Sketchy Rendering to Aid the Recollection of Regular Visualizations

M. Larsen[1], W. Han[1], and H.-J. Schulz[1]

[1]Aarhus University, Aarhus, Denmark

Abstract

Some visualizations have a more regular visual appearance than others. For example, while stream graphs or force-directed network layouts feature a unique, almost organic look&feel, matrices or unit treemaps can become rather bland, grid-like visualizations in which one data item is hard to tell apart from the next. In this paper, we investigate the use of sketchy rendering for such grid-like visualizations to give them a slightly more unique look&feel themselves. We evaluate our approach in a lab study ($N = 16$) where participants were asked to re-find a given grid cell in regular and sketchy grids. We find that users who make conscious use of the sketchy features can benefit from certain forms of sketchy rendering in terms of task completion times.

CCS Concepts

• ***Human-centered computing*** → ***Visualization techniques;*** • ***Computing methodologies*** → ***Graphics systems & interfaces;***

1. Introduction

In the context of visualization, sketchy rendering is a drawing style that mimics the imperfections of human-drawn sketches – e.g., a slightly shaking hand, varying pressure along strokes, or lines overshooting at connection points or corners. The sketchy rendering style is usually applied by perturbing contours of geometric shapes and applying hachures to fill them in a manner a human would [WII*12]. More recently, extensions for 3D graphics [LFH*16] and the use of stippling [GSS*19] have been investigated. So far, sketchy rendering has been shown to work reasonably well for indicating uncertainties in data [BBIF12,CKB*19], yet not so much for de-emphasizing imputed values [SS19].

We propose another use of sketchy rendering: enhancing visualizations that have a very regular appearance so as to improve the discernibility and memorability of their individual visual elements. Such regular visualizations can, for example, be matrices exhibiting a grid-like look and feel that makes it very hard to orient oneself on a large scale and equally challenging to discern one matrix cell from another on a detailed scale (cf. Fig. 1). But also unit-size treemaps are known to produce very regular, grid-like layouts [WD08]. The most common way to counter the uniform appearance of these visualizations is to use some form of coloring or shading of the individual elements – e.g., using a striped pattern [EDG*08] or a multi-hue color mapping [WD08].

In this work, we aim to achieve a similar effect through the use of sketchy line drawing. There are several reasons for choosing sketchy lines over fill colors for this task:

- Sketchy lines are most likely not used as a visual variable to encode data in the first place, as they are a rather poor channel that can only accurately convey between three and four different ordinal levels [BBIF12].
- Sketchy lines are first and foremost perceived as an element of visual style, not so much as carrying actual meaning [BBIF12], so that added sketchiness is less prone to be misinterpreted as a data characteristic.
- Sketchy lines perturb the gridded outline of the visualization layout, but leave the interior of the visual element available for showing additional data attributes.

To this end, we make a two-fold contribution: First, we propose an extended algorithm for sketchy line drawings that is highly parametrizable and which can thus be modulated and altered to generate grids with different regional appearances. And second, we evaluate the generated sketchy grids in a user study that looks at whether sketchy grids improve visualization users' ability to re-find a specific element in them.

2. Sketchy Rendering for Regular Visualizations

The rendering of complex line geometries, strokes, and textures to generate a rich and natural visual appearance has seen continued interest over the years. Early works include multiresolution curves [FS94] for editing the appearance of complex lines, as well as using differently styled lines for creating depth cues similar to hand-drawn sketches [Elb95]. In visualization, these techniques have been used for the illustrative rendering of anatomical geometry [SSSS98] and of flow field datasets [EBRI11].

In our work, we are mainly interested in creating a highly parametrizable sketchy line drawing, so that we can modulate its

DOI: 10.2312/evs.20201062

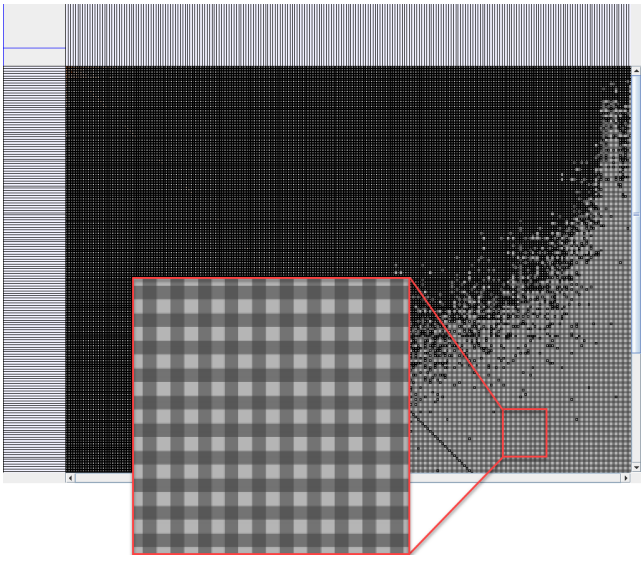


Figure 1: *Adjacency matrix of Wikipedia data. Despite the striped pattern, the visualization exhibits in most parts a uniformity of display that impedes the recognition and recollection of data cells of interest. Image adapted from* `https://engineering.purdue.edu/~elm/projects/zame.html`

parameters differently in different regions of a visualization to create unique regional appearances. We detail our approach in three steps: first we introduce our parametrizable sketchy line model, then we discuss its use for drawing rectangular shapes, and finally we show how to use either lines or shapes to generate entire layouts.

2.1. Parametrizable Sketchy Lines

Much of the literature is concerned with creating sketchy lines that are as close to hand-drawn lines as possible, going as far as using mathematical models of the physics behind human arm movements [AWI*09]. Our goal is different, as we are trying to generate as many distinctive looking line styles as possible without regarding their likeness to human drawings. Unhindered by the constraint of realism, we extend the line model introduced by Wood et al. [WII*12]. In addition to the end points of a line, their model introduces two randomly placed control points that are then interpolated using Catmull-Rom-splines [CR74]. The intervals for placing the two control points are the only parameters of their model.

Our principal idea for extending this model is to increase the number of randomly placed control points, so that we have more flexibility and control over the appearance of each line. As a result, the following parameters govern the geometry of a line, with each of them also being illustrated in Figure 2:

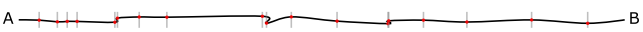
- ***p*: The number of control points.** The more control points one uses, the more deviations from a straight line will appear, and thus the more ragged will the resulting sketchy line look.
- ***a*: The amount of displacement on each control point.** This amount captures how far from the straight line a control point can be placed – i.e., how much the line is contorted at each point. It is randomly chosen from an interval of $[-a,+a]$.
- ***d*: The distribution of the control points along the line.** Here we use the Bates distribution [Bat55], where $d = 1$ yields a uniform distribution and $d \geq 2$ yields a probability distribution skewed towards the center of the line. For $d = -n$, we compute the "inverse" of the Bates distribution for n to generate a skew towards the endpoints of the line:

$$invBates(x,n) = \begin{cases} Bates(x+0.5,n) & x < 0.5 \\ Bates(x-0.5,n) & x \geq 0.5 \end{cases}$$

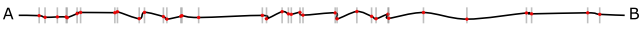
- ***o*: The order in which the control points are connected.** We specify the order as a list of offsets in which the points are used – e.g., $[+1]$ for a simple sequential order or $[+3,-1]$ to connect the points in order $1,4,3,6,5,8,7,10$, etc. for an overdrawn line in a back&forth style.

Finally, we can also use the control points to vary the stroke width along the line through the following two parameters – both being illustrated in Figure 2(f):

- ***sw*: The stroke width to be used.** If it shall be varied, this is usually an interval of permissible stroke widths from which a random value is chosen when changing the stroke.
- ***sc*: The count of points for which a stroke retains its width.** $sc = 1$ means that at every control point the stroke width will be changed, $sc = 2$ means that the stroke width is to be changed after every second point, and so forth until $sc = p$ means that the stroke width is only to be assigned once in the beginning.



(a) A sketchy base line against which to compare the following.
$p = 18$, $a \leq 10$, $d = 1$, $o = [+1]$, $sw = 2$, $sc = 0$



(b) Increasing the number of control points.
$p = 36$, $a \leq 10$, $d = 1$, $o = [+1]$, $sw = 2$, $sc = 0$

(c) Increasing the amount of displacement at each control point.
$p = 18$, $a \leq 18$, $d = 1$, $o = [+1]$, $sw = 2$, $sc = 0$

(d) Placing the control points more towards the middle of the line.
$p = 18$, $a \leq 10$, $d = 3$, $o = [+1]$, $sw = 2$, $sc = 0$

(e) Jump over points in the connection order for back&forth effect.
$p = 18$, $a \leq 10$, $d = 1$, $o = [+3,-1]$, $sw = 2$, $sc = 0$

(f) Change stroke width at every third point to value between 2...8.
$p = 18$, $a \leq 10$, $d = 1$, $o = [+1]$, $sw \in [2..8]$, $sc = 3$

Figure 2: *The different sketchy line parameters in effect.*

2.2. Turning Sketchy Lines into Shapes

To draw shapes using our sketchy lines, we have in principle two fundamental options: concatenate the lines into a path or keep the lines disjoint. Both are illustrated in Figure 3. Concatenation allows us to generate shapes that look as if drawn with a single movement. This effect is reinforced by matching the stroke widths at the joints of the individual lines. The opposite effect is achieved by keeping the individual lines as such. Gaps and overspill at their connection points can be created by randomly displacing the endpoint of each line in a similar manner as described by Wood et al. [WII*12].

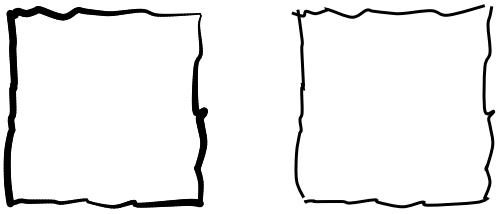


Figure 3: *Different shape styles generated from the same four lines. Left: Concatenated lines to create a continuous shape. Right: Individual lines to create a shape as if drawn with multiple strokes.*

Note that when rendering such sketchy shapes, a number of additional considerations need to be made, which include the style of the line cap and the need to combine the individual lines into a closed path if a fill color is to be assigned to the sketchy shape. With these aspects in place, any shape consisting of lines can be "sketchy-fied" using the extended sketchy line model from Section 2.1, as is demonstrated in Figure 4.

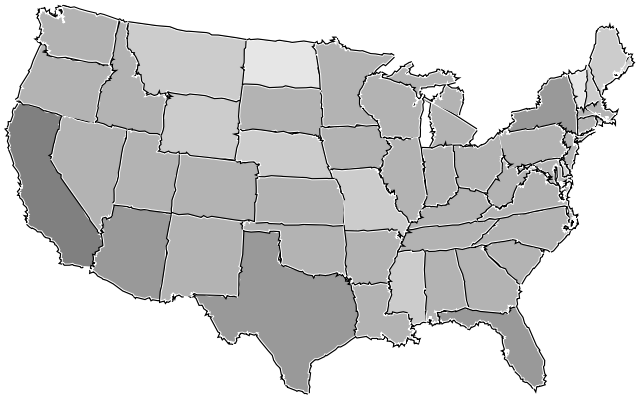


Figure 4: *A sketchy map with each state/shape being "sketchy-fied" using the line parameters* $p = 10$, $a = 4$, $d = 1$, $o = [+1]$, $sw = 5$, $sc = 0$. *Image adapted from Wikipedia.*

2.3. Generating Sketchy Grid-Like Layouts

For highly symmetric grid-like layouts, the literature is mainly concerned with means of distortion to either reflect properties of the data (e.g., registration errors [BFW02]) or interactive changes (e.g., of the user focus [SSTR93]). In contrast to these approaches, we do not aim to distort the grid, just to perturb the grid line to some degree using our sketchy rendering.

For this, we propose two approaches: a *line-based approach* that pieces together the grid from individual sketchy lines of length equal to the grid cell size, and a *shape-based approach* that uses sketchy squares of grid cell size instead. By using these small lines/shapes, we can alter the appearance for each grid cell individually. This would not be possible when, for example, using long sketchy lines that run across the whole grid.

To determine the line/shape parameters, we generate random 2D Perlin noise fields [Per85] – one for each of the four line parameters we vary: p, a, d, sw. An example of such a noise field can be seen on the left of Figure 5 with low values shown as white and high values shown as black. We then superimpose the grid on the noise fields. For the line-based approach, we determine the noise value at the middle of each individual line and parametrize it proportionally. This can be seen in the grid on the right side of Figure 5 that uses the noise field on the left to set the stroke widths sw of its sketchy lines. The shape-based approach does it likewise, but takes the noise value at the center of each grid cell and uses this value to proportionally parametrize all four sides of the shape in the same way. By using these random noise fields to assign parameter values instead of simply assigning any random parameter value, we yield parameter assignments that create smooth transitions without too abrupt changes between different styles of sketchy lines.

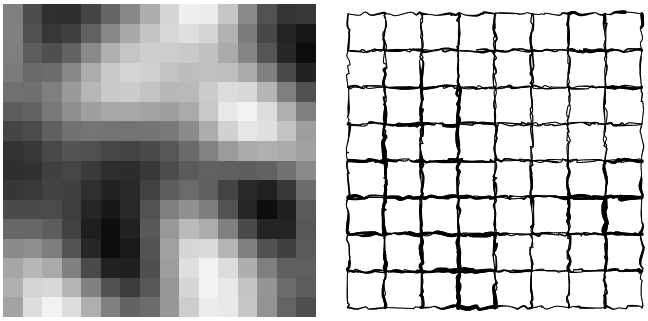


Figure 5: *2D Perlin noise field and its mapping onto* $sw \in \{1..5\}$.

Setting the line parameters according to different Perlin noise fields yields grid regions with different line styles – from thin lines with barely noticeable jitter on both ends of the line to thick lines with a singular huge bump in the middle. The outcome for both grid styles – line-based and shape-based – can be seen in Figure 6.

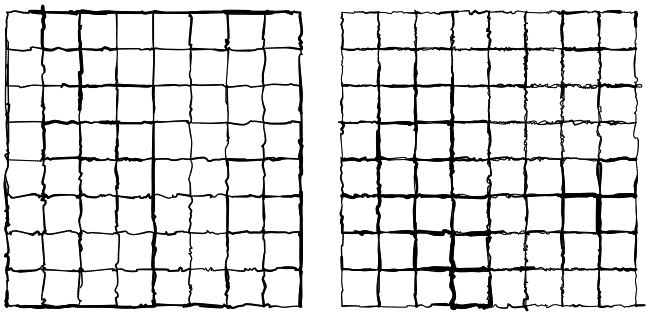


Figure 6: *Line-based (left) and shape-based (right) grids with* $p \in [1..18]$, $a \in [1..5]$, $d \in [\pm 1..\pm 15]$, $o = [+1]$, $sw \in [1..5]$, $sc = 2$.

3. Evaluation of Sketchiness for Recollecting Grid Locations

We created the sketchy grids with the intent of improving users' ability to orient themselves and to re-find cells of interest, as opposed to regular grids with straight lines. This section reports on a small lab study we did ($N = 16$, all students or researchers, ages between 23 and 34, 5 female and 11 male) to evaluate whether users actually use the sketchy line features to this end and if so, whether it indeed leads to improvements.

3.1. Setup of the Evaluation

The participants were asked to perform a memorization task on 16×16 sized grids – straight-line grids, line-based sketchy grids, and shape-based sketchy grids, with the sketchy line parameters specified as in Figure 6. To randomize the order of the different grid types, the study was divided into 6 rounds. In each round, one trial was performed on each grid type with the order of the grid types being randomized. This results in $6 \times 3 = 18$ trials overall, which took the average participant around 10 minutes in total.

On these grids, for each trial one grid cell among the inner 14×14 cells was chosen at random, highlighted in red, and shown to the participant for 5 seconds. The restriction to inner grid cells was made to ensure that all highlighted cells are similar in the sense that they are surrounded by eight other cells. Afterwards, the participants were asked to answer a simple math question as a distraction, for which we used a 2-digit by 1-digit multiplication. Once completed, they were shown the same grid again and asked to re-find the location of the previously highlighted cell and click on it.

For each trial, we recorded the timestamp and position of every mouse click together with the position of the target cell and the grid type. From this raw data, we computed the following time and error metrics for further analysis:

- Time: For those trials in which the participants hit the target cell on their first try, this measure captures the time span between showing the grid and registering the mouse click.
- Error: For those trials in which the participants did not hit the target cell on their first try, this measure captures the Manhattan distance between the clicked cell and the target cell in the number of cells.

In case other metrics like the number of tries are to be used at a later point or to compare with other studies, we made the logged data available together with the software used in the evaluation at `https://vis-au.github.io/sketchyrendering`. Given this setup, our hypotheses were as follows:

H1: The average time for re-finding the highlighted cell on sketchy grids will be significantly lower than the one for straight-line grids.

H2: The average error of re-finding the highlighted cell on sketchy grids will be significantly lower than the one for straight line grids.

3.2. Results of the Evaluation

Among all participants, the results of the evaluation were inconclusive with the only statistically significant result ($p < 0.01$) being that line-based grids ($\mu = 4.135s$, $\sigma = 1.378s$) yielded better response times than the shape-based grids ($\mu = 5.297s$, $\sigma = 1.974s$).

Looking at the post-study interviews, it became clear that different participants followed different strategies in solving the task. This was to be expected, as we did not instruct the participants beforehand on the possible use of sketchiness and thus some discovered it as a feature that they actively and consciously used, while others did not pay much attention to it.

For the subset of those 6 participants who reported in the post-study interview to have used the sketchiness, we found that their response times with line-based sketchy grids ($\mu = 3.151s$, $\sigma = 0.729s$) were significantly better ($p < 0.05$) than with both – straight-line grids ($\mu = 5.237s$, $\sigma = 2.009s$) and shape-based grids ($\mu = 4.654s$, $\sigma = 1.548s$). Yet their differences in errors remained statistically inconclusive. The results for this subgroup are shown in Figure 7.

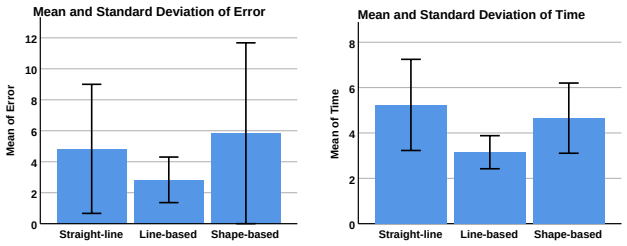


Figure 7: *Means and standard deviations of error (left) and time (right) for participants who reported conscious use of sketchiness.*

When asked in the interview what exactly their strategies for solving the task were, all participants reported to have counted in one way or another in most instances. The difference between the counting strategies was that the conscious users of the sketchiness counted from some nearby visual anchor produced by the sketchy rendering, whereas the other participants counted from the nearest borders or from the diagonals. This would explain the better response times with the line-based grids, as these produce more such visual anchors – e.g., tiny bends and wrinkles that serve as "landmarks" when trying to re-find a cell. The reason for this lies in the fact that shape-based grids overplot the lines between neighboring grid cells with the border lines of both adjacent squares. This makes it harder to use individual line features in shape-based grids, while such features appear clearly and without being overplotted in line-based grids. Thus, we can partially confirm H1 when users make conscious use of the sketchiness and uses a line-based sketchy grid.

4. Conclusions

Sketchiness has been used in the past mainly to communicate levels of uncertainty. In this paper, we looked at a different possibility: using sketchy rendering to create more distinctive grid drawings. Through a small lab study, we gained partial confirmation for our intuition that the sketchy visual features could provide additional cues for orientating in very homogeneous visualizations.

We can easily envision this idea being applied beyond the realm of matrices and treemaps. For example, bubble charts with many bubbles being globbed together could benefit from a similar sketchy treatment – in particular, as their irregular circle-packing layouts lend themselves even less to counting strategies.

References

[AWI*09] ALMERAJ Z., WYVILL B., ISENBERG T., GOOCH A. A., GUY R.: Automatically mimicking unique hand-drawn pencil lines. *Computers & Graphics 33*, 4 (2009), 496–508. doi:10.1016/j.cag.2009.04.004. 2

[Bat55] BATES G. E.: Joint distributions of time intervals for the occurrence of successive accidents in a generalized Polya scheme. *The Annals of Mathematical Statistics 26*, 4 (1955), 705–720. doi:10.1214/aoms/1177728429. 2

[BBIF12] BOUKHELIFA N., BEZERIANOS A., ISENBERG T., FEKETE J.: Evaluating sketchiness as a visual variable for the depiction of qualitative uncertainty. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2769–2778. doi:10.1109/TVCG.2012.220. 1

[BFW02] BASTIN L., FISHER P. F., WOOD J.: Visualizing uncertainty in multi-spectral remotely sensed imagery. *Computers & Geosciences 28*, 3 (2002), 337–350. doi:10.1016/S0098-3004(01)00051-6. 3

[CKB*19] CHEONG L., KINKELDEY C., BURFURD I., BLEISCH S., DUCKHAM M.: Evaluating the impact of visualization of risk upon emergency route-planning. *International Journal of Geographical Information Science* (2019), 1–29. to appear. doi:10.1080/13658816.2019.1701677. 1

[CR74] CATMULL E., ROM R.: A class of local interpolating splines. In *Computer Aided Geometric Design*, Barnhill R. E., Riesenfeld R. F., (Eds.). Academic Press, 1974, pp. 317–326. doi:10.1016/B978-0-12-079050-0.50020-5. 2

[EBRI11] EVERTS M. H., BEKKER H., ROERDINK J. B. T. M., ISENBERG T.: Illustrative line styles for flow visualization. In *Pacific Graphics'11: Short Paper Proceedings* (2011), Chen B.-Y., Kautz J., Lee T.-Y., Lin M. C., (Eds.), Eurographics, pp. 105–110. doi:10.2312/PE/PG/PG2011short/105-110. 1

[EDG*08] ELMQVIST N., DO T.-N., GOODELL H., HENRY N., FEKETE J.-D.: ZAME: Interactive large-scale graph visualization. In *PacificVis'08: Proceedings of the IEEE Pacific Visualization Symposium* (2008), Fujishiro I., Li H., Ma K.-L., (Eds.), IEEE, pp. 215–222. doi:10.1109/PACIFICVIS.2008.4475479. 1

[Elb95] ELBER G.: Line illustrations ∈ computer graphics. *The Visual Computer 11*, 6 (1995), 290–296. doi:10.1007/BF01898406. 1

[FS94] FINKELSTEIN A., SALESIN D. H.: Multiresolution curves. In *SIGGRAPH'94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (1994), Glassner A. S., Keeler M., (Eds.), ACM, pp. 261—268. doi:10.1145/192161.192223. 1

[GSS*19] GÖRTLER J., SPICKER M., SCHULZ C., WEISKOPF D., DEUSSEN O.: Stippling of 2D scalar fields. *IEEE Transactions on Visualization and Computer Graphics 25*, 6 (2019), 2193–2204. doi:10.1109/TVCG.2019.2903945. 1

[LFH*16] LIMBERGER D., FIEDLER C., HAHN S., TRAPP M., DÖLLNER J.: Evaluation of sketchiness as a visual variable for 2.5D treemaps. In *IV'16: Proceedings of the 20th International Conference Information Visualisation* (2016), Banissi E., Bannatyne M. W. M., Bouali F., Burkhard R., Counsell J., Cvek U., Eppler M. J., Grinstein G., Huang W. D., Kernbach S., Lin C.-C., Lin F., Marchese F. T., Pun C. M., Sarfraz M., Trutschl M., Ursyn A., Venturini G., Wyeld T. G., Zhang J. J., (Eds.), IEEE, pp. 183–189. doi:10.1109/IV.2016.61. 1

[Per85] PERLIN K.: An image synthesizer. *ACM SIGGRAPH Computer Graphics 19*, 3 (1985), 287–296. doi:10.1145/325165.325247. 3

[SS19] SONG H., SZAFIR D. A.: Where's my data? evaluating visualizations with missing data. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (2019), 914–924. doi:10.1109/TVCG.2018.2864914. 1

[SSSS98] SCHLECHTWEG S., SCHÖNWÄLDER B., SCHUMANN L., STROTHOTTE T.: Surfaces to lines: Rendering rich line drawings. In *WSCG'98: Proceedings of the 6th International Conference in Central Europe on Computer Graphics and Visualization* (1998), Skala V., (Ed.), pp. 354–361. URL: http://wscg.zcu.cz/wscg1998/papers98/schlechtwegwscg98.ps.gz. 1

[SSTR93] SARKAR M., SNIBBE S. S., TVERSKY O. J., REISS S. P.: Stretching the rubber sheet: A metaphor for viewing large layouts on small screens. In *UIST'93: Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology* (1993), Hudson S. E., Pausch R., Zanden B. T. V., Foley J. D., (Eds.), ACM, pp. 81–91. doi:10.1145/168642.168650. 3

[WD08] WOOD J., DYKES J.: Spatially ordered treemaps. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1348–1355. doi:10.1109/TVCG.2008.165. 1

[WII*12] WOOD J., ISENBERG P., ISENBERG T., DYKES J., BOUKHELIFA N., SLINGSBY A.: Sketchy rendering for information visualization. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2749–2758. doi:10.1109/TVCG.2012.262. 1, 2, 3