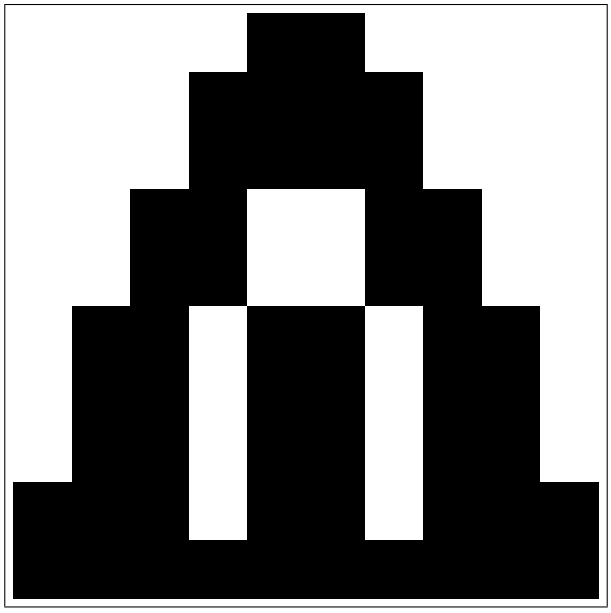

Figure 2: sp-1

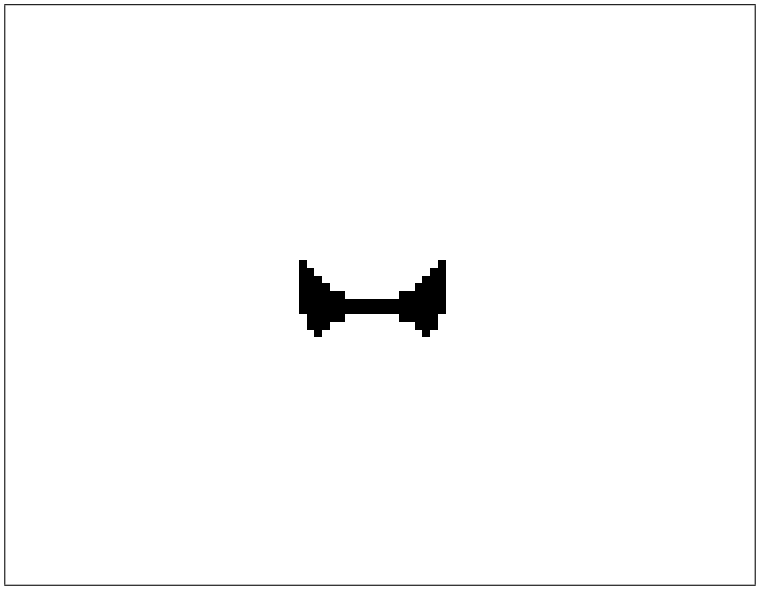

Figure 3: sp-2

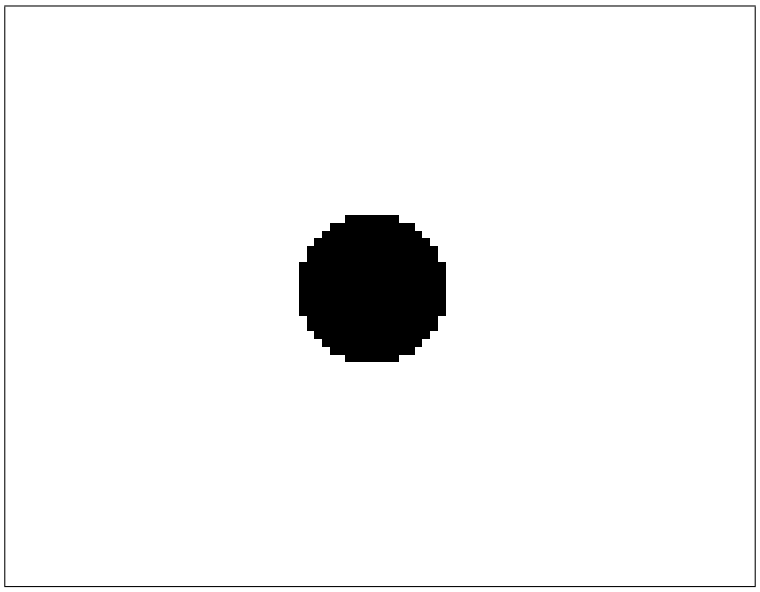

Figure 4: sp-3

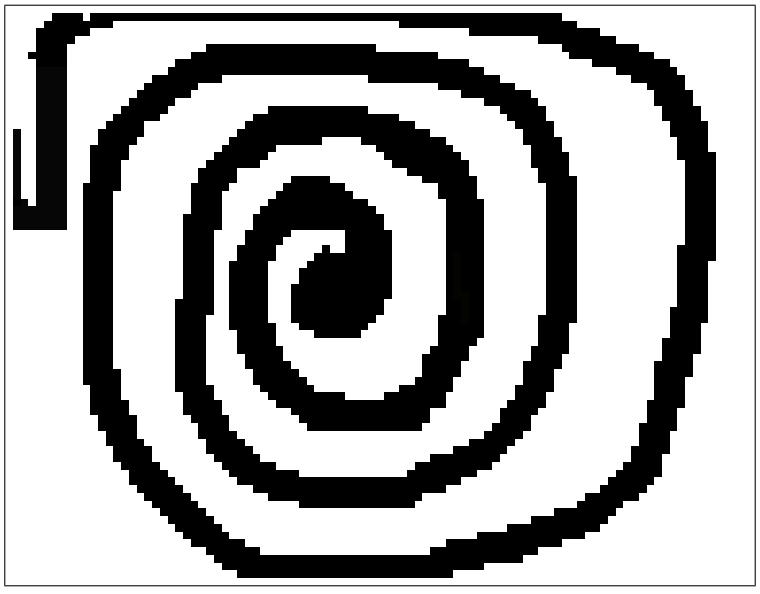

Figure 5: sp-4

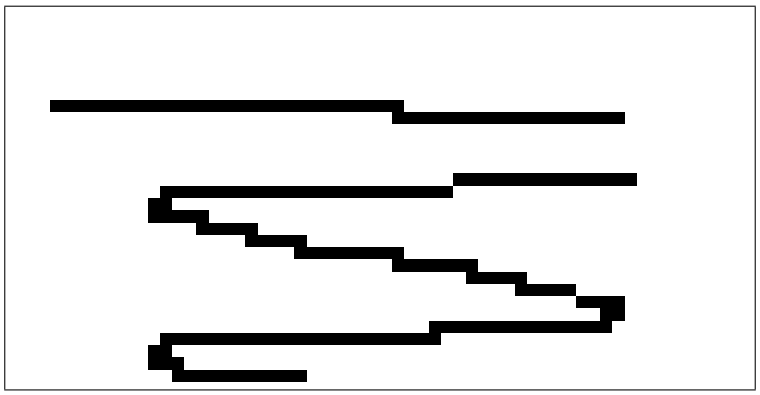

Figure 6: sp-5

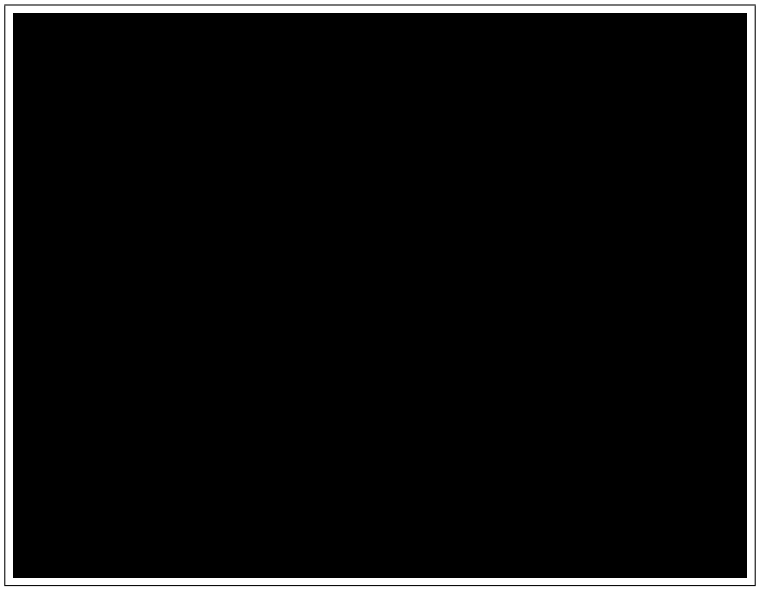

Figure 7: sp-6

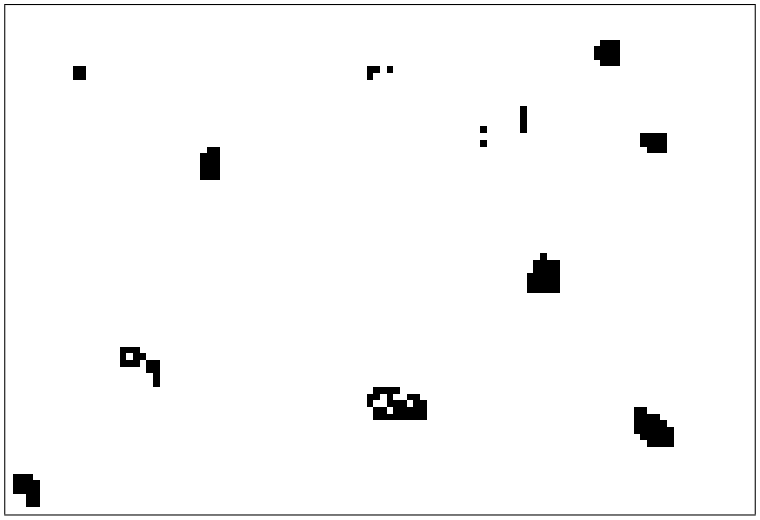

Figure 8: sp-7

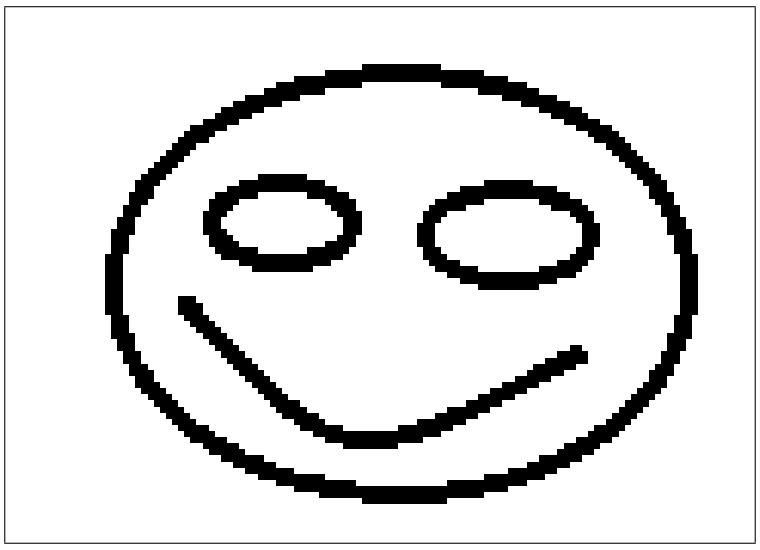

Figure 9: sp-8

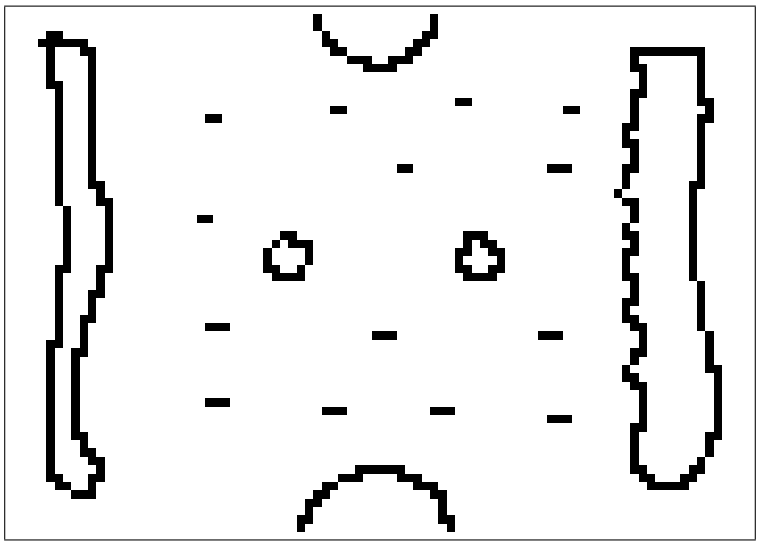

Figure 10: sp-9

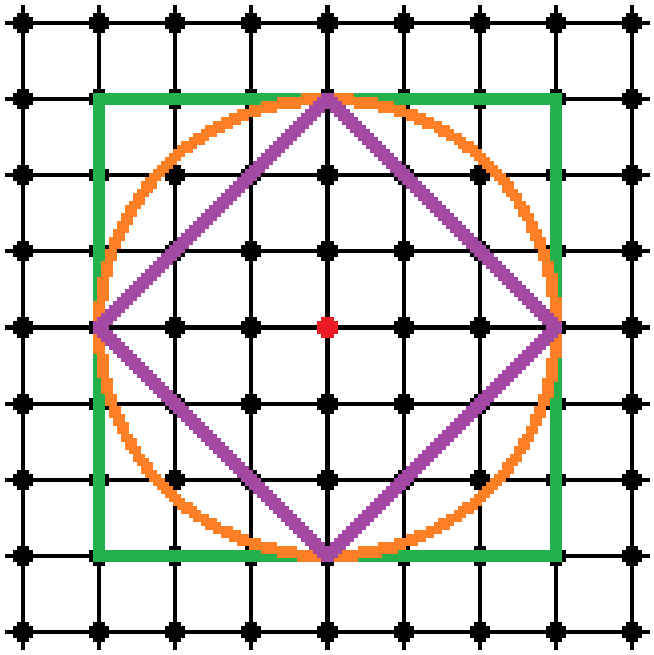

Figure 11: Points with cost from red point equal to 3.
Orange for distance, green for time, and violet for energy

7 Own contribution

Michał's contribution:

- application for tests and graphical representation of results,
- Left-To-Right algorithm,
- Snake algorithm,
- Edge-Following algorithm,
- layers sp-5 and sp-6,
- time and energy cost functions,
- experiments and this report.

Piotr's contribution:

- layers from sp-7 to sp-9,
- experiments and this report.

8 References