Erschienen in: Graph Drawing and Network Visualization : 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers / Hu, Yifan; Nöllenburg, Martin (Hrsg.). - Cham : Springer International Publishing, 2016. - (Lecture Notes in Computer Science ; 9801). - S. 625-627. - ISBN 978-3-319-50105-5
https://dx.doi.org/10.1007/978-3-319-50106-2

Flexible Level-of-Detail Rendering for Large Graphs

Jan Hildenbrand(✉), Arlind Nocaj, and Ulrik Brandes

Department of Computer and Information Science, University of Konstanz, Konstanz, Germany
{jan.hildenbrand,arlind.nocaj,ulrik.brandes}@uni-konstanz.de

Motivation

The visualization of graphs using classical node-link diagrams works well up to the point where the number of nodes exceeds the capacity of the display. To overcome this limitation Zinsmaier et al. [5] proposed a rendering technique which aggregates nodes based on their spatial distribution, thereby allowing for visual exploration of large graphs. Since the rendering is done on the graphics processing unit (GPU) this process is reasonably fast. However, the connection between input graph and visual image is partially lost, which makes it harder, for instance, to process weights and labels of the input graph.

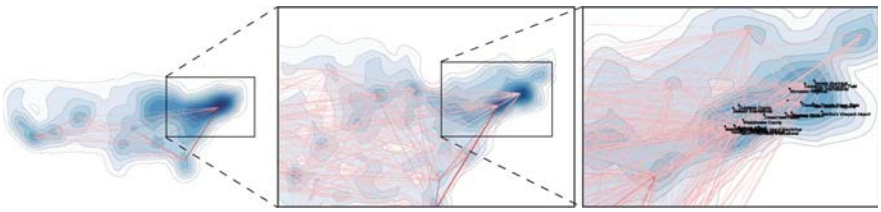

Fig. 1. Level-of-Detail rendering of the US air-traffic dataset.

We reproduce their approach with the goal of establishing a flexible structure to improve the connection between input data and visualization. Additionally, we control the layout features in a more direct way. For example, contour lines are explicitly drawn in order to remove fuzziness of the density visualization. Though the proposed CPU-based approach cannot render at interactive rates, it can be computed as a preprocessing step and then interactively explored given some predefined resolution constraints.

Approach

The visualization consists of two main parts: the construction of the terrain and the aggregation of the edges depending on the underlying terrain. Our terrain is a triangulation, where the triangle corners consist of the nodes of the graph. The node clutter is reduced by a density visualization where each node gets a height assigned and the resulting terrain is visualized.

The heights of the nodes are computed by a kernel density estimation (KDE), which approximates unknown density distributions by overlaying kernel functions at different positions. The density of a particular point is the sum over all kernel functions evaluated at that point. We use the Improved Fast Gaussian Transform (IFGT) [4], which takes on average $\mathcal{O}(N)$ time for N sources and N evaluations.

We use a Delaunay triangulation to create a triangulated irregular network in $\mathcal{O}(N \log N)$ time [1]. Each node of the TIN has a height assigned by the KDE and we assume a linear interpolation between two nodes on the TIN. Large triangles can lead to a false depiction of the graph of the terrain. For instance, let us assume that two neighboring nodes in the Delaunay triangulation represent a hilltop with a valley between them. Without an additional point between the hilltops the edge between them represent a ridge. Therefore, Ruppert's Delaunay refinement algorithm [2] is used to insert points in the circumcenter of triangles which have a minimum angle of 15° or triangles which are particularly large. Additionally, a convex hull is created around the input to prevent confusing non-closing contour lines.

A contour line represents all points with a specific height and is often used to visualize the 3D terrain of topographic maps. We extract equidistant contour lines with van Kreveld's find-isolines algorithm [3]. The contour lines are polylines because of the TIN and get smoothed with splines to be more visual pleasing. The contour lines form a hierarchy, which is used to aggregate the edges: A contour tree (i.e. a hierarchical representation of the contour lines, where a parent has a child if the child is completely contained in the parent) is constructed and only edges between leaves of the contour tree are created.

In practice many of the contour trees are degenerated and consist of list-like substructures. Nevertheless, there could be nodes that are not represented by the edge visualization and therefore, edges of the non-represented nodes are moved to the nearest (in Euclidean distance) leaf of the contour tree. Additionally, the aggregated edges are scaled in width and opacity depending on the sum of the weights of the original edges.

In our implementation of this approach, graphs with up to 42 thousand nodes and 1.5 million edges (e.g., the net150 graph from the University of Florida sparse matrix collection[1]) can be handled in less than 10 s.

References

1. Lee, D.T., Schachter, B.J.: Two algorithms for constructing a delaunay triangulation. Int. J. Comput. Inf. Sci. **9**(3), 219–242 (1980)
2. Ruppert, J.: A delaunay refinement algorithm for quality 2-dimensional mesh generation. J. Algorithms **18**(3), 548–585 (1995)
3. Van Kreveld, M.: Efficient methods for isoline extraction from a TIN. Int. J. Geogr. Inf. Syst. **10**(5), 523–540 (1996)
4. Yang, C., Duraiswami, R., Gumerov, N.A., Davis, L.: Improved fast gauss transform and efficient kernel density estimation. In: Proceedings of the Ninth IEEE International Conference on Computer Vision, vol. 1, pp. 664–671 (2003)
5. Zinsmaier, M., Brandes, U., Deussen, O., Strobelt, H.: Interactive level-of-detail rendering of large graphs. IEEE Trans. Vis. Comput. Graph. **18**(12), 2486–2495 (2012)