Flexible Trees: Sketching Tree Layouts

Javad Sadeghi
University of Calgary
jsadeghi@ucalgary.ca

Charles Perin
University of Calgary
charles.perin@ucalgary.ca

Tamara Flemisch
Technische Universität Dresden
tamara.flemisch@gmail.com

Mark Hancock
University of Waterloo

Sheelagh Carpendale
University of Calgary

mark.hancock@uwaterloo.casheelagh@ucalgary.ca

ABSTRACT

We introduce Flexible Trees, a sketch-based layout adjustment technique. Although numerous tree layout algorithms exist, these algorithms are usually bound to fit within standard shapes such as rectangles, circles and triangles. In order to provide the possibility of interactively customizing a tree layout, we offer a free-form sketch-based interaction through which one can re-define the boundary constraints for the tree layouts by combining ray-line intersection and line segment intersection. Flexible Trees offer topology preserving adjustments; can be used with a variety of tree layouts; and offer a simple way of authoring tree layouts for infographic purposes.

CCS Concepts

•**Human-centered computing** → **Information visualization;** *Gestural input; Dendrograms; Cladograms;*

Keywords

Visualization; trees; sketching; interaction; authoring; infographics.

1. INTRODUCTION

We present Flexible Trees, a sketch-based technique for deforming tree layouts. Flexible Trees offer a new approach to creating emphasis and focus+context, and a simple way of creating aesthetically pleasing customized hierarchical data visualizations.

Hierarchical data are ubiquitous (e.g., family trees and file system directory trees), and hierarchical data visualizations have a long history. Tree layouts are certainly the most standard family of hierarchical data representations. They usually represent data entities as the nodes of a tree, and the relationships between these data entities as links, adjacency, or nesting [14] (see [13] for an overview). While these tree layouts provide a variety of representations, they are generally designed to fit into formal shapes such as rectangles, circles, and triangles; and they emphasize the root of the tree.

Constraining tree visualizations to pre-defined shapes makes these layouts ill-suited to narrative visualizations [15] and storytelling [9] – where flexibility, authoring capabilities, and aesthetics are important.

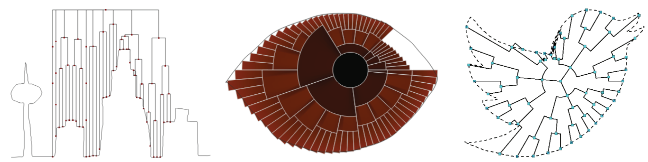

Figure 1: Tree layouts created using Flexible Trees, with the shapes of a city skyline, an eye, and Twitter's logo.

For example, one may want to illustrate a blog article with an aesthetically pleasing tree visualization that visually relates to the article; or may want to emphasize particular parts of the data, e.g., a sub-tree or a node. To empower non-experts in visualization, we designed a sketch-based interaction to deform tree layouts (see Figure 1) so that one can freeform layouts sketch directly instead of using buttons, menus and dragging operations.

We first discuss the literature related to Flexible Trees. Then we describe the Flexible Trees algorithm. Finally, we discuss the possibilities for authoring tree visualizations for infographic purposes.

2. RELATED WORK

Our work with Flexible Trees relates to research in the areas of tree layouts, sketch based interaction, and data story telling.

2.1 General background about trees

Tree layouts for visualizing hierarchical data can be implicit or explicit [14]. A treemap [7], where each child node is nested inside its parent, is an implicit layout which specifies each node size according to the underlying data, and the size of a node depends on the size of its children. Implicit layouts are not amenable to freeform deformation as it would interfere with the data representation.

In contrast, explicit layouts such as Reingold and Tilford's trees [11] encode data relationship in the tree structure, which can be maintained when distorting the tree. However, such tree layouts are automatically computed and are not interactively adjustable. Flexible Trees offer layout adjustment and personalization through sketching.

2.2 Sketch-based Interaction

Hand-drawn sketching is known to be effective in promoting innovation, creativity, and thinking [3, 6]. Studies have investigated how people manipulate representations, e.g., observing how people spatially arrange nodes and links [18]. These studies showed that pre-defined layouts are not in accordance with layouts people draw.

Researchers have considered sketch-based interaction [17] for visualization [9, 20] as pre-defined visualizations can limit people in expressing their thinking about data [19]. Also, NapkinVis [4] and SketchVis [2] investigate sketching simple visualizations. However, we found no work about sketching tree layouts.

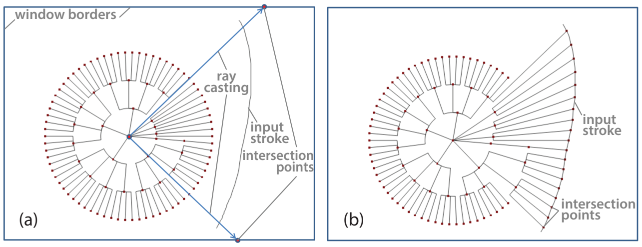

Figure 2: Finding intersection points a) of the rays with the window borders, and b) on the stroke, for a Radial Cladogram.

2.3 Data Story Telling

The visualization community is recognizing the importance of narrative visualization [15], and of telling data stories [9]. Throughout human history people have drawn trees to convey messages about hierarchical data. These hand-crafted trees differ widely: they usually have irregular layouts, make extensive use of metaphors and personalized styles, and contextualize data to convey a message. These historical trees relate strongly to modern infographics (see Lima's book [10] for a history of tree visualizations).

These growing movements towards both sketch-based interaction and narrative visualization have many benefits that have not yet been applied to the popular tree layouts.

3. FLEXIBLE TREES

Flexible Trees are a way of deforming tree layouts via sketching. We describe how to interpret sketched strokes; then we detail how to deform implicit tree layouts using ray casting and line intersections.

3.1 Sketch Interpretation

Let a sequence of n sketch points $p_i = (x_i, y_i)$ be a *stroke* $S = \{p_1, p_2, \ldots, p_n\}$. Sketched strokes have noisy and unevenly distributed sketch points. We filter the sketch points using four-point interpolatory subdivision [5] (DLG subdivision), which applies subdivision masks on the coarse points c_i to find the fine points f_i:

$$f_{2i} = c_i, \quad f_{2i+1} = -\frac{1}{16}c_{i-1} + \frac{9}{16}c_i + \frac{9}{16}c_{i+1} - \frac{1}{16}c_{i+2} \quad \text{(Eq. 1)}$$

which satisfies C^1 continuity at the limit. To filter the input strokes, we go in the reverse direction i.e. start from fine points to find the coarse points. We first apply the reverse DLG subdivision filter [1]:

$$\begin{aligned} c_i \; = & \tfrac{1}{64}f_{2i-4} + 0f_{2i-3} - \tfrac{1}{8}f_{2i-2} + \tfrac{1}{4}f_{2i-1} + \tfrac{23}{32}f_{2i} \\ & + \tfrac{1}{4}f_{2i+1} - \tfrac{1}{8}f_{2i+2} + 0f_{2i+3} + \tfrac{1}{64}f_{2i+4} \end{aligned} \quad \text{(Eq. 2)}$$

three times on the fine points f_i to find coarse points c_i. This filter discards high-frequency information and noise. Resulting coarse points are then subdivided three times using DLG subdivision (see Eq. 1) to create the filtered stroke. To filter open curves, we use the insights from J-splines [12] to derive boundary filters:

$$f_0 = c_0, \quad f_1 = \frac{7}{16}c_0 + \frac{10}{16}c_1 - \frac{1}{16}c_2 \quad \text{(Eq. 3)}$$

to find the beginning fine points $F = \{f_0, f_1\}$ from the beginning coarse points $C = \{c_0, c_1, c_2\}$. We then adjust the indexes to find the ending fine points. The rest of the fine points are found from Eq. 1. Next, we derive corresponding reverse subdivision filters:

$$c_0 = f_0, \quad c_1 = \frac{7}{16}f_0 - f_1 + \frac{26}{16}f_2 + 0f_3 - \frac{1}{16}f_4 \quad \text{(Eq. 4)}$$

to find the beginning coarse points $C = \{c_0, c_1\}$ from the beginning fine points $F = \{f_0, f_1, f_2, f_3, f_4\}$. We adjust the indexes to find the ending coarse points and Eq. 2 gives the rest of the coarse points.

Finally, the control points are simply the coarse points. With this approach, one can manipulate the stroke by moving control points directly on the stroke – which is not the case with B-Splines.

3.2 Layout Deformation: Radial Cladogram

We first illustrate Flexible Trees with the Radial Cladogram (see Figure 2). This explicit tree layout arranges the nodes of a tree in a circle centered around the root node: 1) the leaf nodes are evenly spaced on the circumference of the circle; 2) the internal area of the circle is sliced into d concentric rings; 3) each parent node is positioned on the inner ring corresponding to its depth such that it bisects the angular distance between its children; 4) finally, the edge originating at a child is drawn towards the root; when this edge intersects the concentric ring of its parent it forms a join and completes the edge line of the ring. Figure 2(a) shows a Radial Cladogram and a sketched stroke S that sets the layout boundary.

We first find the closest intersection point between the rays shot from the tree's center to the borders of the window. Rays are shot at uniform angular intervals, with one ray being shot for each leaf node. A ray $R(t) = p + td$ is defined by a point $p = (p_x, p_y)$ and a unit direction $d = (\cos\alpha, \sin\alpha)$; $t \geq 0$ is a scalar indicating time. Let w and h be the width and height of the window. For each ray, we compute the intersection times with each border, with $t_{left} = -\frac{p_x}{\cos\alpha}$, $t_{right} = \frac{w-p_x}{\cos\alpha}$, $t_{bottom} = -\frac{p_y}{\sin\alpha}$, and $t_{top} = \frac{h-p_y}{\sin\alpha}$. Given $t^* = min(t_{left}, t_{right}, t_{bottom}, t_{top})$, we find the closest intersection point $p^* = (p_x + t^*\cos\alpha, p_y + t^*\sin\alpha)$ for each ray.

Next, we find all intersection points between the line segments running from root to p^*s and the line segments of S as follows. Let $p_1^* = (a_1, b_1)$ and $p_2^* = (a_2, b_2)$ be the two extremities of the line segment running from the root to p^*, and $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$ a line segment of S. Then we use the slopes $m^* = \frac{b_2-b_1}{a_2-a_1}$ and $m = \frac{y_2-y_1}{x_2-x_1}$ to find the coordinates of the intersection point (x, y):

$$\begin{aligned} x &= \frac{m^*a_1 - b_1 + y_1 - mx_1}{m^* - m} \\ y &= \frac{mm^*a_1 - mb_1 - mm^*x_1 + m^*y_1}{m^* - m} \end{aligned} \quad \text{(Eq. 5)}$$

If S intersects multiple times with the ray, we choose the intersection closest to the root. The last step is to adjust the radius of the Radial Cladogram if a ray intersects the stroke. We calculate the distance between the intersection point and the root node and use it as the leaf node's new radius. Finally, we update the position of all ancestors of this leaf node (except the root) using the average radius of the leaf nodes in the corresponding subtrees, as shown in Figure 2(b).

3.3 Other Layout Deformations

We have detailed the Flexible Trees algorithm for the Radial Cladogram (a node-link radial layout). We now describe how to adapt the algorithm to a rectilinear node-link layout (Cladogram), a rectilinear space-filling layout (Icicle Plot) and a radial space filling layout (Sunburst). Figure 3 shows these layouts being deformed.

The **Cladogram**, or *dendogram*, is the rectilinear version of the Radial Cladogram. All leaves are drawn uniformly on the bottom of the tree, regardless of their distance from the root, or depth, d. Then the space between root and leaves is equally sliced according to d at that point. Parents are aligned to be at the center of their children, and the edge lines are drawn from parents to their children with a 90° angle turn at the same horizontal position of the children.

Implementation: As this layout only allows changes to the area below the root node, we need only find the intersections between S and the vertical line segments V from the root to the bottom border. One vertical line $v \in V$ is shot for each leaf vertex. We find the closest intersection point p^* for v and update the layout to be bound at p^* similarly to updating the radius of the Radial Cladogram tree.

The **Icicle Plot** [8] is a rectilinear space-filling layout. It only represents nodes – as filled rectangles; edges are implied by adjacency. All nodes have the same height and are drawn at their exact depth. Parent nodes are sized according to their number of children.

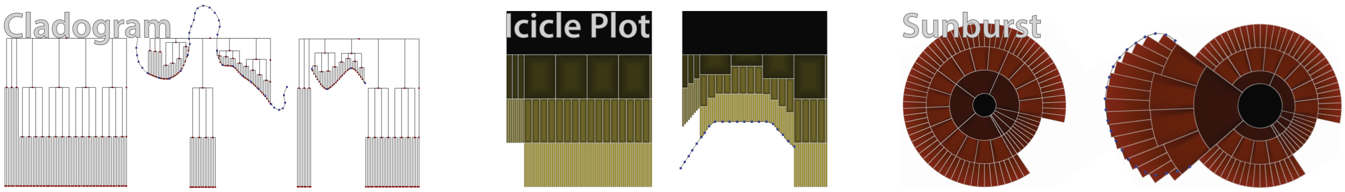

Figure 3: Adapting a Cladogram, an Icicle plot, and a Sunburst layout.

Implementation: The approach is similar to the one for the Cladogram. However, because nodes are arranged at their exact depth, the intersection between S and v is not necessarily a leaf node. Thus, we first uniformly divide the vertical distance between the root and the intersection point; then we locate each node based on its depth.

The **Sunburst** [16] is the radial version of the Icicle Plot.

Implementation: The approach uses the ray casting and line segment intersection algorithms of the Radial Cladogram; and arranges the nodes at the level proportional to their depth as in the Icicle Plot. These levels are represented by annuli between co-centered rings around the root. After finding an intersection point p^*, we uniformly divide the radial distance between the root and p^* to find the inner and outer radius of the annuli using the depth of the node.

4. DISCUSSION AND FUTURE WORK

We discuss applications of sketch-based layout and show the applicability of Flexible Trees to the authoring of infographics.

4.1 Layout Deformation via Sketching

Our sketch interpretation method can interpret freeform shapes. However, our layout deformation algorithm uses the closest stroke intersection after ray casting. We plan to investigate more complex shapes such as concave shapes by adapting the curvature of links and moving nodes, e.g., using physics.

Our algorithm produces fine results for node-link layouts. However, deformed space-filling layouts are more difficult to understand. We plan to explore new strategies for such layouts in future work.

Finally, we implemented the algorithm for four implicit layouts but adapting adjacency and nested layouts is a challenge. For example, drawing an arbitrary shape and fitting a treemap into it raises problems regarding maintaining the relative sizes of the nodes.

4.2 Flexible Trees for Focus+Context

Flexible Trees can create focus+context views. Using a sketched curve to expand part of the layout is similar to what can be achieved with a fisheye magnifying glass, but it acts differently. Figure 4 shows a sketched focus+context view of a Radial Cladogram: the leaf nodes of the condensed tree (a) are difficult to distinguish visually; but a simple hand-sketched stroke can open up the subtrees sitting next to that curve (b). It makes the nodes readable locally.

4.3 Flexible Trees for Infographics

Flexible Trees are extremely suitable to generating infographics. By drawing different shapes one can add more information about the tree itself (e.g., provide the context of the data) and about its data (e.g., provide information about the tree nodes and links). In this section, we provide examples of infographics that we created by modifying the original outputs of the algorithm in an SVG editor.

Figure 5 shows a tree in the shape of a cup that represents a subset of the **SCAA Flavor Wheel** dataset. The shape does not add data but it contextualizes the tree by conveying the topic. The coffee cup shape hints at a beverage and the brown colour is typical for coffee.

Figure 7 shows data from **WorldLifeExpectancy**. This pie chart shows the top causes of death for the United States, Germany and

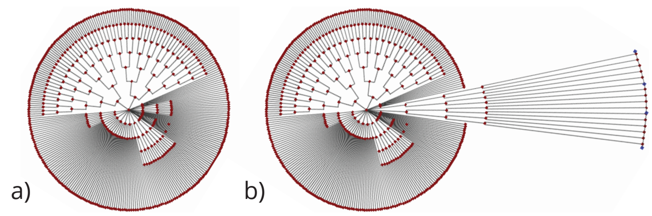

Figure 4: Focus+Context effect using free-form strokes on a dense Radial Cladogram layout.

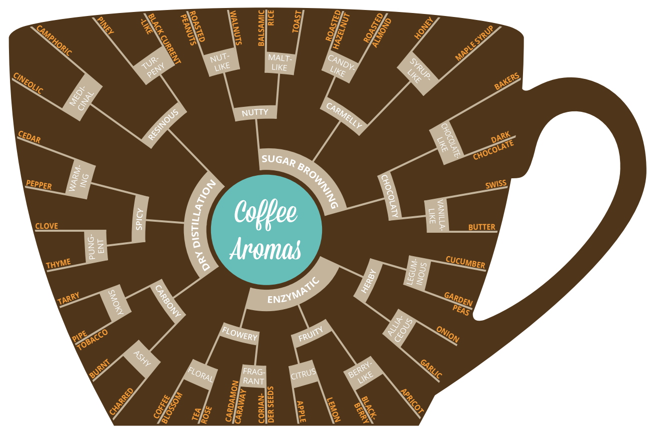

Figure 5: Infographic based on a tree that represents coffee aromas in a coffee cup shape.

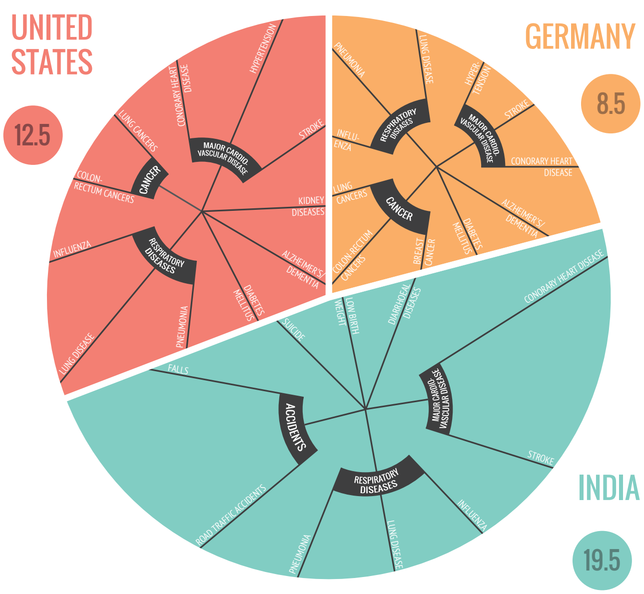

Figure 6: Pie chart showing the birth rates of Germany, the United States, and India. Inside the pie slices the trees represent the most common death causes for each country.

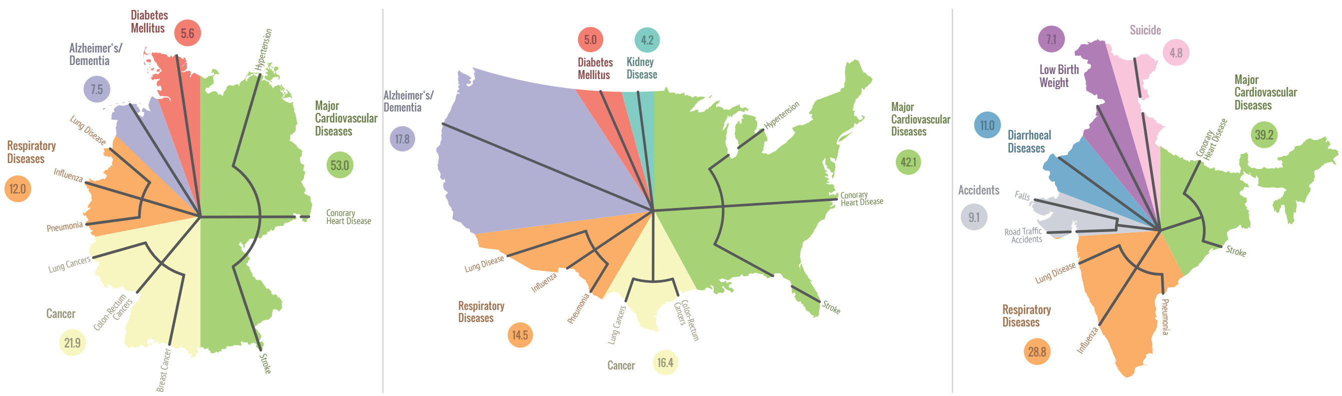

Figure 7: Infographics using small multiples of deformed trees in the shape of Germany, the United States, and India. The trees and the coloured areas represent the most common causes of death and their percentage.

India. The shape adds quantitative data: slice angles represent birth rates for each country. In addition to conveying the information contained in the trees, this infographics puts this information in perspective by also comparing the areas of the trees in the pie chart.

Figure 6 shows data from **WorldLifeExpectancy**. Color distribution and slice angles represent the impact of each cause of death on total percentage of deaths. The shape adds information to the tree: it conveys the country that the data is related to. Filling parts of the countries also conveys quantities like a pie chart does.

Flexible Trees benefit infographics as they make it possible to include context (e.g., coffee cup), qualitative data (e.g., country shapes), and quantitative data (e.g., birth rates in a pie chart layout).

5. CONCLUSIONS

Flexible Trees is a topology-preserving distortion technique for adapting the layout of trees according to hand-sketched strokes.

This initial exploration of sketch-based layout paves the way for further research. That includes exploring hand-sketching algorithms for implicit and nested tree layouts; and beyond trees, applying the method to e.g., graph drawing. Generalizing this approach has the potential to reach a large number of people, as it makes it possible to easily and rapidly create layouts of customized shapes. Our discussion of the applicability to infographics generalizes to other visualization types such as graph visualizations, and contributes to filling the gap between Infovis research and infographic design.

6. ACKNOWLEDGMENTS

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), SMART Technologies, and Alberta Innovates Technology Futures (AITF). We also thank Lindsay MacDonald for proofreading the paper.

7. REFERENCES

[1] R. H. Bartels and F. F. Samavati. Reversing subdivision rules: local linear conditions and observations on inner products. *Journal of Computational and Applied Mathematics*, 119(1-2):29–67, 2000.

[2] J. Browne, B. Lee, S. Carpendale, N. Riche, and T. Sherwood. Data analysis on interactive whiteboards through sketch-based interaction. In *Proc. ITS '11*, pages 154–157. ACM, 2011.

[3] B. Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, 2007.

[4] W. O. Chao, T. Munzner, and M. van de Panne. Napkinvis: Rapid pen-centric authoring of improvisational visualizations. 2010. Posters Compendium InfoVis 2010.

[5] N. Dyn, D. Levin, and J. A. Gregory. A 4-point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, 4(4):257 – 268, 1987.

[6] S. Greenberg, S. Carpendale, N. Marquardt, and B. Buxton. *Sketching user experiences: The workbook*. Elsevier, 2011.

[7] B. Johnson and B. Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proc. IEEE Visualization'91*, pages 284–291, 1991.

[8] J. B. Kruskal and J. M. Landwehr. Icicle plot: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.

[9] B. Lee, R. H. Kazi, and G. Smith. Sketchstory: Telling more engaging stories with data through freeform sketching. *IEEE TVCG*, 19(12):2416–2425, 2013.

[10] M. Lima. *The Book of Trees: Visualizing Branches of Knowledge*. Princeton Architectural Press, 2014.

[11] E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Trans. Softw. Eng.*, 7(2):223–228, Mar. 1981.

[12] J. Rossignac and S. Schaefer. J-splines. *Comput. Aided Des.*, 40(10-11):1024–1032, Oct. 2008.

[13] H. Schulz. Treevis.net: A tree visualization reference. *IEEE CG&A*, 31(6):11–15, Nov 2011.

[14] H.-J. Schulz, S. Hadlak, and H. Schumann. The design space of implicit hierarchy visualization: A survey. *IEEE TVCG*, 17(4):393–411, 2011.

[15] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *IEEE TVCG*, 16(6):1139–1148, Nov. 2010.

[16] J. Stasko and E. Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Proc. InfoVis'00*, pages 57–65, 2000.

[17] I. E. Sutherland. Sketch pad a man-machine graphical communication system. In *Proc. SHARE Design Automation Workshop*, pages 6.329–6.346. ACM, 1964.

[18] F. van Ham and B. Rogowitz. Perceptual organization in user-generated graph layouts. *IEEE TVCG*, 14(6):1333–1339, Nov 2008.

[19] J. Walny, S. Huron, and S. Carpendale. An Exploratory Study of Data Sketching for Visual Representation. *Computer Graphics Forum*, pages 231–240, 2015.

[20] J. Walny, B. Lee, P. Johns, N. H. Riche, and S. Carpendale. Understanding pen and touch interaction for data exploration on interactive whiteboards. *IEEE TVCG*, 18(12):2779–2788, Dec 2012.