The Aesthetics of Rapidly-Exploring Random Trees

Michael Burch* and Daniel Weiskopf†
VISUS, University of Stuttgart

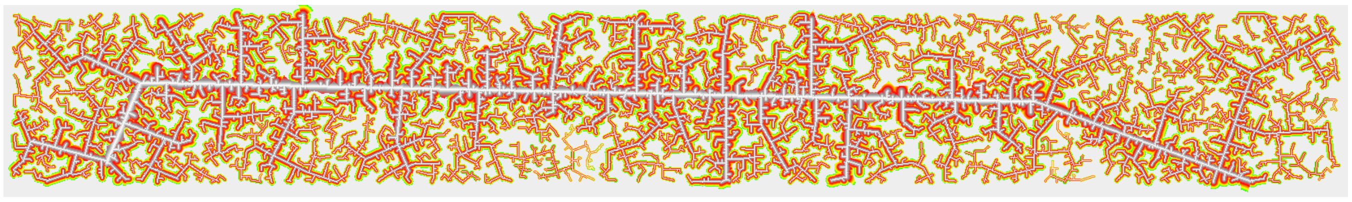


Figure 1: *A Rapidly-Exploring Random Tree after 5,000 algorithm iterations. The tree diagram is visually enhanced by encoding the tree depth by link color and line thickness.*

Abstract

Rapidly-Exploring Random Trees (RRTs) have been introduced as an algorithmic concept for the rapid exploration of configuration spaces targeting fast path planning, mainly applied in the field of robotics. Typically, such structured space organizations are only used on an algorithmic level but not for direct visual representation. In this paper, we illustrate the aesthetics of such RRTs by displaying them in a visual form that serves as a basis to generate algorithmic art. Apart from the visual encoding of such space-filling node-link diagrams, we demonstrate how these trees grow in the configuration space for RRT layouts with and without incremental distances from the initial point. Additionally, RRTs can be visually enhanced by several inherent tree metrics such as tree depth, subtree size, and branching factors to make the diagrams more aesthetically appealing and readable. We provide examples of different tree sizes and illustrate the effect of changes to several control parameters such as color coding, line segment thickness, layouts, and shape constraints.

CR Categories: I.3.0 [Computer Graphics]: General;

Keywords: Algorithmic art, Rapidly-Exploring Random Tree, hierarchy visualization, node-link diagram

1 Introduction

Rapidly-Exploring Random Trees (RRTs) have been introduced as a general concept to rapidly explore a given configuration space in order to solve path planning problems efficiently [LaValle and Kuffner 1999; LaValle and Kuffner 2001]. The RRT originally does not stand for any kind of visual representation although the term “tree” indicates that it might be a visual metaphor for a hierarchy. In fact, the RRT is traditionally used in the form of a data structure and algorithm focusing on efficient searching in non-convex high-dimensional search spaces. Such a tree is computed incrementally by adding a new sample to the tree randomly, computing the least distant already existent sample in the tree by a distance function, and finally connecting both samples by a straight line that produces a new branch in the tree. By this strategy, it is guaranteed that the corresponding tree diagrams are free of link crossings, i.e., the aesthetic criterion of planarity is preserved. Furthermore, such tree diagrams become space-filling variants of node-link diagrams. Furthermore, the RRT can be classified as a Monte-Carlo-like method of biasing search into largest Voronoi regions. The class of those tree algorithms produces different types of stochastic fractals [Mandelbrot 1982]. The fractal behavior is beneficial because many natural phenomena show fractal characteristics.

The planarity, space-filling, and fractal properties of the RRT are useful characteristics for producing aesthetic diagrams. Therefore, we have chosen RRTs as a basis for algorithmic art. We focus on improving the visual representation of the RRTs to increase their aesthetics and readability. Therefore, we do not just use traditional line drawings of node-link tree diagrams but additionally take into account tree-inherent metrics such as tree depth, subtree size, and branching factor. Those metrics are visualized by

- color coding and
- line thickness.

We focus on the tree depth because it illustrates the length scale of the depicted part of the tree. Descending the hierarchy, we visit shorter links shown by thinner lines. Therefore, the length scale of the fractal corresponds naturally to the line thickness. The visual representation is further improved by additional color coding, which makes the diagrams easier to perceive and structures easier to be traced, especially for densely packed RRT regions.

Furthermore, we provide several additional parameters that allow us to modify and control the appearance of the RRT diagrams:

- the starting point at which the algorithm is initiated,
- the number of incrementally added vertices (iterations),
- the incremental maximal distance from the initial point,
- obstacles or space constraints,
- and the shape that drives the construction of the RRT.

We illustrate the effects of those parameters on the visual appearance by including several example images in Section 4.

*e-mail:michael.burch@visus.uni-stuttgart.de
†e-mail:daniel.weiskopf@visus.uni-stuttgart.de

2 Related Work

RRTs have been introduced as a search tree generation algorithm by LaValle and Kuffner [1999; 2001]. They guarantee a complete exploration of a pre-defined configuration space. Optimal control theory [Bellman 1957], non-holonomic planning [Laumond et al. 1998], and randomized path planning [Amato and Wu 1996; Kuffner and LaValle 2000] served as basis ideas for the development of RRTs.

The concept of RRTs is related to Diffusion-Limited Aggregation (DLA), which is described as the process in which particles undergoing random walks cluster together to form aggregates of themselves due to Brownian motion [Witten and Sander 1981]. The formed clusters are denoted as Brownian trees and can be classified as a fractal having a fractal dimension of 1.71 in 2D space, which is used as a complexity measure in fractal theory [Peitgen and Richter 1986; Peitgen and Saupe 1988; Peitgen et al. 1992]. In our work, we also investigate a variant of such Brownian trees—although not explicitly described—where the tree is growing radially outward starting at a single initial point; see Figure 2.

RRTs have some important benefits. They are well-known for their fast expansion as tree structures and hence, they cover yet unexplored regions of the pre-defined configuration space. This tree generation and fast evolving hierarchical structure building process can be efficiently used for searching possible targets, hence their particular suitability in path planning. Some algorithm variants even take obstacles or space-constraints into account to safely guide a robot from a source to a target [Rodriguez et al. 2006]. In our work, we also experiment with such space constraints. The structure of RRTs has recently also been applied to guide the edge bundling in node-link diagrams of graphs [Burch et al. 2013b]. In the domain of game design, RRTs can be applied for constructing a compact graph representation to automatically explore existing levels of a 2D platform game [Bauer and Popovic 2012].

Although the powerful concept of the RRTs has successfully been applied to path planning problems in high-dimensional spaces [Nieto et al. 2010], research on aesthetically visualizing the data structure enhanced by tree-specific metrics such as tree depth, branching factor, and subtree size is rare. From a visualization perspective, these diagrams have an important advantage compared to existing node-link tree visualizations: they are space-filling representations of hierarchical data, which is discussed for various visual metaphors for hierarchies by McGuffin and Robert [2009]. This space-efficient representation makes the RRTs hard to understand since they do not provide a clearly structured tree diagram.

For this reason, visually encoded extra information can be used to understand and explore such hierarchical structures in a better way and it allows the viewer to visually compare similar and non-similar patterns in the data. The perceptual abilities of the human visual system can be better exploited due to its strengths in fast pattern recognition when the representation is emphasized by tree-specific features.

In the field of information visualization, hierarchical data has been visualized since ancient times. Five major visual metaphors for this type of data have been developed over the years that are used depending on the application domain, including the prominent node-link diagrams [Reingold and Tilford 1981]. The RRT diagrams also belong to the class of node-link diagrams although the vertices are not explicitly shown. Further metaphors are indented plots [Burch et al. 2010] that follow the principle of indentation, layered icicles [Kruskal and Landwehr 1983] exploiting the strategy of stacking, treemaps [Shneiderman 1992] that apply nesting layout strategies, and fractal approaches [Mandelbrot 1982; Barnsley et al. 1988; Rosindell and Harmon 2012], to which RRTs also may belong due to their self-similarity property.

For the prominent example of node-link diagrams, layout and aesthetic criteria are discussed in the literature [Wetherell and Shannon 1979]. Although node-link diagrams are intuitive and easy to draw, visual scalability and labeling often are an issue. For RRTs, visual scalability is a highly relevant issue because those trees are procedurally constructed and can therefore become large with little effort on the data-production side. A benefit of the RRTs is their embedding in a crossing-free manner into the configuration space. We further increase the visual scalability by not drawing the nodes explicitly but only implicitly at the connection points and line ends. Another strength of RRTs is their space-filling character that allows for good visual scalability. Finally, we enhance the visual representation of the RRT by color coding and varying line thickness for better readability and more aesthetic display.

In this sense, our goal is to generate algorithmic art. Therefore, our work is related to previous work on algorithmic art, such as Traveling Salesman Problem art [Bosch and Herman 2004; Kaplan and Bosch 2005]. That approach also produces aesthetically appealing diagrams but not for hierarchical structures as in our work.

3 Rapidly-Exploring Random Trees

In this section, we describe how RRTs are generated and used for rendering. We start with the actual RRT generation, partially following our previous discussion from [Burch et al. 2013b]. Then, visual enhancements for an aesthetic visual representation are presented. Finally, we discuss our pixel-based rendering algorithm for the visually enhanced RRTs.

3.1 RRT Generation

Algorithm 1 provides the pseudo code for RRT generation. A general configuration space $\mathbb{S}$ serves as the basis for building the tree that starts with an initial vertex: the root vertex v_{init}. This vertex can be placed randomly, in the display center, or manually at any location demanded by the user. The algorithm works incrementally by adding N new vertices to an existing RRT that initially contains only the root vertex v_{init} with its corresponding coordinates.

The incremental steps of the algorithm work as follows: A random node v_{rand} is chosen and the nearest already existing node v_{near} in the RRT or point on an already existing line is computed. An incremental distance Δv can be used to move from v_{near} in the direction of the randomly chosen node v_{rand}. This incremental distance can be used to avoid newly added links that are longer than a given

Algorithm 1 Rapidly-Exploring Random Tree

```
Generate_RRT($v_{\text{init}}$, $N$, $\Delta v$):
  // $v_{\text{init}}$ : Initial root position
  // $N$     : Number of nodes to be added
  // $\Delta$     : Incremental distance from $v_{\text{near}}$ to $v_{\text{new}}$
  $RRT.init(v_{\text{init}})$;
  for $i := 1$ to $N$ do
    $v_{\text{rand}} \longleftarrow Random\_Node()$;
    $v_{\text{near}} \longleftarrow Nearest\_Node(v_{\text{rand}}, RRT)$;
    $v_{\text{new}} \longleftarrow New\_Node(v_{\text{near}}, \Delta v)$;
    $RRT.add\_node(v_{\text{new}})$;
    $RRT.add\_link(v_{\text{near}}, v_{\text{new}})$;
  end for
  return RRT;
```

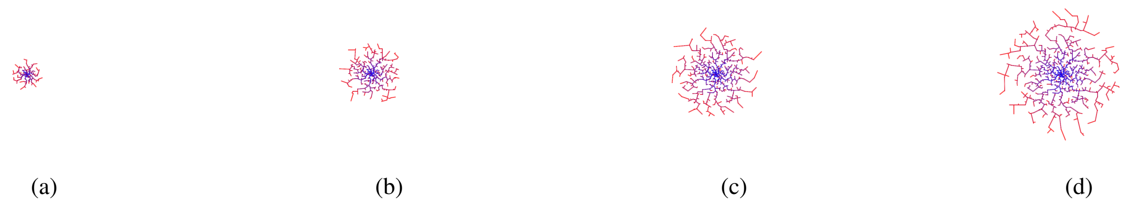

(a) (b) (c) (d)

Figure 2: *A growing RRT with an incremental distance from the initial vertex for the newly added vertices: (a) after 250 iterations, (b) after 500 iterations, (c) after 750 iterations, (d) after 1,000 iterations. The initial vertex is located in the center, which results in a radially outward growing tree. Red color indicates edges and vertices added in most recent iterations, whereas blue color indicates earlier ones.*

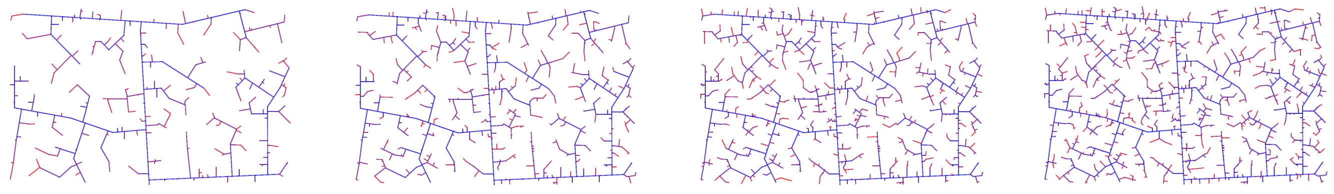

(a) (b) (c) (d)

Figure 3: *The complete configuration space can be incrementally filled by randomly chosen vertices: (a) after 250 iterations, (b) after 500 iterations, (c) after 750 iterations, (d) after 1,000 iterations. There is no incremental distance for added vertices, i.e., there are no constraints for newly added vertices. Red color indicates edges and vertices added in the most recent iterations.*

threshold, e.g., to prevent nodes to be located further away from the initial vertex position than the threshold distance. As the last step, the new node v_{new} and new edge $(v_{\text{near}}, v_{\text{new}})$ are added to the RRT.

Figure 2 illustrates how an RRT is incrementally built. Here, 1,000 iterations are used to produce an RRT whose initial vertex position is located at the center of the display. An incremental distance is used when the tree evolves, i.e., new vertices can only be added if those are below a given threshold distance from the initial vertex. This threshold is enlarged over time, leading to circular shaped RRT layouts. Figure 2 shows four snapshots after $N = 250$, 500, 750, and $1,000$ iterations. A blue-to-red color gradient is used to visualize the time at which vertices and edges were added: red colored vertices and edges were added later and blue ones earlier—just for illustrative purposes.

Figure 3 shows snapshots of RRTs after $N = 250$, 500, 750, and $1,000$ iterations. Here, the initial vertex position is selected randomly and no incremental distance is used, i.e., vertices are placed anywhere and the already existing vertex with a minimal distance to the added one has to be computed. Then, a new edge is introduced connecting both vertices. The tree generation process generates a layout that is free of link crossings and that benefits from its space-filling property in contrast to other existing node-link tree diagrams. Again, the time when edges and vertices were added is visualized by a blue-to-red color gradient.

Figures 2 and 3 illustrate how the RRT layout can be modified and controlled by varying the position of the initial vertex and the distance theshold for the newly added vertices.

3.2 Visually Enhanced RRTs

Large RRTs that show the tree structure only by equally thick lines and uniform color make it hard for the viewer to identify global and local tree structures. Those drawbacks of the traditional line representation without the enhancement of tree-specific metrics come from its space-efficient representation. The viewer is not able to explore the tree because it is not separated into distinguishable layers as it is, for example, done in traditional node-link tree diagrams following a top-down layout approach; see the evaluation of tree-reading strategies in an eye tracking experiment [Burch et al. 2011; Burch et al. 2013a]. However, even if the separation into single tree layers is not that clear in an RRT, we can achieve readable and aesthetic diagrams. The RRT representation can be extended and visually enhanced by integrating visual features to each of the used line segments; see Figure 4.

Each hierarchy or subhierarchy contains some inherent specific properties that can be visually encoded to make a tree structure clearer:

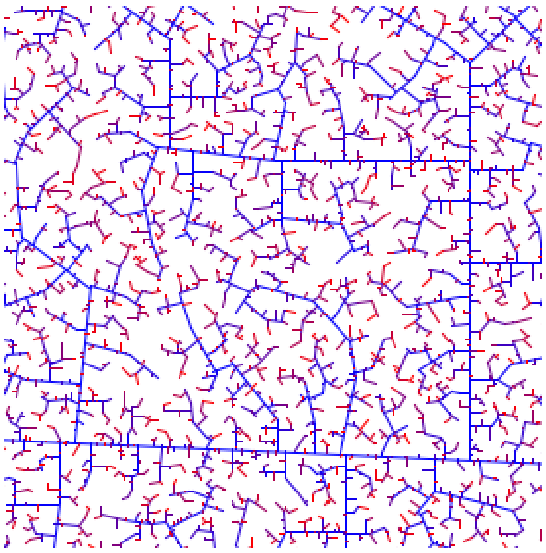
- **Tree branching:** The branching factor at a specific vertex is defined as the number of splittings into subhierarchies, i.e., the number of direct child vertices. In fractal theory, this branching can be used for computing the fractal dimension as a complexity measure.

- **Tree depth:** The depth of a tree is defined as the longest path from the root to a leaf vertex.

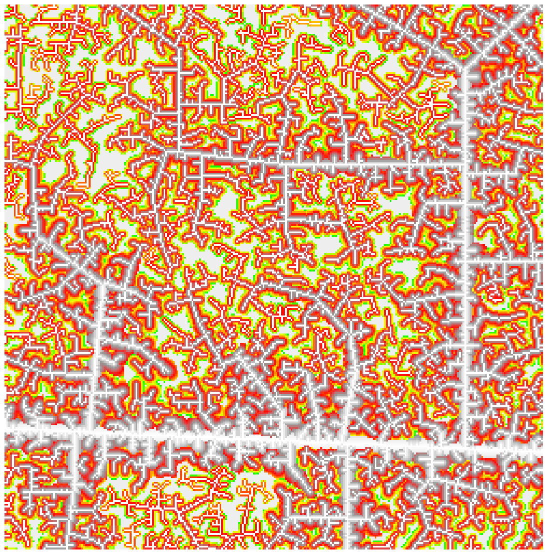
- **Tree size:** The size of a tree is defined as the number of vertices contained in it: root vertex, inner vertices, and leaf vertices.

Generally, all these properties can be used to enhance the representation of an RRT. We first compute the tree properties and then visually encode those raw quantitative numbers in each link (i.e., line segment) of the RRT by color coding and line thickness. Both color coding and line segment thickness can be changed on user demand to control the visual appearance.

From experimenting with tree visualizations, we identified the tree depth as the most important property that should be enhanced to make the tree diagram more readable and aesthetically appealing in the sense of computational art. The advantage of the tree depth metric is that it naturally fits to the fractal characteristics of the RRT: the tree depth corresponds to the length scale of the depicted part of the tree and, accordingly, the line thickness is adapted.



(a)



(b)

Figure 4: *A selected region of an RRT: (a) Only equally thick line segments of the RRT are shown; color coding is used for the age of a segment. (b) The tree depth is visually encoded by the thickness of line segments and additionally enhanced by color coding.*

3.3 Rendering Algorithm

We use a pixel-based rendering algorithm for generating the visually enhanced RRTs. For each pixel of the final image, the vertex in the precomputed RRT with the least distance to that pixel is accessed in an array along with its corresponding value for the tree metric. From this information, we compute a tree-metric specific color by applying a color map. The color coding can be adapted by applying value mappings, e.g., with a logarithm function or by changing the exponent in the used formula; see Algorithm 2.

Algorithm 2 Visually Enhanced RRTs

```
Aesthetic_RRT($M$):
  // $M$ : Array with metric values
  // $x_{\max}$ : Number of pixels in $x$-direction
  // $y_{\max}$ : Number of pixels in $y$-direction
  // $p$ : Parameter to change thickness
  // $d_{\min}$ : Distance to closest vertex
  // $I_{\text{final}}$ : Final values
  for $i := 1$ to $x_{\max}$ do
    for $j := 1$ to $y_{\max}$ do
      $I_{\text{final}}[i][j] := d_{\min} \cdot \log^p(M[i][j])$;
    end for
  end for
  return $I_{\text{final}}$;
```

Algorithm 2 shows the details of our visually enhanced RRTs in pseudo code. We assume that a rectangular image with $x_{\max} \times y_{\max}$ pixels is used as rendering viewport. When generating an RRT (Algorithm 1), we already update arrays of metric values, i.e., for each pixel position, we store in an array M the metric value (i.e., depth, size, or branching factor) of the least distant RRT edge. This array is permanently being updated and finally used in Algorithm 2 to visually enhance the space-filling tree diagram by visually encoding the tree-specific metric. The resulting value is computed by $I_{i,j} := d_{\min} \cdot \log^p(M_{i,j})$. The parameter p can be used to vary the thickness of the line segments. The array I_{final} is used for the final rendering of the image, i.e., a user-defined color coding can be chosen and applied. This procedure allows the interactive update of the visually enhanced RRTs on user demand. The logarithm function may be deselected, but then the line segments representing branches closer to the root vertex look unnaturally thick compared to the smaller branches in very deep tree levels. Therefore, we use the logarithmic function for all enhanced RRT images of this paper.

4 Exploring the Design Space

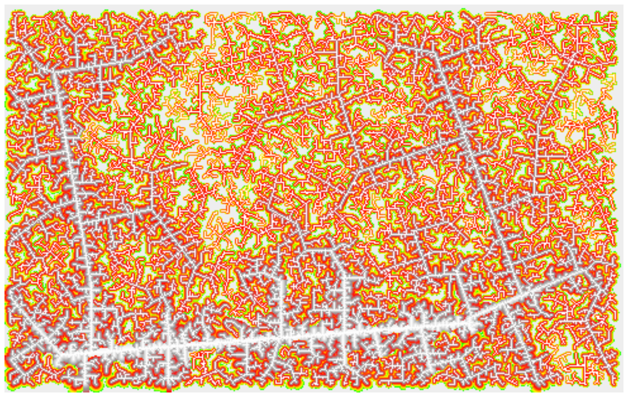
In this section, we provide several example images that serve as illustrative figures for the visually enhanced RRTs. With those examples, we briefly explore the design space of the control parameters for RRTs. The trees can be generated in the complete configuration space, i.e., there is no constraint for their expansion, obstacles can occur that have to be taken into account when generating the RRT and when visually enhancing them, and finally the configuration space itself can be bound to a given shape.

4.1 Constraint-Free RRTs

In the constraint-free variant of the RRT generation algorithm, we can let the tree expand over the complete (rectangular) display space, leading to rectangular shaped RRTs.

Figure 5 shows larger examples of RRTs after $N = 10,000$ iterations for combinations of parameter settings, i.e., for an RRT without an incremental distance where the initial vertex position is randomly chosen and where it is fixed in the display center. Here, we can see that the variant with the randomly chosen initial vertex position generates diagrams with more white space. This effect can be explained by the fact that many more tree branches are located in deeper levels than in the center-based approach, leading to much thinner line segments.

Figure 6 shows an RRT with $N = 1,000,000$ algorithm iterations. Also here, the main branches are visible by following the red colored and thicker line segments. The branchings into deeper subhierarchies can be detected by inspecting the line segments colored in green, whereas the deepest tree branches are visually encoded


(a)

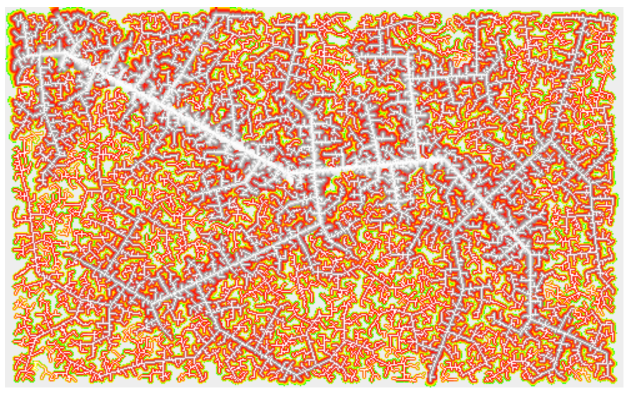

(b)

Figure 5: *Examples of RRTs after N = 10,000 iterations: (a) no incremental distance and randomly chosen initial vertex position, (b) no incremental distance and with the initial vertex position in the display center. There are no constraints for the RRTs, i.e., they can expand in the complete configuration space. Color coding and line thickness show tree depths; elements that are deeper in the tree are mapped to thinner lines.*

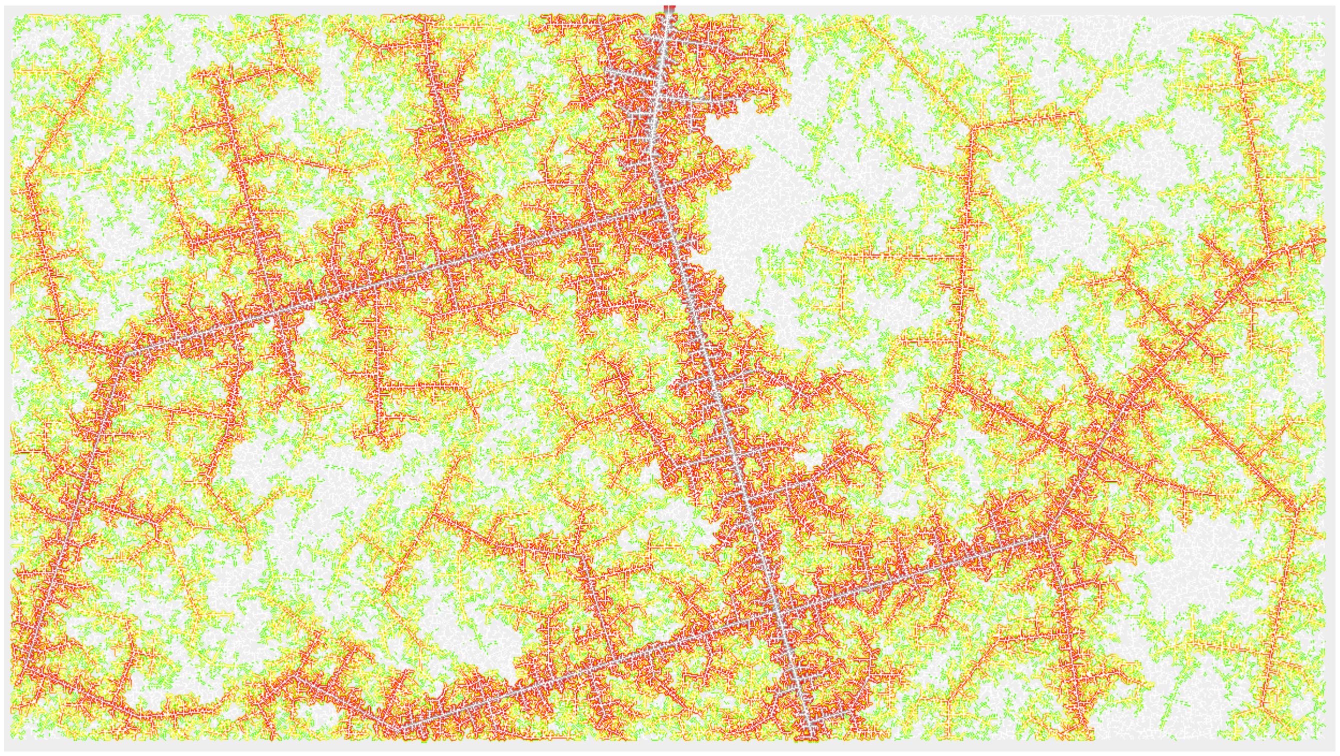


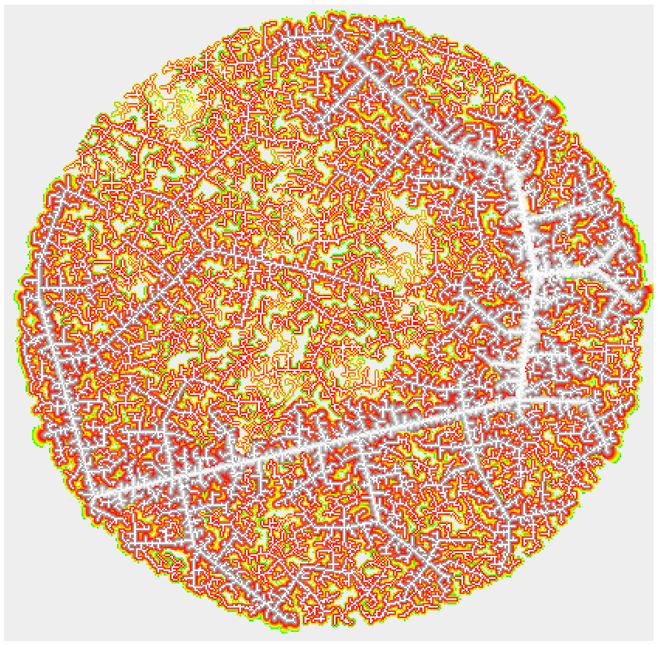
Figure 6: *An RRT after N = 1,000,000 algorithm iterations.*

in white. The structure of the tree becomes apparent and it is, for example, easy to visually ascend or descend along the tree.

4.2 Obstacle-Based and Constraint-Based RRTs

Obstacles are a natural phenomenon for growing tree-like structures that must be circumvented in some way. If we think of a city-shape metaphor, similar visual patterns as in RRTs occur when inspected from a bird's eye view. Larger and wider streets such as highways connect bigger city districts. Inside the single districts, streets become smaller and smaller until a hierarchical level of single households is reached. In this scenario, obstacles are also prominent, for example, in the form of parks or lakes; streets are not allowed to cross those obstacle regions and hence must be guided on a different way around them. For RRTs, a similar phenomenon can occur.

Figure 8 illustrates an example of such an RRT after 20,000 algorithm iterations. The algorithm cannot generate a tree diagram by extending it into the complete configuration space but, instead, a rectangular obstacle region must stay empty, i.e., no edges can cross it and hence, a different way has to be computed to organize the tree structure in the upper left region for example.


(a)

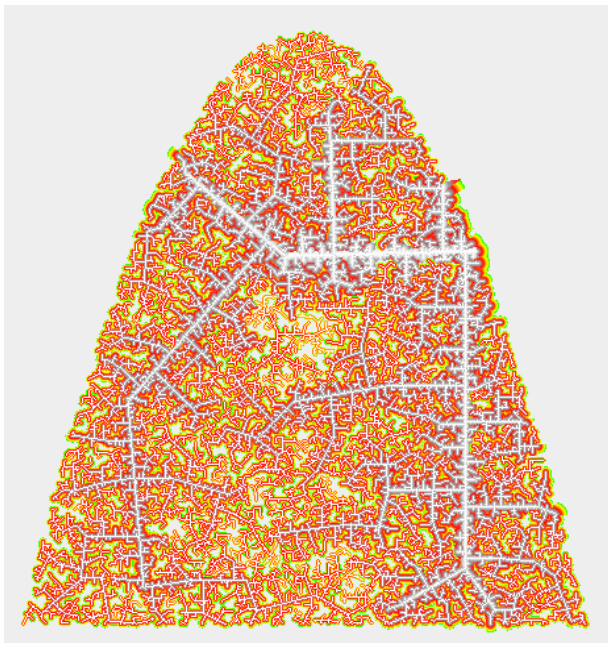

(b)

Figure 7: *Shape-driven RRTs: (a) circular shape $x^2 + y^2 < r^2$, (b) parabola shape $y - x^2 < r$.*

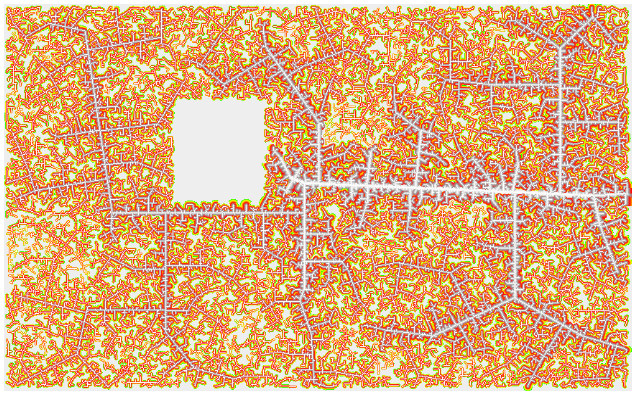


Figure 8: *An RRT after N = 20,000 algorithm iterations with a rectangular empty region illustrating an obstacle.*

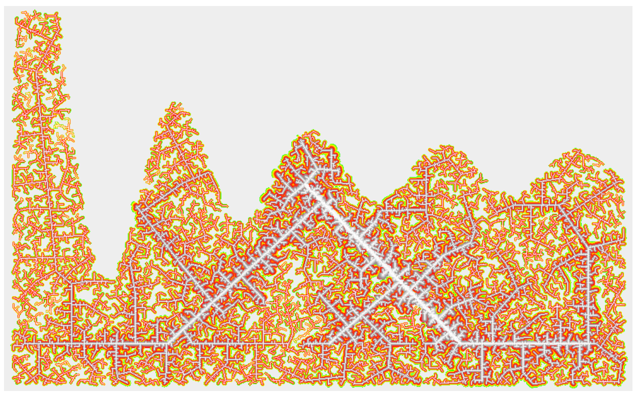


Figure 9: *Shape-driven RRT in a sine shape with decreasing amplitude $y - \frac{c}{x}\sin(x) < r$.*

4.3 Shape-Driven RRTs

Also, the shape on which an RRT is inscribed can be modified and controlled. We show how specific shapes such as a circle, a parabola, or a sine with decreasing amplitude can be used to generate a configuration space of a specific form in which an RRT is generated and then visually enhanced by the tree depth metric; see Figure 7 and Figure 9. Such a scenario can also occur in nature, e.g. in Diffusion-Limited Aggregation (DLA) where the space is typically limited. Also, the growing behavior of cities is limited in many cases due to their location close to an ocean or mountains.

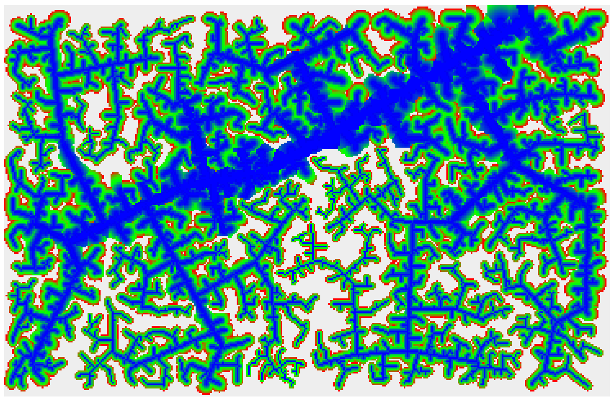
4.4 Color Coding and Line Segment Thickness

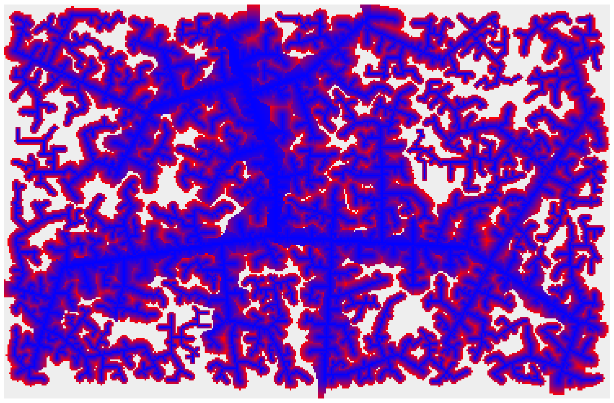
Our tool allows the user to interactively change the color coding as well as the parameter p for the line segment thickness. Figure 10 illustrates how the parameter changes influence the appearance of different RRTs.

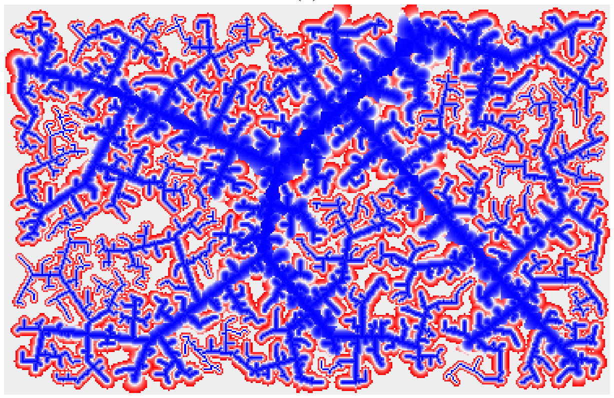
5 Discussion

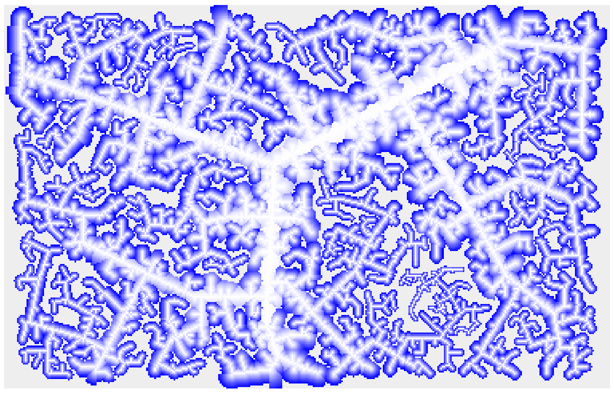
RRTs are driven by randomly selected positions for the vertices in an incremental way. This means that the user is not able to fully control the layout of the finally generated vertex positions. However, we do not think that this is a problem for algorithmic art. In fact, the aesthetics might benefit from some stochastic behavior, making the images come with some "natural" characteristics.

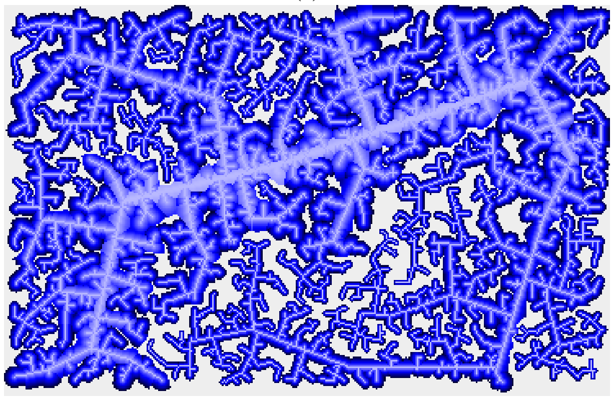
For the visual enhancement of the tree diagrams, we use line thickness and color coding. However, we do not have any fully computational way to determine those parameters. Yet, useful parameter choices depend on the size and structure of the RRT. For example, the segments should be not too thick because otherwise many overlaps with thinner segments will occur, which will result in overplotting and visual clutter [Rosenholtz et al. 2005]. For this reason, we allow the user to interactively adjust the line thickness and color coding, essentially delegating the responsibility for finding good parameter choices to the user.


(a)


(b)


(c)


(d)


(e)

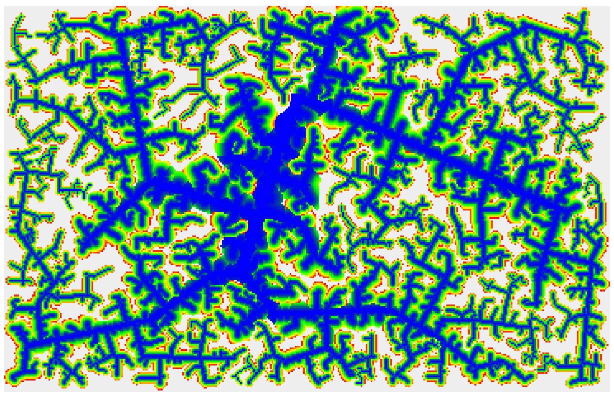

(f)

Figure 10: *RRTs with varying color coding: (a) blue-green-red, (b) cold-to-hot, (c) blue-white-red, (d) white-to-blue, (e) ocean color map, (f) vegetation color map. Any other color coding can be used on user's demand. Changes to the parameter p influence the thickness of the line segments. The trees are generated by 2,000 algorithm iterations.*

We argue that the tree depth metric is most suitable for making the RRTs more readable and aesthetically appealing. A user study could help find out if this is really the case. Apart from visualizing the tree depth, also the Horton-Strahler number may be of interest to make the diagrams more aesthetic and interpretable. It may also be discussed if the line segments can be used to visually encode more than one of tree-specific metric simultaneously.

We have restricted ourselves to using RRTs for algorithmic art. However, aesthetic usability could also help improve the effectiveness or efficiency of tree diagrams for data visualization. The question arises if a real-world abstract relational dataset might also be used as the basis for guiding the tree layout in an incremental way. In such a scenario, a distance function between two elements must be present in order to place the next processed node in the already existing layout. Possible application domains include social network visualization or software visualization. A hierarchical structure is typically generated by applying hierarchical clustering algorithms exploiting a given distance function, i.e., working directly on the abstract data and not by incrementally mapping the data points to spatial positions. If such a strategy is found, we will have the opportunity to algorithmically compute a hierarchical organization for relational data that can be represented in a space-filling node-link visual metaphor. The investigation of aesthetic RRT data visualization is left as open question for future work.

6 Conclusion and Future Work

We have illustrated how RRTs can be visually enhanced by adding information about tree-specific metrics, in particular, the tree depth. The tree metric is visually encoded by color mapping and the thickness of line segments. This visual enhancement allows a better readability and a more aesthetical visualization than those traditionally used to illustrate the RRT data structure. The construction and rendering of the RRTs can be affected and controlled by a number of user-specified parameters. We have illustrated the effects of those parameters for a couple of illustrative examples.

As discussed in the previous section, future work could extend the enhanced RRTs to the visualization of hierarchical data. Furthermore, our visualization approach could be adapted to other types of line-based diagrams with an additional line-oriented metric. One possible application could be graph visualization: traditional node-link diagrams might be presented in a way that shows the importance of relations by varying link thickness and color coding. Another application could be the visualization of trajectory data.

References

AMATO, N. M., AND WU, Y. 1996. A randomized roadmap method for path and manipulation planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 113–120.

BARNELEY, M. F., DEVANEY, R. L., AND MANDELBROT, B. B. 1988. *The Science of Fractal Images*. Springer, New York.

BAUER, A., AND POPOVIC, Z. 2012. RRT-based game level analysis, visualization, and visual refinement. In *Proceedings of the Conference on Artificial Intelligence and Interactive Digital Entertainment*.

BELLMAN, R. E. 1957. *Dynamic Programming*. Princeton University Press.

BOSCH, R., AND HERMAN, A. 2004. Continuous line drawings via the traveling salesman problem. *Operations Research Letters 32*, 4, 302–303.

BURCH, M., RASCHKE, M., AND WEISKOPF, D. 2010. Indented pixel tree plots. In *Proceedings of International Symposium on Visual Computing*, 338–349.

BURCH, M., KONEVTSOVA, N., HEINRICH, J., HÖFERLIN, M., AND WEISKOPF, D. 2011. Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study. *IEEE Transactions on Visualization and Computer Graphics 17*, 12, 2440–2448.

BURCH, M., ANDRIENKO, G., ANDRIENKO, N., HÖFERLIN, M., RASCHKE, M., AND WEISKOPF, D. 2013. Visual task solution strategies in tree diagrams. In *Proceedings of Pacific Visualization*, 169–176.

BURCH, M., SCHMAUDER, H., AND WEISKOPF, D. 2013. Edge bundling by rapidly-exploring random trees. In *Proceedings of the International Conference on Information Visualisation*.

KAPLAN, C. S., AND BOSCH, R. 2005. TSP art. In *Renaissance Banff: Bridges 2005: Mathematical Connections in Art, Music and Science*, 301–308.

KRUSKAL, J., AND LANDWEHR, J. 1983. Icicle plots: Better displays for hierarchical clustering. *The American Statistician 37*, 2, 162–168.

KUFFNER, J. J., AND LAVALLE, S. M. 2000. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 995–1001.

LAUMOND, J. P., SEKHAVAT, S., AND LAMIRAUX, F. 1998. Guidelines in nonholonomic motion planning for mobile robots. In *Robot Motion Planning and Control*, J. P. Laumond, Ed. Springer, 1–53.

LAVALLE, S. M., AND KUFFNER, JR., J. J. 1999. Randomized kinodynamic planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 473–479.

LAVALLE, S. M., AND KUFFNER, JR., J. J. 2001. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch, and D. Rus, Eds. A K Peters, 293–308.

MANDELBROT, B. 1982. *The Fractal Geometry of Nature*. W.H. Freeman and Company. New York.

MCGUFFIN, M., AND ROBERT, J. 2009. Quantifying the space-efficiency of 2D graphical representations of trees. *Information Visualization 9*, 2, 115–140.

NIETO, J., SLAWINSKI, E., MUT, V., AND WAGNER, B. 2010. Online path planning based on rapidly-exploring random trees. In *Proceedings of the IEEE International Conference on Industrial Technology*, 1451–1456.

PEITGEN, H.-O., AND RICHTER, P. H. 1986. *The Beauty of Fractals – Images of Complex Dynamical Systems*. Springer.

PEITGEN, H.-O., AND SAUPE, D., Eds. 1988. *Science of Fractal Images*. Springer.

PEITGEN, H.-O., JÜRGENS, H., AND SAUPE, D. 1992. *Chaos and Fractals – New Frontiers of Science*. Springer.

REINGOLD, E. M., AND TILFORD, J. S. 1981. Tidier drawings of trees. *IEEE Transactions on Software Engineering 7*, 2, 223–228.

RODRIGUEZ, S., TANG, X., LIEN, J.-M., AND AMATO, N. M. 2006. An obstacle-based rapidly-exploring random tree. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 895–900.

ROSENHOLTZ, R., LI, Y., MANSFIELD, J., AND JIN, Z. 2005. Feature congestion: A measure of display clutter. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 761–770.

ROSINDELL, J., AND HARMON, L. 2012. OneZoom: A fractal explorer for the tree of life. *PLOS Biology 10*, 10, e1001406.

SHNEIDERMAN, B. 1992. Tree visualization with tree-maps: 2-D space-filling approach. *ACM Transactions on Graphics 11*, 1, 92–99.

WETHERELL, C., AND SHANNON, A. 1979. Tidy drawings of trees. *IEEE Transactions on Software Engineering 5*, 5, 514–520.

WITTEN, T. A., AND SANDER, L. M. 1981. Diffusion-limited aggregation, a kinetic critical phenomenon. *Physical Review Letters 47*, 1400–1403.