A Slime Mold System Driven by Skeletonization Errors

Yufan Xie[1], Jingsen Lian[2], and Yufang Zhou[2(✉)]

[1] University of Southern California, 850 Bloom Walk, Los Angeles, CA 90089, USA
[2] Central Academy of Fine Arts, Huajiadi South Street 8, Beijing 100102, China
zhouyufang@cafa.edu.cn

Abstract. This paper proposed a new way to generate slime mold patterns using a typical voronoi-based skeletonization method. As a recursive system, it redraws and expands the resulting trails of skeletonization and feeds them back as an image source for skeletonization. Through iterations, it utilizes the difference before and after skeletonization to generate slime-mold-like patterns. During the whole process, we tested different growth types with different parameter settings and environmental conditions. Since most researches on skeletonization focus on minimizing errors, on the opposite side this method utilizes errors of skeletonisation (e.g. subtracted skeletons at "branch" areas of the bitmap are different from the original brush trails or the best result we expect) as the basis of the generative process. The redraw process makes it possible to reconnect skeletons via intersected brushes, continuously changing the topology of the network. Unlike the traditional slime mold algorithm which operates on every single agent, our method is driven by image-based solutions. On the output side, this system provides a condensed vector result, which is more applicable for design purposes.

Keywords: Slime mold · Physarum · Skeletonization · Generative · Error

1 Background

Skeletonization algorithms are generally used in pattern recognition and image subtraction. There are many precedents on pixel-based and vector based models, ranging from 2 to 3D space. By far, there is no known exploration using the skeletonization process as a generative system. Meanwhile, to realize the slime mold system, a general method is using an agent based model to simulate the process. (The physarum model by Jones [1] is one of the most widely used frameworks.) A collection of points are defined, with detection ranges and movements, reacting to each other to form a connected dynamic network like an ant farm. These two algorithms—one for subtraction and another for generation—seem to be non-related in most conditions. Our research discovered a new simple, but effective way to integrate both through errors.

P. F. Yuan et al. (eds.), *Hybrid Intelligence*, Computational Design and Robotic Fabrication,
https://doi.org/10.1007/978-981-19-8637-6_4

This exploration was an accidental result I discovered when learning skeletonization for image data processing. After doing research on skeletonization, I found most skeletonization methods are not “perfect”. Under some conditions, they even partially distort features of the original figure, which are considered as problems and errors to be solved in most research. In an opposite way, this research is not an optimization, nor efficient-oriented exploration, but to discover a new possibility inspired by errors. It simply asks—can we use error as a key factor to drive a system, or design? Can subtraction algorithms be used in a generative way?

2 The System

Errors of skeletonization can be easily observed and paths are not perfect, although most features of the figure are preserved. Our research started with analyzing the performance of 2D skeletonization. We implemented a voronoi-based skeletonization method [2]. It subtracts points from the gray-scale silhouette of the figure to generate a voronoi diagram, from which the voronoi edges are selected based on their intersection relationship to the silhouette. Additionally we also remove unnecessary paths by selecting segment lengths. We compared a figure with its skeletonization result (Fig. 1). Deviations can be found in these specific areas:

1. Sharp corners, even secondary corners are taken into account, which results in extra skeleton paths. Though from the result we generally consider them as unwanted.
2. "Branch” areas of a figure are not as sharp as the path we expect or the original path used for “drawing the figure”.
3. Once two figures touch each other, two figures are recognized as one connected figure and skeletonized as continuous paths—although as humans we conditionally consider it as “not connected”.
4. Due to the error of pixel distribution of the figure, extra short skeletons are created around the “tip” of a round shape.

By utilizing the difference in areas above, we can develop a recursive generative system by simply connecting the start and the end of the process, with a redraw operation. The base framework of the skeleton-based slime mold system (Fig. 2) consists of four steps for each iteration:

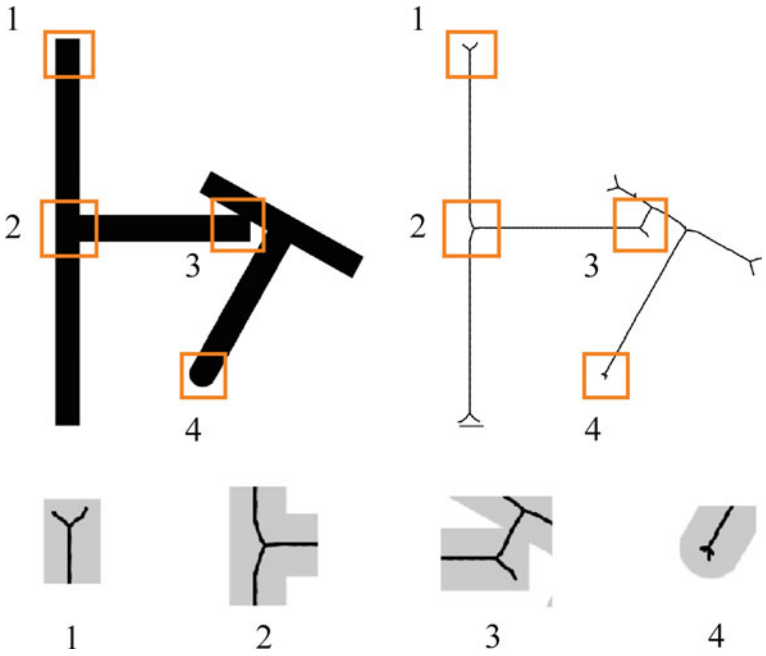

Fig. 1. Details of skeleton errors in different areas. By overlapping the two, we can observe the difference between the original figure and the best path we expect/perceive.

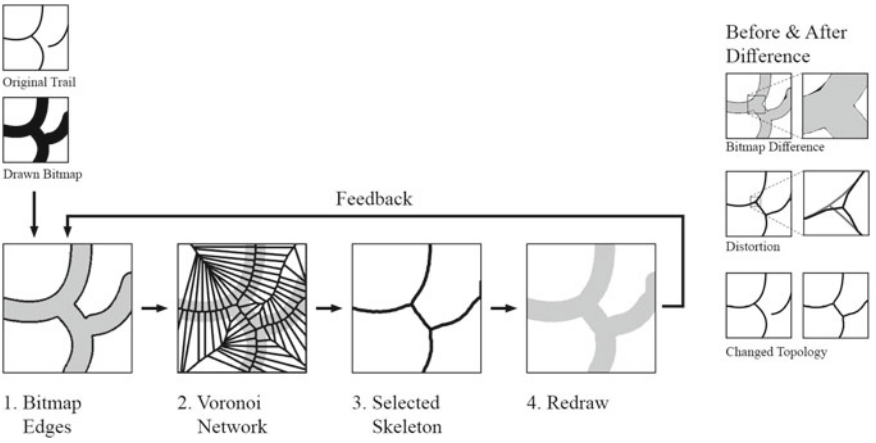

Fig. 2. Pattern generation by accumulated errors.

Input: image, length L, redraw diameter D, background color, foreground color.

Output: redrawn image
1. edges = new mid-value edges of the grayscale mesh of image
2. centers = new points by subdividing edges with length L
   voronoi units = new voronoi network based on centers
3. for each segment of each voronoi units
   if any of the following are false, delete the segment:
   a. both ends of the segment are in the foreground color area of image
   b. the length of the segment is smaller than redraw diameter D
4. fill redrawn image with background color
   draw segment in diameter D as foreground color to redrawn image
   image = redrawn image
repeat 1.

This system is developed on the Grasshopper platform [3]. C# and Rhinocommon API [4] are used for edge operations and voronoi generation. In the base framework we use a round brush to redraw the skeleton. In our experiments (100*100 size and 240*240 resolution mesh, with a 720*720 bitmap, $D = 5$, $L = 5$), maximum speed could reach 15 fps. The efficiency is largely related to the amount of branches in iterations, which directly affects the amount of redraw and skeletonization.

Based on the previous analysis of skeletonization errors, as well as details of the pattern generation, we found four behaviors that determine the pattern generation process: bending, merging, reconnecting and branching (Fig. 3).

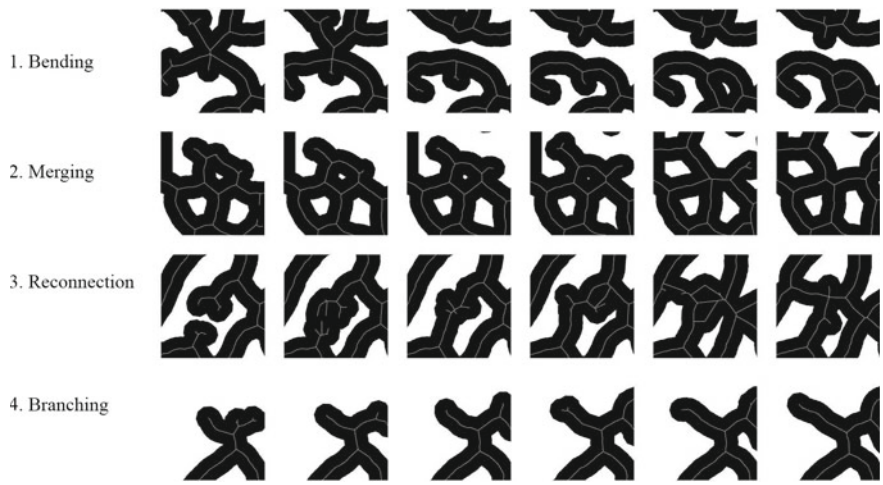

Fig. 3. Detail of pattern formation.

1. We found that skeletons of branched paths are slightly bent to approach evenly divided branch nodes. For example, if a supplemented branch node has 3 paths as 180°, 90° and 90° distribution—the distribution of this node in a skeletonized result is slightly closer to regular 60°, 60° and 60°. This is also a key factor affecting an enclosured unit to "bend" inwards, which slightly scales down the unit by iterations.
2. Merging occurs in closed units of the skeleton. Such "circles" are more likely to shrink than other parts, when angles of branches on the "circle" bend the shape inwards to the center. They vanish when the unit size is smaller than the redraw brush diameter—in which the unit is filled by redrawn paths.
3. Reconnection occurs at the tips, when ends of two paths accidentally touch each other (even just a few pixels). Such processes extend paths at tips. This behavior is triggered by errors in pixels and density of voronoi units, which affects the resolution of the skeleton. Once reconnections produce enough connection to form closed units,
4. Branching generally occurs at the convex side of paths, especially when the redrawn figure is wide. New short branches are created and extended. Larger the curvature a path has, the more likely it creates branches. The width of the figure, or redraw brush diameter also increases branching possibilities.

Four types of pattern formations are related, each of them are causes of others.

Parameters define different generative processes. When we increase the ratio of redraw diameter D to segment length L, the system shows more splitting and branching. In our analysis, when D increases, the accuracy of the network reduces, as details of branches and corners are replaced, and even merged by a thick brush. When L increases, the resolution of subtracted edges/voronoi units reduces, and more extra branches are kept in skeletonization. This feature can be seen in skeletonization, when the original figure is too wide—especially when the width of the figure is uneven, skeletons are less accurate in wide areas than narrow ones. From the pattern formation process (Fig. 4), we found that if no extra information or control is applied to the system, by iterations, the pattern will lose features of the original figure and result in a homogenized distribution. After the pattern fills the canvas, the density is balanced at an approximate ratio, despite skeletons still continuing to grow and split. After a certain amount of iterations, the pattern is stabilized and generation stops. Under such conditions, only two parameters—the redraw diameter and segment length can effectively change the pattern. Simply controlling these two parameters are limited, which doesn't meet our goals in design, only affecting the density and connectivity of the pattern, not responding to the original figure, or any additional context. In the next step of our research, we attempt to make the system more reactive to additional brush information, transform information, pixel feeds and vector feeds. Especially because the system is based on image processing, we have different options to manipulate the process by changing the drawing methods.

2.1 Changing Brush Shape

In our earliest analysis on the errors of skeletonization, small protrusions of the figure are a key factor resulting in extra unwanted paths. The sharp corners of brushes are key

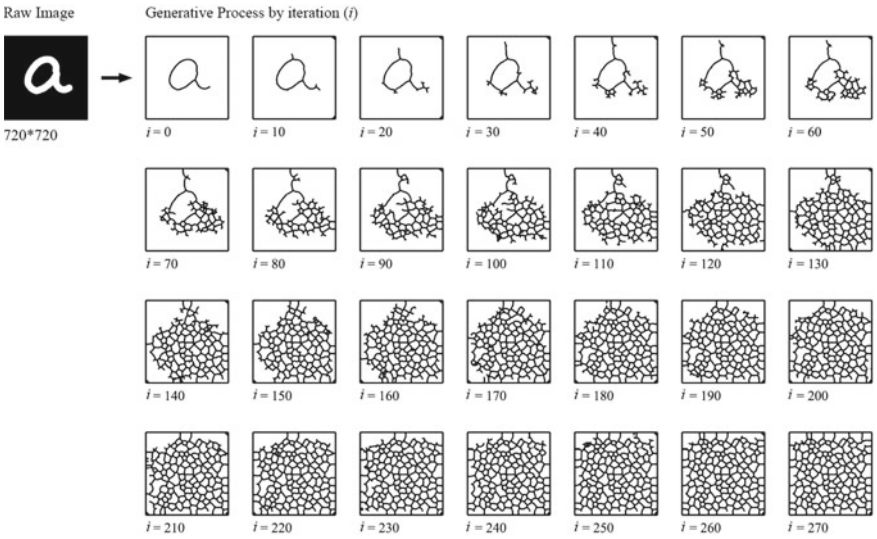

Fig. 4. Slime mold pattern with larger D/L ratio—pattern quickly expands from the original figure "a" and fills the canvas.

factors guiding branches to grow. As the brush we used in our base framework is round, in later experiments we attempted to use other brush shapes to give the redrawn figure more sharp corners. For instance, the square brush biasedly guides skeletons to grow along corners, which turns out to be more branches (Fig. 5). As we reduce the brush density and define the direction of square brush as orthodoxical, the direction of growth is more guided along four uniformed corners.

Fig. 5. Square brush redrawing the path as a saw-shaped figure, further guiding the skeleton to grow in other directions.

2.2 Additional Transforms

When we apply transforms to skeletons at each iteration, the variations of patterns are different in aspects below:

1. When scaling is applied, patterns are stretched or merged based on brush size and trail connectivity with their neighbors, affecting branching possibilities. While scaling is non uniform (e.g. scale along x axis), the pattern is more stretched in one direction than others, which results in a series of near-parallel paths.

2. When rotation is applied, patterns are cut off by the boundary of the image, since no information is given out of the image. Result of rotation usually turns out to be a circle-shaped pattern with a diameter equal to the short side of the image. Paths within the circular culling area are not affected by rotation.
3. When movement is applied, the pattern will continuously move from one side and disappear on the other side. While new paths are generated on the side it moves away from, the connectivity of the main pattern is stable. If there is any skeleton drawn on the edge of the opposite side of the vector, a stretched pattern will be continuously generated. Basically, differences made on network topology by movement are less obvious than scaling and rotating, since the relative distance between paths does not change.

Difference made by scaling information is the most effective one among others (Fig. 6). In our analysis, the density of the pattern is maintained, if redraw diameter D and segment length L are not changed. As scaling constantly changes the density of pattern/the distance between a segment of skeleton and its neighbors, paths merge or grow to maintain the "balanced" density. The further a segment is to the scaling center, the bigger absolute movement is caused by scaling, the more possible it is going to split. Vise versa, the closer it is, the more possible it is going to merge. Unlike multi-agent systems, the pattern information contained in canvas is largely limited by the resolution of the image. This feature is obvious when transform is applied.

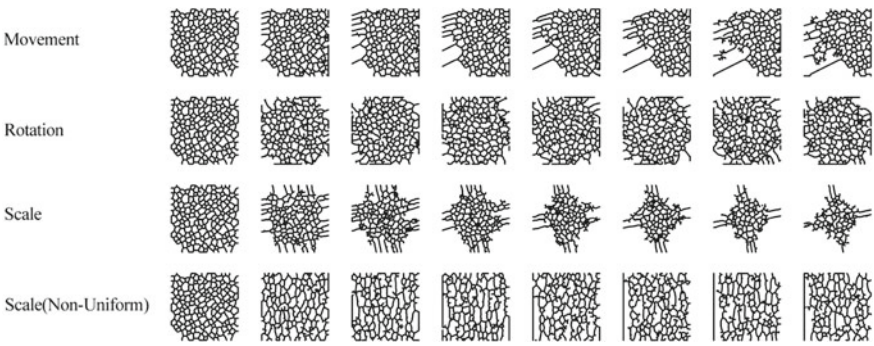

Fig. 6. Different progression of the same network, by transforming the skeleton in each iteration.

2.3 Additional Contexts

We attempted to use external bitmaps and vector trails to make this system more adaptive for design contexts. Before feeding the redrawn skeleton into the next iteration, overlaying a specific area with background or foreground information before the next iteration will largely change the way the system responds. In our experiments, we found the way

the system responds to figure-ground context is more dynamic and adaptive than directly culling areas from a general non-context pattern.

For instance (Fig. 7), under the settings of "skeletonizing white figure from a black background", continuous feeding a context figure image of black building blocks can define non-design areas which omit all generated paths. Meanwhile, moving the skeleton along a direction will turn the pattern into a shape adapted to the image context. On the "frontside"—sides facing against the movement direction, skeletons moving towards the context figure are mostly omitted, but merged along the edge of the context figure. On the "backsides"—sides facing along the movement direction, blank spaces are created like the backside of obstacles in a flushing fluid field, because no information is provided in the non-design area. Based on the ratio between D and L, the patterns on the "backsides" grow to backfill culled areas. The second variation of the experiment is, under the same skeletonization settings, redrawing the skeleton with additional vector curves (e.g. a circle or rectangle) to provide a continuous growing source from curves. In this case, whatever the pattern is, even when a huge amount of scaling is applied (e.g. scaling down around the center), paths are not strictly defined by given curves or moved away, they are still re-generated approximately around fed curves. The results of both experiments turned out to be a solution similar to road generation in an urban/landscape context.

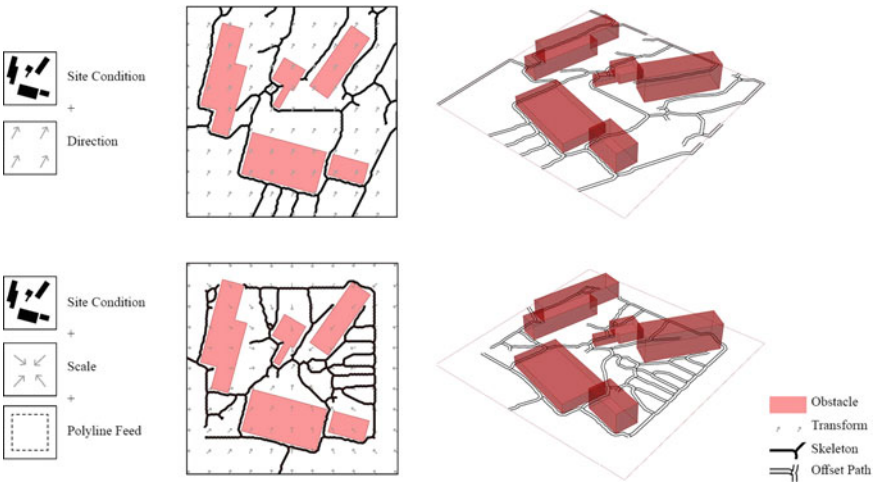

Fig. 7. Two experiments on implementing the system to a site, with obstacles and transform settings.

In our research, we also attempt to realize different pattern density by controlling redraw diameter with a gradient color map (Fig. 8). The gray scale information is remapped in range, to control redraw diameter. In this case, the patterns are more operable to meet design needs.

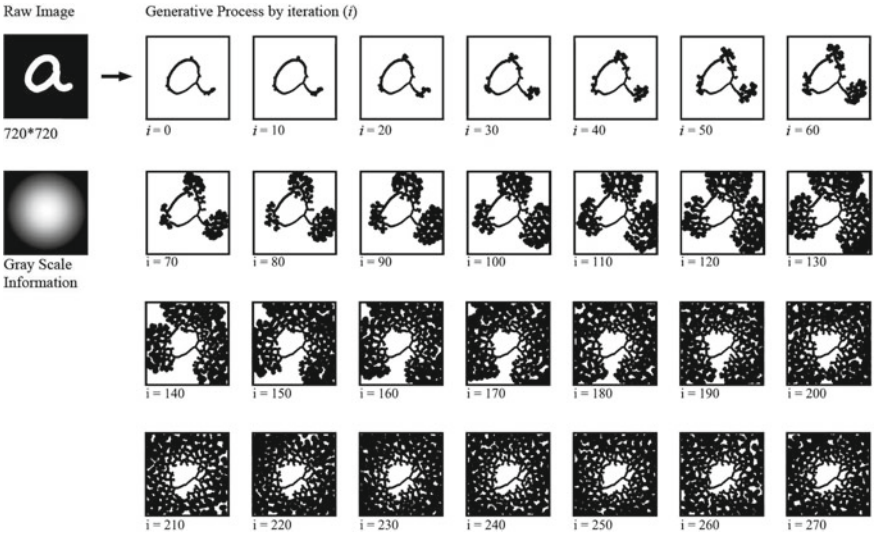

Fig. 8. Grayscale to pattern density. Networks in areas of thicker redraw diameter are more likely to grow.

3 Result Analysis

After manipulating the process, we analyze the final pattern for design purposes. Generally, to subtract paths from a traditional multi-agent slime mold system, additional operations are required to extract the medial axis from discrete point clusters. Since the patterns in our research are skeletonized, outputs are clean vector lines rather than a series of points. The paths are usable directly after removing duplicate paths—since the system uses voronoi-based skeletonization, the direct branch results are two overlapped paths in opposite directions from neighboring voronoi units. By analyzing the topology, segment length and connectivity of the network (Fig. 9), design operations can be more specific—such as generating a varied rectangle pattern along the direction of each unit.

Even though we can conduct similar analysis on a typical agent-based physarum system by voxelizing/redrawing/expanding each agent into a solid figure, and skeletonize the result, it is still not a process driven by attributes of skeleton (topology, curvature, density etc.). Influenced by the image-based method, this system shows features below:

1. Results are the medial axis, ignoring the uneven thickness/width of the foreground figure. But patterns are more likely to change, when the figure is wider, which is less controllable.

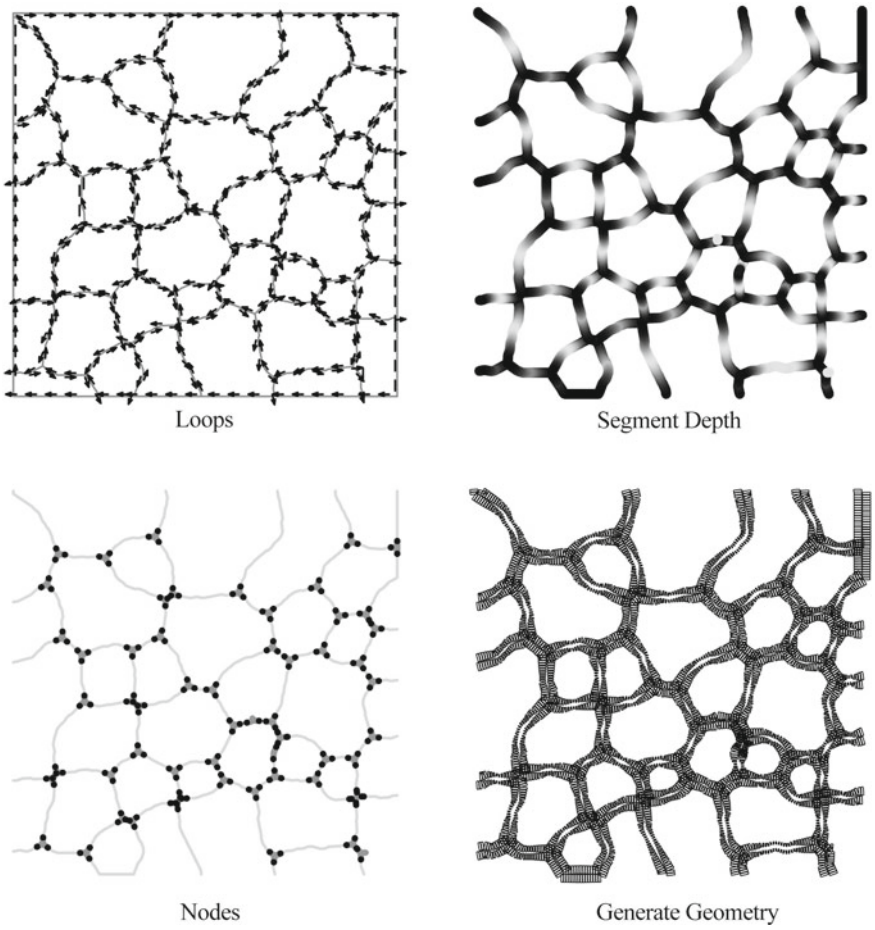

Fig. 9. Analysis of a generated network. New geometries can be generated along the path, and based on their relationship to network nodes. We can notice that three-connection nodes are most common in generated patterns (bottom left), which are also approximately averagely divided.

2. Attributes of redraw brushes (size,shapes) define the behavior of the whole system.
3. The results are connected vector trails, instead of disconnected points/agents. Directions and lengths of skeleton segments can be quickly sorted.
4. The system is reactive to context, allowing different controls over parameters.

Meanwhile, influenced by this image-based solution, we also found challenges in the aspects below:

1. Difficult to track the growth history of a network segment, like tracking an agent in a standard physarum system.

2. The accuracy of the network generation is limited, which is affected by image-related parameters (e.g. the system cannot provide precision smaller than redraw diameter, since neighboring paths are merged by the redraw process).

4 Conclusions

Our research turned out to be a similar pattern to a traditional physarum/slime mold pattern, but in a totally different process. Skeleton is defined not just as a result, but the core of the generative process.

This method is not aimed to optimize, nor as efficient as existing slime mold or skeletonization algorithms. It attempts to creatively reveal an undiscovered possibility by errors in efficiency-driven models. The result turned out to be interesting. However, there are some limitations—the practical value of this system is still unclear, since in the design field, most physarum/slime mold research are experimental or conceptual projects. It is still undeniable that this system proposed a creative way of utilizing skeletonization and a new alternate workflow to generate slime mold/physarum pattern, in a different data format.

By far we have not tested this system with pixel based skeletonization methods. It is still unclear if the generative mechanism of the system works with the same efficiency and behavior, as they run with different precision and speed. The next step of this project is examining the practical use, such as path optimization and connectivity analysis. We are also planning to test its compatibility with real-time images from devices such as webcams, to integrate the image-based process with image-based interaction—we expect the exploration to be "imprecise", as more noises and errors from reality and hardwares are accounted for. Meanwhile we will continue exploring the 3-dimensional version, which is expected to be volumetrically thickening 3D skeletonization and feeding back to the loop.

References

1. Jones J (2010) Characteristics of pattern formation and evolution in approximations of physarum transport networks. Artif Life 16(2):127–153
2. Mayya N, Rajan V (1994) Voronoi diagrams of polygons: a framework for shape representation. J Math Imag Vis - JMIV 6:638–643. https://doi.org/10.1109/CVPR.1994.323787
3. Grasshopper3d, Grasshopper. https://www.grasshopper3d.com/
4. Rhino3d, RhinoCommon API. https://developer.rhino3d.com/api/RhinoCommon/html/R_Project_RhinoCommon.htm