Continuous Line Drawings On Dendrites

Hua Li
hli1@connect.carleton.ca

David Mould
mould@scs.carleton.ca

Carleton University, Ottawa, Canada

Abstract— **Continuous line drawing (CLD) is a drawing style where a picture consists of a single long closed line without self intersection. We present an algorithm for constructing CLDs from input images. We maintain the connectivity of the line through a tree generated by path finding with consideration of the key features for a given image. A branching tree structure is first grown incrementally by selecting pixels by a cost function, relating to both the tone map and an importance map. After labeling each branch, an artificial wall is then constructed through a two-stage labeling propagation process to produce a single boundary, interpreted as the final CLD. Our method is effective and automatic, and provides some opportunities for variations.**

Index Terms— **Computer Graphics [I.3.3]: Picture/Image Generation—Line and curve generation; Computer Graphics [I.3.3]: Picture/Image Generation—Display algorithms; Computer Graphics [I.3.m]: Miscellaneous—visual arts.**

I. Introduction

Continuous line drawing can be described as follows: when drawing a picture on paper with a pen, the pen will never leave the paper until the picture is finished. It is sometimes given as a drawing challenge to young children not yet good at holding a pen; even in advanced drawing classes, artists use this technique for contouring or sketching practice. We specify this task in this paper as the problem of creating a final picture that consists of a single non-self-intersecting closed curve. In addition, we want the strokes obtained from the CLD algorithm to be meaningful. The perfect CLD should preserve the key features of an image, including both tones and edges.

Previously, algorithmic CLD art was created by solving the traveling salesman problem (TSP) [5, 10]. Given a set of locations, the salesman is required to visit each one, once and only once (Hamiltonian cycle). The graph-theoretic form is, given an undirected graph with weighted edges, find the lowest-cost Hamiltonian cycle. It is a notorious NP-Complete problem. TSP art involves placing locations over the image plane with density proportional to image darkness, and then finding the Hamiltonian cycle; to improve the appearance of the final image, object boundaries may be selected manually and no nodes placed on the exterior. However, from the view of non-photorealistic rendering [8, 19, 9], applying TSP to the CLD produces an overconstrained problem: every TSP tour gives a CLD, but not every CLD is a TSP tour. Some researchers [14, 15, 23, 24] created CLDs in the form of mazes, specifically unicursal mazes. Maze creation usually relied on interactive user assistance to have good results.

We argue that TSP-based approaches are too costly and are difficult to modify to effectively demonstrate image features, but we do not want to impose demands on the user and prefer an entirely automatic method. In this paper, we propose a new method for building a CLD in image space by path finding followed by a labeling propagation process. Unlike previous methods that concentrated on tone matching, our method respects the image structure as well. Our approach involves developing a single closed loop from a tree structure; the tree is created by path finding among a structure-aware point distribution. A depth-first traversal of the tree automatically generates a cycle, which we can interpret as a CLD. The awareness of structure is maintained throughout the entire CLD generation process. Figure 1 shows our two examples.

The paper is organized as follows. Section II. mentions existing methods for constructing the CLD. In Section III., we propose our main algorithm including tree construction, labeling, and propagation. We next give some discussion in Section IV. before concluding in Section VI..

II. Related Work

The earliest work on CLDs involved solving a the Traveling Salesman Problem (TSP) [17, 2], which is defined as finding a minimum-cost tour in a graph by which a salesman visits each node once and returns to the origin. Exact methods are feasible only for very small numbers of nodes. For large numbers of nodes, the most successful algorithms are based on tricky heuristics, such as nearest neighbors and tour improvements. In general, such approximate methods cannot guarantee optimality or correct connectivity, but the quality of a close-to-optimal tour is satisfactory for CLD applications. Some researchers [4, 10, 5] employed the existing Concorde solver to solve the TSP in order to construct different CLDs. The quality of the resulting CLDs to some degree depends on the quality of the heuristics. The Concorde solver [1] implements the Lin-Kernighan heuristic [17]. Bosch did mention that the Nearest-Neighbor heuristic may produce annoying self-crossings. Though the LK algorithm based on k-opt has an average running time of order $n^{2.2}$ and it is very difficult to get an implementation done, it has very high quality, close-to-optimum, within 2% of the Held-Harp lower bound. Though current results

(a) Original image (b) Continuous line drawing (c) Original image (d) Macaroni art

Figure 1: Continuous line drawings.

from TSP art are fascinating, if the goal is the generation of a CLD, we suggest a less computationally intensive approach.

Existing TSP-based methods concentrated on the tone of an image. The big difference between previous approaches is in the generation of starting points to represent an image, another key factor which affects the quality of a CLD. Initial results by Bosch and Herman [5] were based on halftoning [6]. Images are partitioned into grids, and the density of points for each grid is determined by the average greyscale intensity. A similar idea by Kaplan and Bosch [10] uses ordered dithering [3]. The points in each grid are not uniformly placed; rather, the placement of each point is based on a fixed patterned matrix which expresses the different intensity. The patterned matrix brings more variable visualization of a CLD, not just a connected segments or a spline curve. However, it causes artifacts from the pattern also. A more sophisticated method is based on Secord's weighted Voronoi stippling [16] to place stipples on the image. In Secord's method, an initial stippling distribution is iteratively improved by relaxation. Since Secord's method seeks to place stipples evenly in space, the outcome from stippling is more appealing than those from other placement methods.

Thanks to the maturity of the approach, TSP-based methods can produce very good results. But from another viewpoint, the well-defined problem costs us flexibility for our CLDs. Adding local control into the implementation is very difficult. The initial configuration for placing stipples almost determines the degree of the CLD quality. Image content sometimes was aided by manually segmenting objects. We argue that optimal tours may not be that important for CLD creation.

Results from Kaplan and Bosch [10] drawing the line with different types of curves are very interesting. Pedersen and Singh [14] proposed a method to build aesthetic labyrinths or mazes. The maze is parametrized by a set of piecewise curves. The curves evolve iteratively under local attraction-repulsion forces. The advantage of this method is to provide users local control over the maze structure and appearance as well as allowing multiple curves to evolve over space and time. The generation of CLDs is one possible result from the curve evolution, obtained by starting with a single closed non-self-intersecting curve. Parameters are set manually in user-defined regions. A pattern library provides users an option to choose different patterns for each region. However, the computation of the attraction-repulsion forces is very costly; also, good results depend on user intervention, while we intend to provide a fully automatic method.

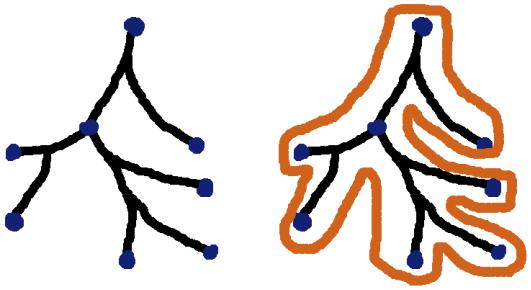
More recently Wong and Takahashi generated continuous line illustrations [21] from a given input image. But they allowed self intersections, which we want to avoid. Xing et al. proposed the concept of surface filling curves [22] and can convert a mesh surface into a single closed 3D curve that follows the mesh surface. Garigipati and Akleman [7] also proposed an extension to subdivision methods to create a 3D closed curve on a mesh. However, the resulting 3D single curve did not deal with image abstraction and thus it is a different problem from ours.

The CLD in the field of maze generation [15] is called a unicursal maze, or a labyrinth in general: a unicursal maze is one without any junctions. The generation is different from the previous evolution-based approach. Roughly speaking, instead of evolving the curves, it directly builds the surrounding walls. The interior space, or the path of the maze, will be carved into a unicursal maze. This type of maze is built on the previous mazes by transforming the dead ends or cycles into a u-turn passage after adding bisections for passages. In this paper, we adapt this idea combined with path finding to generate our CLD.

Path finding is a process to find a path by minimizing a cost function [20]. Recently, Long and Mould [11] adapted path planning for nonphotorealistic rendering, creating dendritic stylization. They crafted an initial stipple distribution and built a dendritic structure by planning paths between stipples. This will be similar to the initial dendrite generation step of our method; once we have the dendrite, we process it further in order to produce a CLD.

III. Continuous Line Drawing Algorithm

The major idea of our method is based on the duality of trees and contours. The observation is that a tree structure – a connected graph with no cycles – possesses an outer boundary that automatically forms a single closed trail, satisfying the connectivity and non-self intersection requirements for a drawing from a continuous line. See Figure 2 for an illustration. Defining the path this way provides numerous opportunities for adapting local features into the structure when finding the outer trail. We emulate Pullen's approach for creating unicursal mazes, which builds an artificial wall around the outside of a tree by transforming a corner (or a fork) into a u-turn. The channel between the man-made wall and the original tree provides a continuous line drawing.

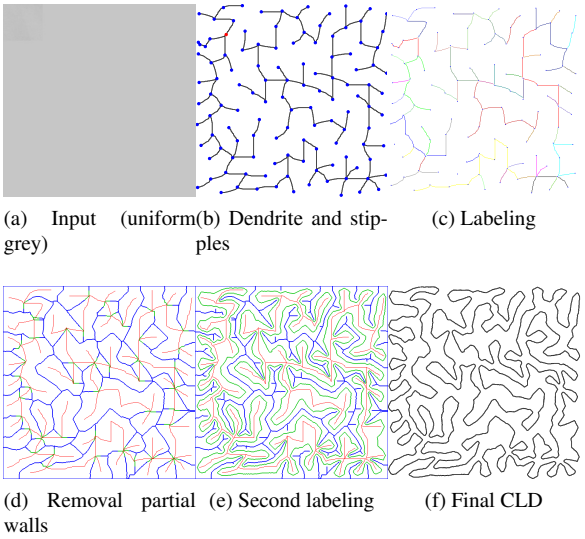


(a) (b)

Figure 2: Our key observation. (a) A tree structure; (b) the associated CLD.

A. *Overview of Our Approach*

Figure 3 shows an overview of our approach. Given an image (a), the first step constructs a structure-aware dendrites by a procedure akin to the stippling algorithm of Mould [12] followed by path finding [20], shown in (b). In the second step, after labeling each branch (as shown in (c)), a brushfire propagation of the labels produces a region map for each branch. The boundary between two different regions (shown in 3(d)) builds an artificial wall around the original tree from the first step. The algorithm then partially removes some walls which had been connected with a node with multiple branches having different labels (shown in green in 3 (d)). The consequence is a U-turn for a fork with two branches, which is similar to the algorithm used to produce the unicursal maze [15]. The third step labels the inner tree as inside wall and the outer wall with distinct identities, then propagates those two labels, and finds the boundary between the two regions, shown in (e). The resulting boundary is a CLD which follows the outer boundary of the tree, shown in (f). In the final stage, we explore three different effects: a vectorization of the CLD, a Jordon map by the Jordon Theorem [13], and a colorful macaroni art by representing the CLD with 3D cylinders.



(a) Input (uniform grey) (b) Dendrite and stipples (c) Labeling

(d) Removal partial walls (e) Second labeling (f) Final CLD

Figure 3: An overview of our approach for a CLD (121 stipples).

B. *Tree Construction*

In the NPR literature, path finding has been used in simulating natural phenomena such as dendrites [12, 11]. Path planning provides a lot of control over the shape of the paths because it allows both local and global management through the location of the endpoints, the graph weights, and the order. Long and Mould abstracted an image with dendrites with good structure preservation [11]. Our approach is similar, but instead builds the paths one by one during stipple placement [12]. Path finding is accomplished at a local scale, which reduces the computation.

Figure 4 demonstrates our placement strategy, based on Mould's stippling algorithm [12]. To build a dendritic structure, our algorithm is based on a regular eight-connected lattice and starts with a stipple A with zero cost, usually the one with high gradient magnitude, and incrementally finds the next stipple B among a set of nodes around the starting node, called a frontier, via Dijkstra's algorithm [20]. The frontier is stored in a priority queue; the nodes with lower cost will be treated first. A new stipple is found by choosing the pixel with the highest gradient magnitude along the frontier after the accumulated cost is beyond a user-defined threshold T. The cost for the new stipple is then set to zero and is again placed into the queue, ultimately updating the cost values of all the nodes around the new stipple. After obtaining two stipples, the first path is generated between them by finding a path using a cost function, which is similar to Long and Mould [11]. However, our cost function is different from their dendritic stylization, since we take the sum of edge weights rather than taking special account of the most expensive edge. As our stipples are created sequentially, the paths can be added into the tree structure step by step. The process terminates when the queue is empty.

During the stipple generation, the cost function $f_1(s,t)$ is

defined as the accumulation of weights between a source s and a node t. It is calculated as follows:

$$f_1(s,t) = \sum G(i,j), \tag{1}$$

where $G(i,j)$ is the magnitude of the gradient map [18]. In this way, the stipples preferentially follow the edges.

For path finding, another priority queue is used to find the shortest distance from a seed point to the existing tree by Dijkstra's algorithm [20]. The gradient map also guides the path towards the nodes with high magnitudes, which was similarly used by Long and Mould [11]. Since their goal was to get natural-looking dendrites, their cost map included some random variation to introduce irregularities. Here is our cost function:

$$f_2(s,t) = \sum (w_1 * G(i,j) - w_2 * D(i,j) + w_3 * K), \tag{2}$$

where $w_1 + w_2 + w_3 = 1$, K is a constant, and $D(i,j)$ is the Euclidean distance between node (i,j) and the node t to guide the path to find the tree quickly. We suggest using $w_1 > w_2 >> w_3$ to emphasize edge information.

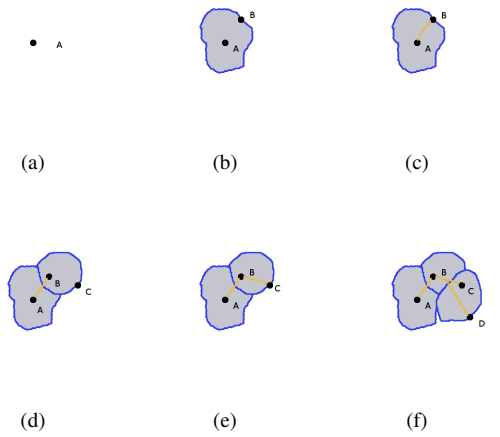


Figure 4: Progressive path finding based on a stippling generation. (a) A starting seed A; (b) expand the frontier and find a new stipple B; (c) create the first path; (d) update and expand to find the next stipple C; (e) find the second path; (f) generate the fourth stipple and the third path.

Figure 5 shows some variations. Denser dendrites from using smaller T, thus more stipples, would create a dense CLD, as shown in (a) and (c). Images in the right column were created using the same procedure as Figure 3 but at the image center, not at the left corner, and show how the starting location affects the final CLD.

Figure 6 applies some variations to the cat image. It shows a sparse version of the cat with very low stipple budget. A dense version is shown in Figure 16. The cat is visible in both results. Figure 6 also shows how the order of construction influences the final result.

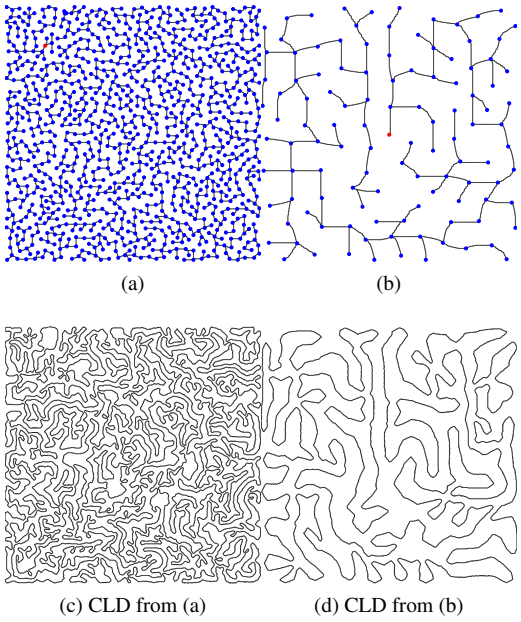


Figure 5: Variations. (a) and (c): a dense version (1045 stipples) and its CLD; (b) and (d): a different start and its CLD.

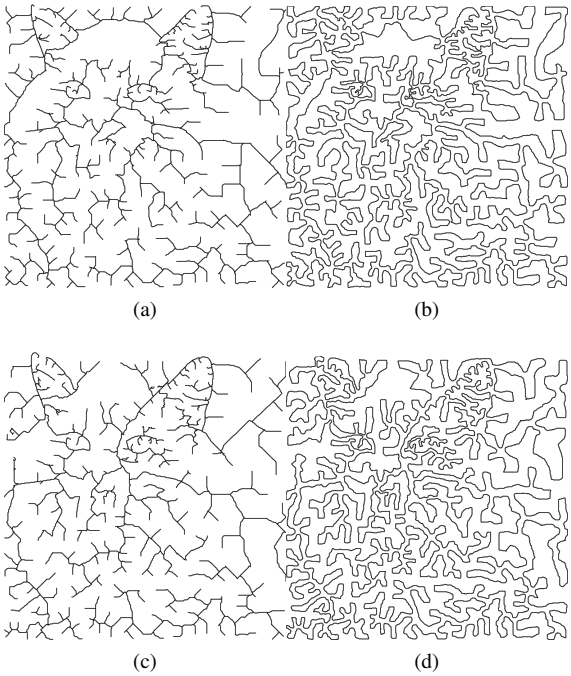
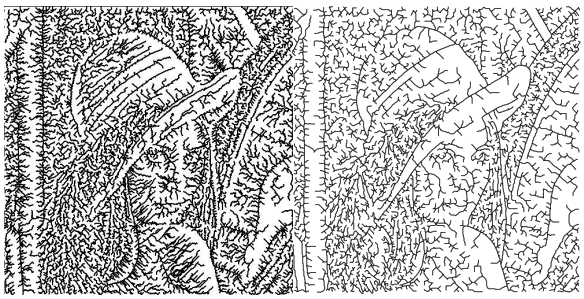


Figure 6: Cat variations. (a) Sparse dendrites (512 stipples); (b) CLD from (a); (c) different starting (541 stipples, center); (d) CLD from (c).

Figure 7 compares our dendrites with those of Long and Mould [11]. While our quality of dendrite is lower, our implementation is much faster with similar number of stipples (around 3800); also, the dendrite is not our final outcome, since we will build on it to create a CLD.

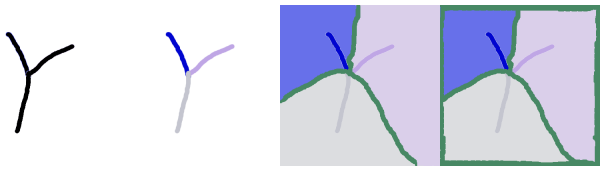


(a) (b)

Figure 7: Comparison with results from Long and Mould. (a) 3838 stipples in 530 seconds by Long and Mould [11]; (b) 3747 stipples under 20 seconds.

C. *Labeling Propagation*

After we have the dendritic structure, the next step is to define a wall around the tree by propagating labels. Our strategy is to create a wall between the two branches and to remove a portion of the wall to transform a corner into a u-turn when there is a fork in the tree structure. See Figure 8 for a visual explanation.



(a) (b) (c) (d)

Figure 8: First labeling propagation. (a) A fork; (b) labeling of branches; (c) propagation of labels; (d) creation of artificial walls.

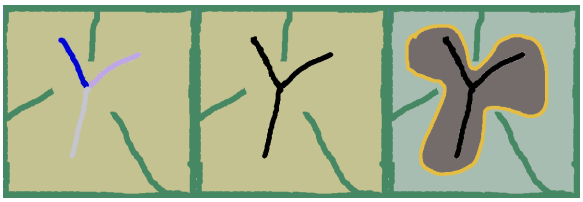
We first find the nodes N_F located at forks, such as Figure 8 (a). We label each branch with distinct identities shown in different colors in Figure 8(b). Then all nodes on the dendrites will be pushed into a priority queue with initial cost zero. The same graph and weights used in finding the stippling will be applied here again, and brushfire propagation used to extend the influence of each branch gradually into the unlabeled nodes, similar to finding a distance map. Since each branch in the fork is assigned a different identity, the method creates a wall (in green) between two branches, as illustrated in Figure 8 (c). To maintain the connectivity of the boundary of an image, we add a boundary box as walls and in stipple placement stipples are not placed in areas near to the boundary box. See the green walls in Figure 8 (d).

Then, the u-turn is constructed by removing the portions of the walls within a distance d of the nodes N_F; The parameter d is calculated based on the average intensity $\bar{I}$ around the node N_F:

$$d = L + M * \bar{I}(i, j)/255, \tag{3}$$

where L and M are constants controlling the minimum and maximum distance.

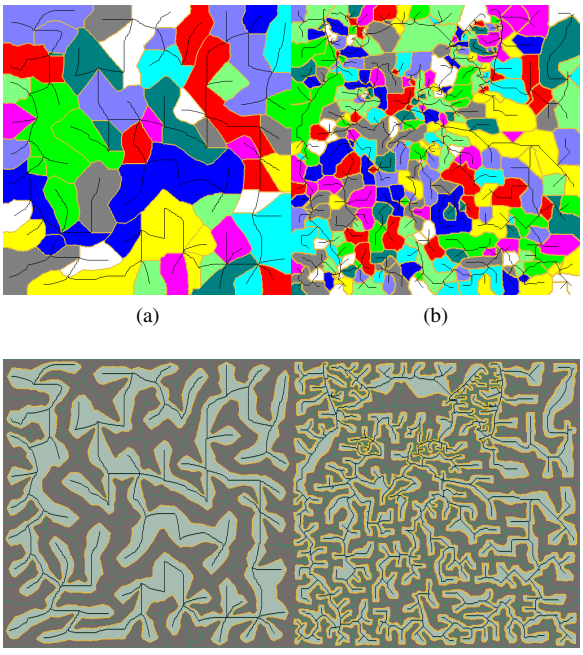
The last step, shown in Figure 9, applies the same labeling strategy to both tree and wall, obtaining a boundary midway between them. Figure 9 shows the label propagation process: the original tree receives one label, the remaining walls receive another, and the label propagation produces a wall between them. This wall is the final continuous-line drawing.



(a) (b) (c)

Figure 9: Second labeling propagation. (a) Partial removal; (b) labeling both tree and the walls; (c) CLD creation.

Figure 10 shows another illustration of the process. In the top row, we see two intermediate maps (those used to generate Figures 3 and 6 (b)) after the first labeling, while the bottom row shows the final labeling. Figure 11 displays



(a) (b)

(c) (d)

Figure 10: Two labelings. Top: first labeling for Figure 3 and Figure 6 (b); bottom: second labeling for Figure 3 and Figure 6 (b).

after finding a CLD result a propagation of labelled branches from dendrite tree affects the neighbouring pixels for the cat. Each color indicates an affected local area.

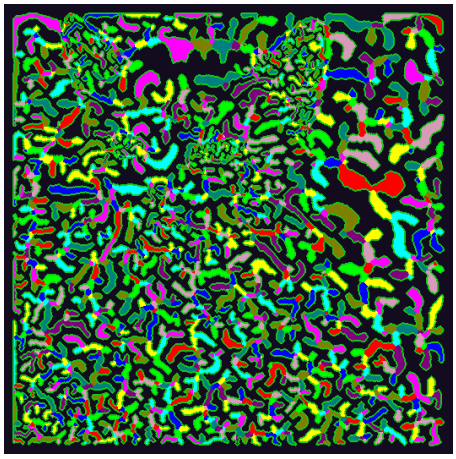


Figure 11: An influence map for the cat.

D. *Post-processing Effects*

After we have the labeling maps, we create three different effects based on it: vectorization, Jordon map, and Macaroni art.

D.1 Vector Graphics

The previous propagation generates a map with two regions separated by the continuous line. We can vectorize the curve by using standard boundary tracing with eight-connectivity [18]. Following this, we can display a vector version of the continuous line, with better quality in print than the raster version.

D.2 Jordan Map

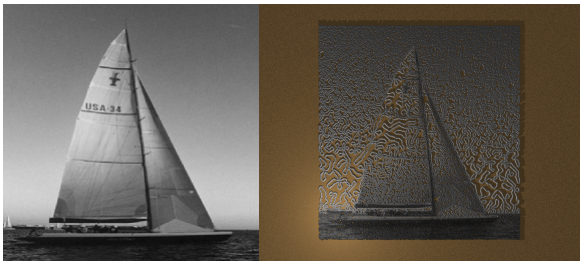
The CLD will be a closed simple curve. As the Jordan Curve Theorem [13] states, any simple closed curve in the plane separates the plane into two regions: one part lies inside the curve and the other part lies outside it. After finding the CLD, we can draw the inside part and outside part with different colors, similar to the duotone surfaces of Garigipati and Akleman [7]. Figure 16 demonstrates the effect where we draw the inner region and the outer region with different colors.

D.3 Macaroni art

We also suggest an effect called Macaroni art, a children's art style produced by pasting different objects (especially dry pasta) onto a canvas to give an impression of depth to an image. We replace the continuous lines with a collection of 3D cylinders and render it using the raytracer POV-ray. Figure 12 and Figure 1 (d) show this effect.

IV. Discussion

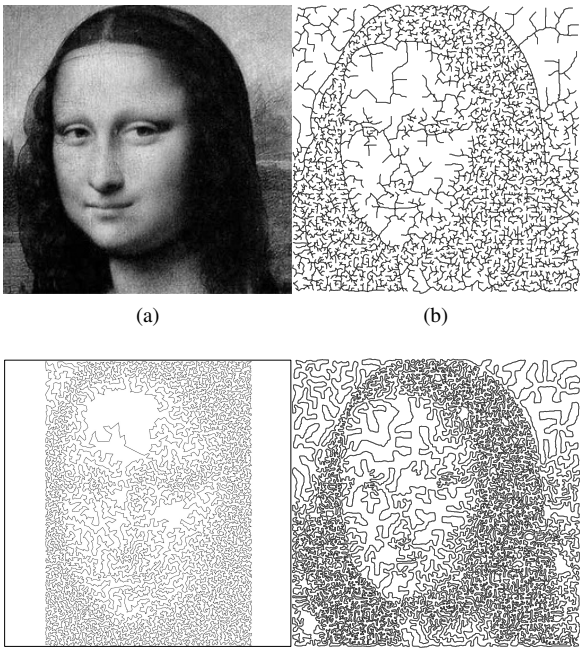
We propose a new method to create a CLD, using progressive methods in both stippling generation and the dendritic structure. Path finding provides a lot of potential flexibility of control over tone, structure, and pattern. We are interested



(a) (b)

Figure 12: Macaroni art.

in algorithmic methods to generate continuous line drawings, so we compare our method with automatic TSP-based methods [5, 10]. Figure 13 compares TSP art with our result. Figure 13 (a) shows the original, (b) shows the dendrites used to create our final CLD, Figure 13 (c) shows the outcome of the TSP method by Kaplan and Bosch [10], and (d) shows our result. Our approach preserves the face profile more clearly than does the TSP method: especially notice the edge profile between Mona Lisas face and her hair on the left side. Since we maintain the connectivity of the CLD based on a tree, we have a simpler problem than creating a full CLD directly.

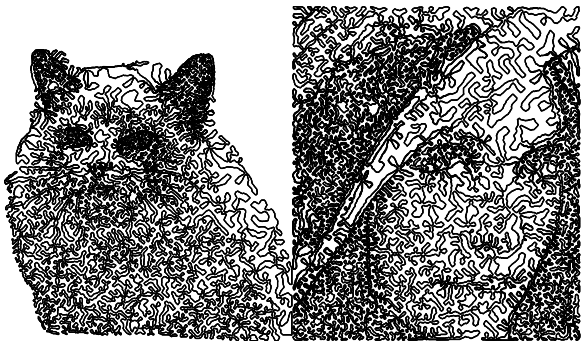


(a) (b)

(c) (d)

Figure 13: (a) Original image; (b) Dendrites; (c) TSP-Art from Kaplan and Bosch [10]; (d) CLDs.

Relevance to the content of an image is a consideration throughout the stippling generation, path finding, and labeling. Thus, our result with fewer stipples could still indicate the content to some extent. In Figure 6, we can still identify the cat. To further enhance the image content, we sug-



(a) (b)

Figure 14: Thickened edges

gest thickening some of the segments of the CLD when they are lying on the edges to indicate the importance. Figure 14 shows clearer structures by indicating the edges with thickening. The cat in this figure is more prominent because the user-defined region for showing the silhouette as a mask improves the result. Figure 15 shows the mask used for creating the cat, also used for Figure 1. Basically, our method is a space-filling curve in image space. This example shows that interactivity may also be incorporated by selecting object regions first and then filling them with the CLD.

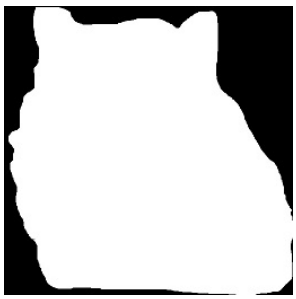


Figure 15: The mask used in the cat example.

More examples are shown in Figure 16 and Figure 17. Both demonstrate a lot of image information such as object silhouettes and large-scale structure. The CLDs for the man's face and the background of the sailboat indicate the tone change while the outline of the man and the sailboat look clear as well.

There are two major issues for our method. First, because our method is image-based, the quality of our method severely depends on the resolution. Figure 13 shows some holes in our CLD because the operation of removal of partial walls might change the topology of the tree and may create some small new trees. One solution is to temporarily increase the resolution when labeling and removing. Another solution to it is to hook the isolated holes later to make it filled.

Another issue is the inconsistency between the CLD and the edge. Our strategy is to follow the edges as closely as we can, yet we do not exactly follow the edges because we also prefer an even spaced propagation.However, we provide users an effective and flexible alternative to TSP-based approaches to create CLDs.

V. Future Work

We next discuss two possible future modifications. One takes advantage of the tree structure and obtains a CLD in a very fast way by two dilation operations. Subjectively, we like this effect for the CLD although it uses double lines. The other approach seeks to improve the structure preservation by combining a Canny result into the tree structure.

A. *Morphological Dilation of Dendrite*

Our dendrites are trees. The trail of the depth-first traversal of a tree is a continuous line. One TSP solution is first to construct the minimum spanning tree and double each edge. A TSP tour is present in this doubled tree. Similarly, if we just want to get the visualization of a CLD, we can apply image processing: once we have the dendrite, two dilations (A and B) with different lengths for a tree can produce a CLD as follows. For an output image C, find $C = A(l_1, c) + B(l_2, c_{bg})$, where l_1 and l_2 are the radii of the dilation circles, with $l_1 > l_2$. The color c of the dilation circle in A is different from the background color c_{bg}. Conversely, the dilation circle in B uses the same color as the background. A preliminary result is shown in Figure 18. One drawback is that the CLD has double lines. Also, we are forced to confront the resolution issue, resolved here by copying the dendrites into a larger image before processing.

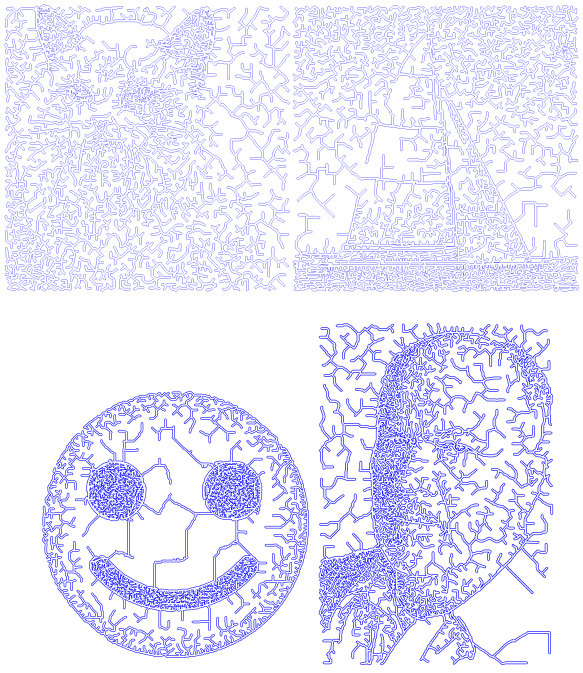
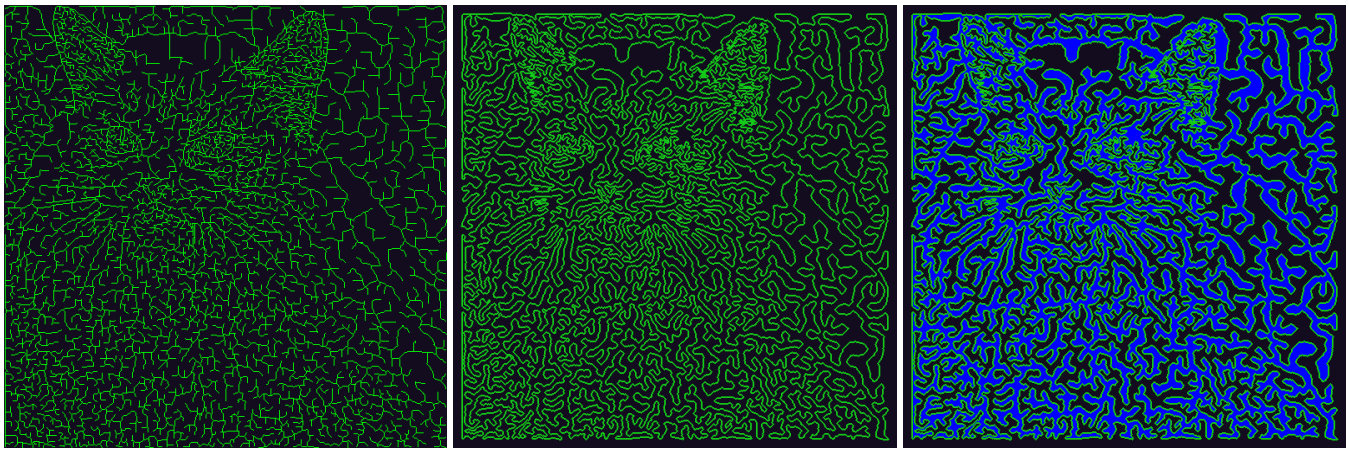
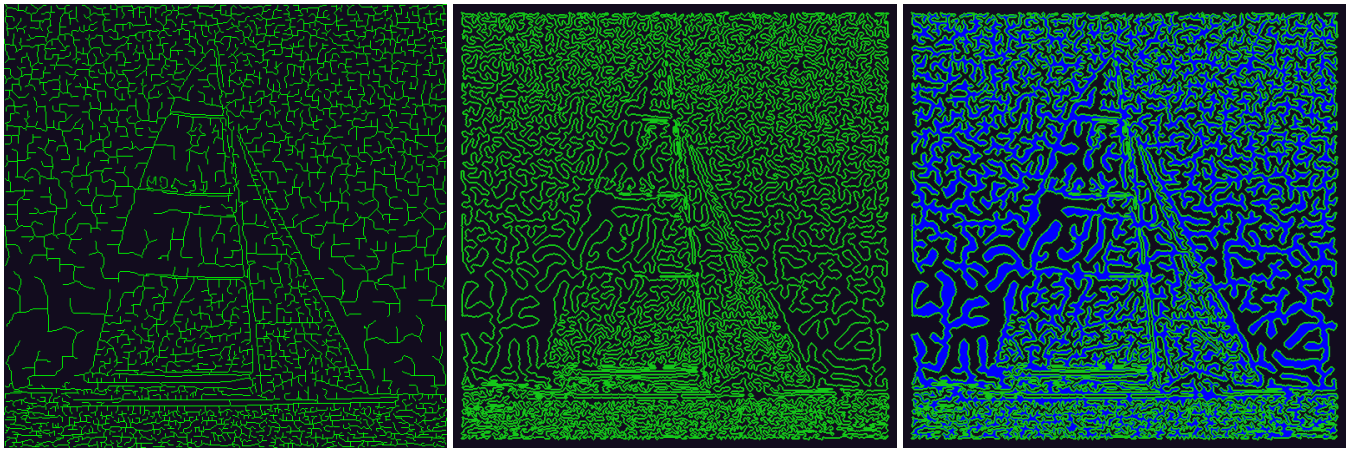


Figure 18: CLD from image morphology.



(a) Dendrite (b) CLD (c) Jordon map



(d) Dendrite (e) CLD (f) Jordon map

Figure 16: More results.

Figure 17: Another CLD example from a dendrite. Left: original image; middle: dendrites; right: CLDs.

B. Incorporating Canny Edges

We notice the branches on the edges affect the quality of the final CLD: they distract our eyes thus reducing the clarity of the edge information. Here we suggest incorporating the Canny edges into our tree; we should try to connect fewer branches to the Canny edges in order to minimize the distraction. The plan is as follows. First, we should simplify Canny edges by removing any closed edges and separating the edges into individual paths without branches. Then, building the tree as before with the Canny edges might improve our structure preservation a lot. We think it will be a very promising future direction.

VI. Conclusion

In this paper, we proposed a new idea about automatically drawing a picture with a continuous line. The process involves first creating a structure-aware dendrite and then expanding it into a region whose boundary is the continuous line. Our method can indicate the edge information of an image as well as providing some tone suggestion. We demonstrate some possible variations based on different stages and show three post-processing effects. Because we do not require the computation of a TSP tour, we should have better performance.

However, automatic continuous line drawing remains a very hard problem. In this paper, we aimed to have a better abstraction. But sometimes the good abstraction is damaged by the necessity to maintain connectivity. Automatically creating a connected line with high quality of abstraction and good aesthetics remains an open problem. In the end, we might have to rely on some user intervention to obtain good results.

REFERENCES

[1] D. Applegate, W. J. Cook, S. Dash, and A. Rohe. Solution of a min-max vehicle routing problem. *INFORMS Journal on Computing*, pages 132–143, 2002.

[2] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[3] B. Bayer. An optimum method for two-level rendition of continuous-tone pictures. *In IEEE International Conference on Communications, IEEE*, pages 26:11–26:15, 1973.

[4] R. Bosch. Opt art. In *Proceedings of the Third international conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, CPAIOR'06, 2006.

[5] R. Bosch and A. Herman. Continuous line drawings via the traveling salesman problem. *Operations research letters*, 32(4):302–303, 2004.

[6] R. W. Floyd and L. Steinberg. An Adaptive Algorithm for Spatial Greyscale. *Proceedings of the Society for Information Display*, 17(2):75–77, 1976.

[7] P. Garigipati and E. Akleman. Duotone surfaces. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, CAe '12, pages 99–106, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[8] B. Gooch and A. Gooch. *Non-Photorealistic Rendering*. A K Peters/CRC Press, 2001.

[9] A. Hertzmann. Non-photorealistic rendering and the science of art. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '10, pages 147–157, New York, NY, USA, 2010. ACM.

[10] C. S. Kaplan and R. Bosch. Tsp art. *Renaissance Banff: Bridges 2005: Mathematical Connections in Art, Music and Science*, pages 301–308, 2005.

[11] J. Long and D. Mould. Dendritic stylization. *Vis. Comput.*, 25:241–253, February 2009.

[12] D. Mould. Stipple placement using distance in a weighted graph. In *Computational Aesthetics*, pages 45–52, 2007.

[13] J. O'Rourke. *Computational Geometry in C (Cambridge Tracts in Theoretical Computer Science)*. Cambridge University Press; 2 edition, 1998.

[14] H. Pedersen and K. Singh. Organic labyrinths and mazes. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 79–86, New York, NY, USA, 2006. ACM Press.

[15] W. D. Pullen. Think labyrinth. Website, 2010. `http://www.astrolog.org/labyrnth.htm`.

[16] A. Secord. Weighted voronoi stippling. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 37–43, New York, NY, USA, 2002. ACM Press.

[17] S. S. Skiena. *The Algorithm Design Manual*. Springer-Verlag, New York, 1997.

[18] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing: Analysis and Machine Vision*. Thomson Learning; 2nd Revised edition edition, 1998.

[19] T. Strothotte and S. Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann; 1st edition, 2002.

[20] P. Winston. *Artificial intelligence*. Addison Wesley; 3 edition, 1992.

[21] F. J. Wong and S. Takahashi. A graph-based approach to continuous line illustrations with variable levels of detail. *Computer Graphics Forum*, 30(7):1931–1939, 2011.

[22] Q. Xing, E. Akleman, G. Taubin, and J. Chen. Surface covering curves. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, CAe '12, pages 107–114, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[23] J. Xu and C. S. Kaplan. Image-guided maze construction. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[24] J. Xu and C. S. Kaplan. Vortex maze construction. *Journal of Mathematics and the Arts*, 1(1):7–20, 2007.