Supplementary Material

This document contains some more details of our work and larger version of the figures.

1 Motivating Examples

There are several motivating scenarios where contour stylization can be useful:

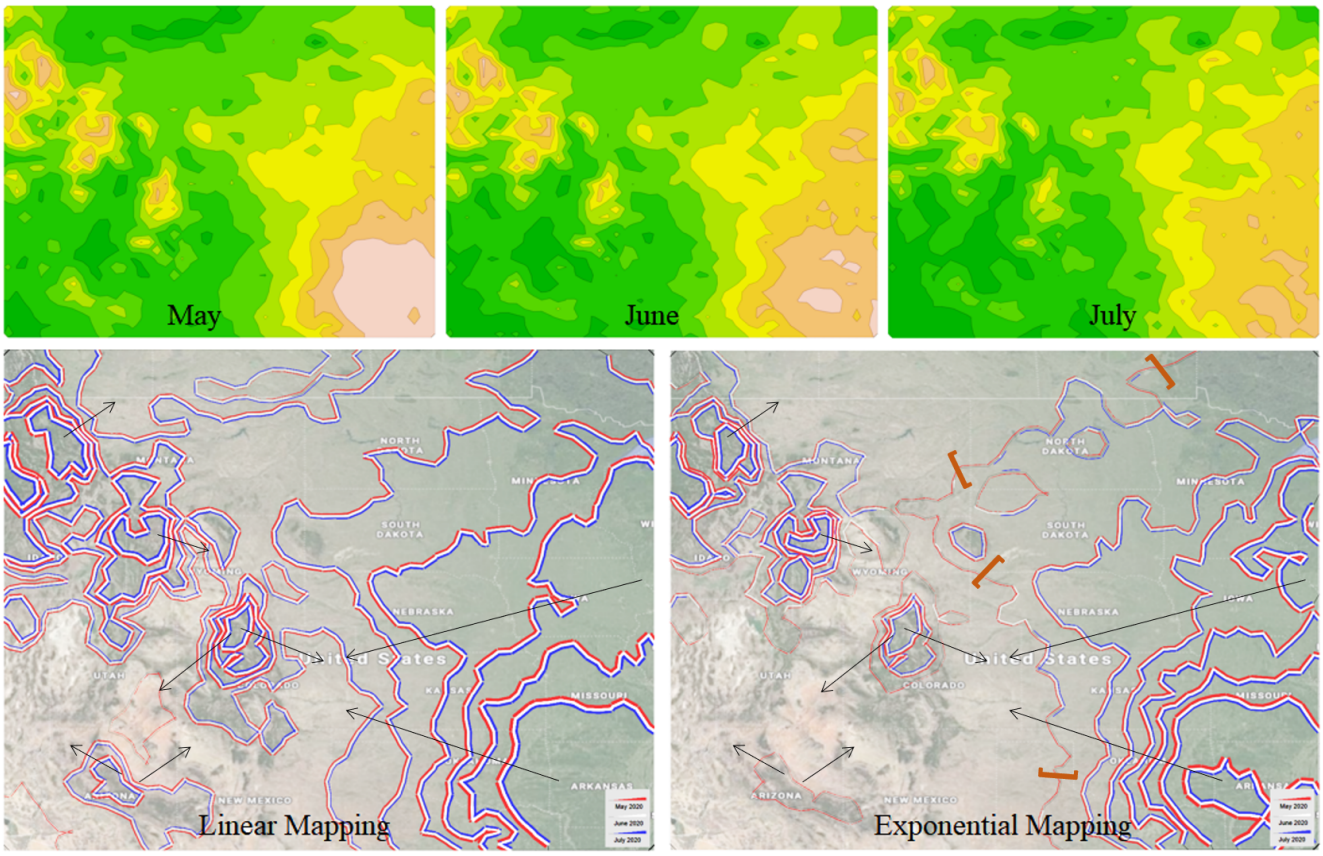


Figure 1: (top) Mean soil moisture plots for May, June and July, where green color denotes a low value. (bottom) The information for June and July is encoded using line width along May's isolines for normalized and exponential mappings. May, June and July are encoded using red, white and blue, respectively.

1.1 Example 1 (Examining Stability and Change Patterns)

Scientists often examine consistencies and trends in historical data using side-by-side contour plots. Figure 1(top) shows such an example where the soil moisture plots for May, June and July 2020 are shown side by side. Aggregating information from three plots requires additional mental effort, and in this scenario, stylized contour lines can readily give access to important insights. Figure 1(bottom) shows our contour stylization that encodes the May isolines (red) with June (white) and July (blue) soil moisture. The moisture values are mapped to line width.

As annotated using the arrows, one can easily see how the soil moisture value is decreasing gradually from along the arrow direction (i.e., when the contour lines intersect the tail of the arrows). One can also examine the change

along an isoline, as marked using the brown intervals. Some contour lines around Arizona contains only a single color (red), i.e., soil moisture value in those regions is higher in May but lower in June and July.

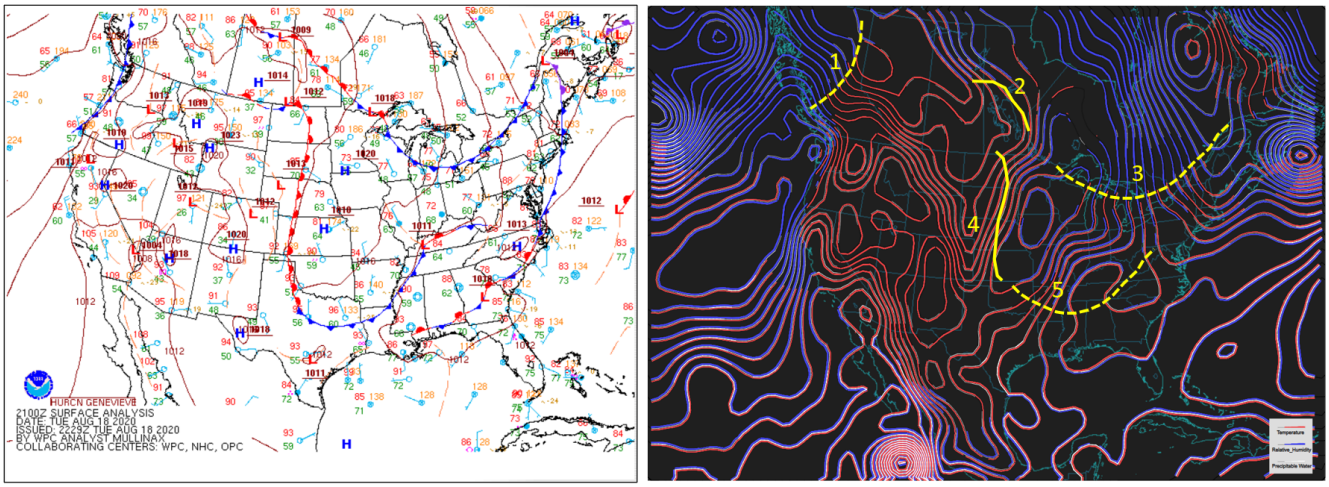


Figure 2: (left) Front detection by NOAA WPC. (right) detection using one of our five visualization techniques.

1.2 Example 2 (Meteorological Analysis)

A number of meteorological analyses requires examining complex interaction among multiple weather parameters. Here we consider weather-front analysis as an example. A front appears along a contact line of two air masses of different temperatures, where in most cases, the moisture levels are also different. The development of a front depends on several factors such as temperature, moisture, wind direction and pressure.

Finding weather fronts is important in weather analysis and prediction, as they cause various pivotal weather conditions such as rain, snow, or thunderstorms. Fronts change their position frequently, and require a continuous prediction of weather conditions due to the dynamic characteristics of weather variables. Figure 2 (left) shows front prediction by the National Oceanic and Atmospheric Administration (NOAA) Weather Prediction Center (WPC) archive, where the curved lines (red, blue or mixed) correspond to various types of fronts (warm, cold or stationary fronts, respectively). Note that such fronts can be derived using software or by painstaking inspection of the numbers plotted on the map representing various weather parameters.

Figure 2 (right) shows our contour stylization for 4 weather variables (pressure, temperature, relative humidity and precipitable water). Our design can readily reveal potential front positions, marked by the yellow curves. The contour lines represent isobars (pressure). The temperature, relative humidity and precipitable water are encoded in red, blue, and white lines respectively.

We discuss the cold fronts (yellow curves 1, 3, and 5) first. Each of these curves has lower temperature and higher relative humidity on one side, indicating a cold air mass. On the other side, there is high temperature with low relative humidity. The border revealed by our visualization is well-aligned with the cold front depicted in the NOAA visualization. Both these curves are located west of low pressure points (see the 'L' and 'H' notations in NOAA figure). Usually cold fronts bring active weather conditions such as rain or severe storms. The presence of precipitable water close to the yellow curves indicates the chance of rain. For warm fronts (curves 2 and 4), we have higher temperature and lower relative humidity on one side, and the opposite trend on the other side. There are more fronts in the NOAA images that are not clearly visible from our visualization. However, contour stylization (Figure 2 (right)) still depicts a much informative and clearer picture of the weather parameters than Figure 2 (left), which shows the multivariate information as a point scatterplot on the map.

Multivariate visualizations that encode data attributes into different preattentive perceptual features of a visual element (glyph) [1, 4, 5] such as size, shape, color, and texture, are typical ways to visualize geospatial information on a map. A well-known limitation of a glyph-based visualization is that it clutters the map [3]. While a dense overlay occludes the view of the base map (Figure 3 (left)), a sparse overlay compromises perception of geospatial connectedness and lacks the gradient information that naturally comes from a contour plot (Figure 3 (right)).

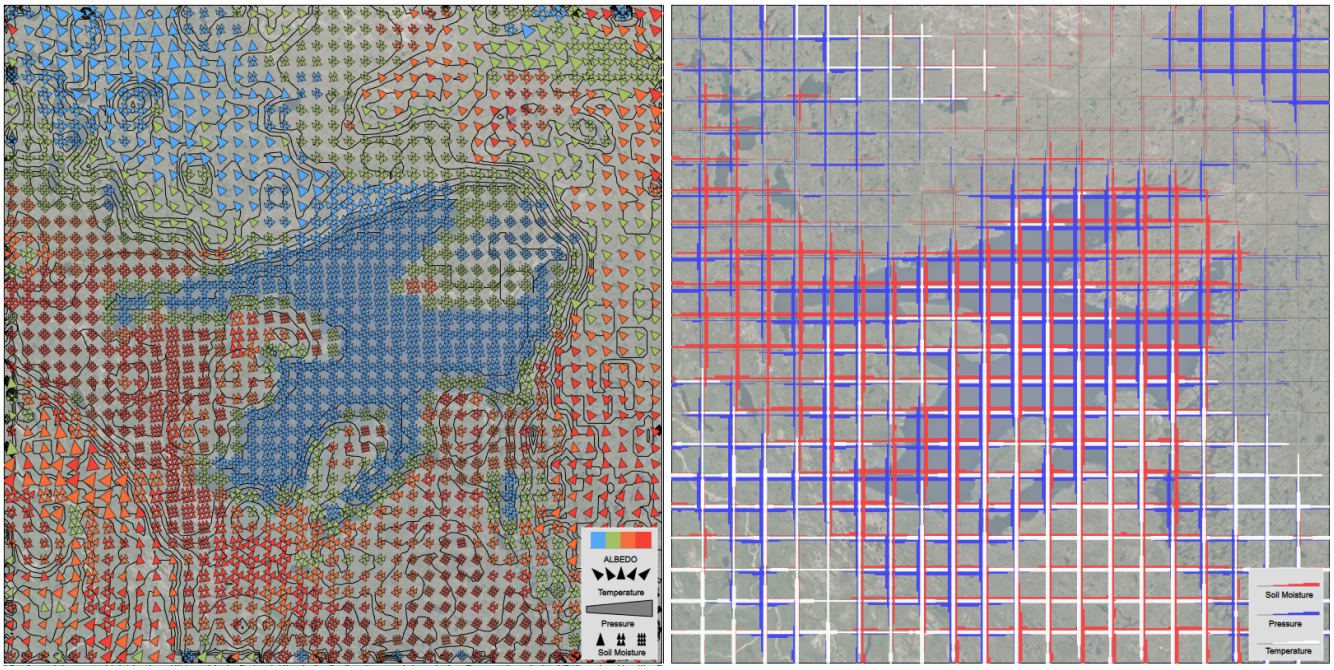


Figure 3: Multivariate visualization with (left) glyphs occludes the map, and (right) grid stylization lacks the gradient information.

2 Visual Encoding

In this section we describe five contour-based designs (Figures 4–5) for encoding geospatial information with four-attributes: A, B, C, and D. We assume that all the attributes are numeric and positive. We create a set of contour lines using the A attribute, and then encode the attributes B, C, and D along the contour lines of A using visual features.

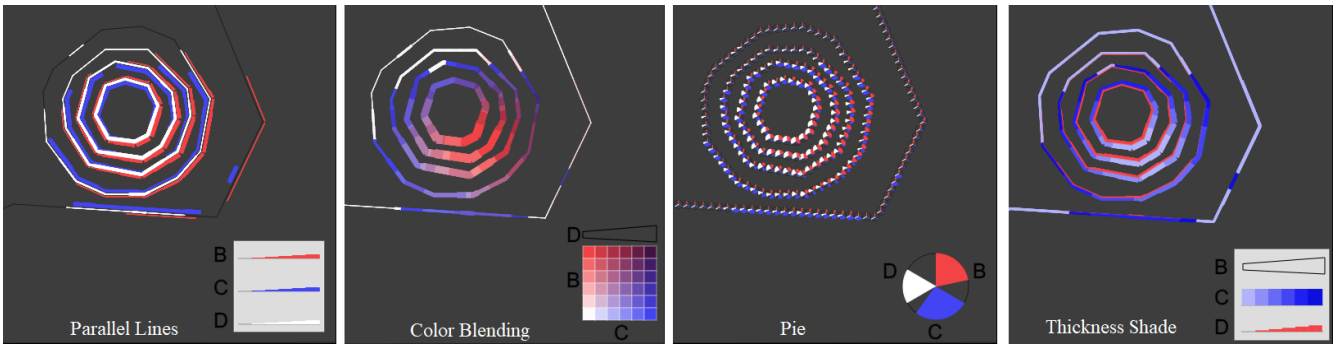


Figure 4: Encoding multivariate information using *Parallel Lines*, *Color Blending*, *Pie* and *Thickness-Shade*.

Design 1 (*Parallel Lines*): This design maps B, C, and D into three lines with distinct colors. The lines for B and C lie on opposite sides of the contour line of A, and the line for D follows the contour line of A. The data values are encoded using line width (between 0 and w), and the value of the attribute is linearly mapped to the range $[0, w]$. If the value of D is 0, then the base contour line of A becomes visible.

Design 2 (*Color Blending*): This design encodes B and C with distinct colors, and then blends them on the contour line of A. The attribute D is mapped to the width of the contour line. Note that since the contour line of A has a non-zero width u, the values of D are mapped to the linewidth range $[u, w]$. Consequently, B and C remain visible even when D is 0.

Design 3 (*Pie*): This design encodes B, C, and D using pie slices of distinct colors, and put them together to create a pie icon. The only difference from a pie chart is that the sum of the values of B, C, and D may not be equal to the total pie area. The pie icons are placed successively along the contour line of A. The pie slices for B, C, D start at 0°, 120° and 240° (assuming the top as 0°), and can grow clockwise to cover an angle of 120°. An attribute

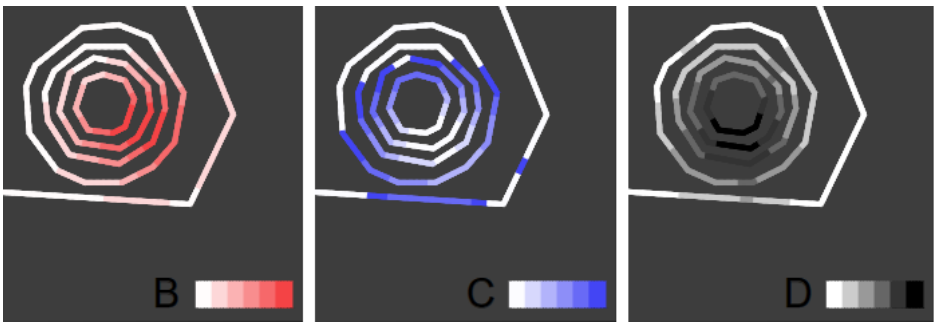


Figure 5: A side-by-side design.

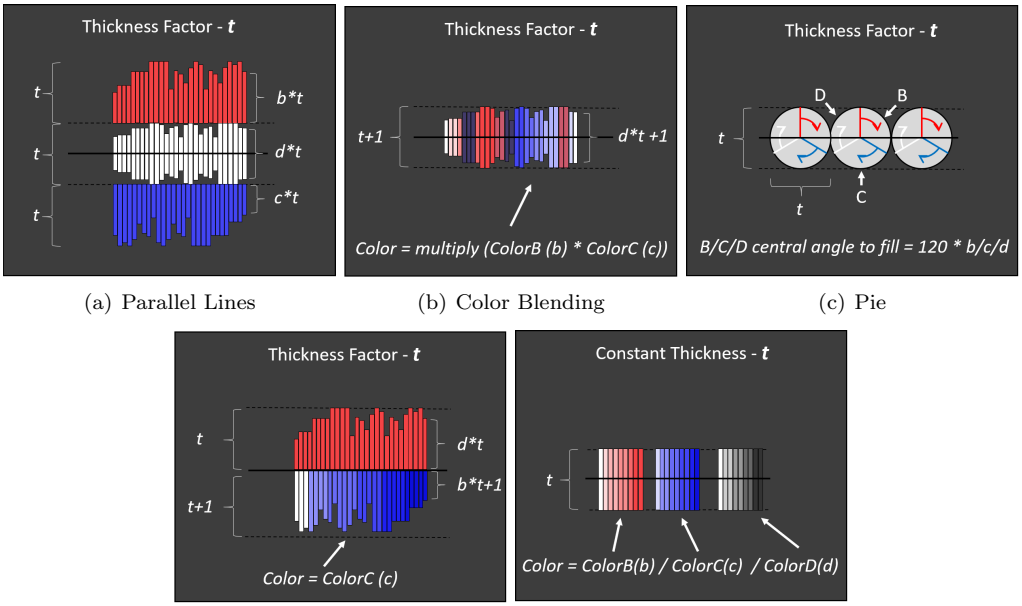
value is encoded into the angle covered by the corresponding pie slice.

Design 4 (*Thickness-Shade*): This design represents B and D using two distinct lines. The lines of B and D lie on opposite sides of the contour lines of A, with values encoded using line width. The values of C are encoded using a monochromatic color scheme, where the color appears on B's line. A low C value corresponds to a lighter shade, and a high value to a darker shade. The minimum line width for B is set to a positive threshold u, making a range of $[u, w]$ so that C remains visible even when B is 0.

Design 5 (*Side-by-Side*): This design shows B, C, and D in separate side-by-side views (Figure 5). Each of B, C, and D is encoded using a distinct monochromatic color scheme. The color appears on the contour lines of A. We ensured that the number of pixels used for *Side-by-Side* is comparable to those of the other designs, i.e., we choose the width and height of each of the *Side-by-Side* view to be $\lceil\sqrt{A/3}\rceil$, where A is the total pixel area of any other design, assuming a square display.

3 Implementation Details

The designs have been implemented using SVG (Scalable Vector Graphics) with the *D3.js* [2] web visualization library. For k contouring thresholds, we first computed the contour lines for A using the k-quantiles as the thresholds, and then further processed these polylines by dividing long line segments uniformly to create fine-grained polygonal chains. We then encoded the attributes by interpolating the values at the endpoints of these tiny segments. We now describe the details of the implementation. For convenience, let b, c and d denote the normalized B, C, and D values, respectively.



(a) Parallel Lines (b) Color Blending (c) Pie

(d) Thickness-Shade (e) Side-by-Side

Figure 6: Illustration for the implementation details for different designs.

Parallel Lines: At every segment on A's contour line, B, C, and D were drawn with perpendicular lines. As illustrated in Figure 6(a), d were drawn on both sides of A's contour line, and B and C were drawn on opposite sides of D. A thickness factor t was used to linearly map the attribute values to the input line-thickness range. If all the attributes are 0, then we only see the thin contour line of A.

Color Blending: Here the value of D determines the line width, where the colors of B and C are blended on D's line. As shown in Figure 6(b), at every segment, one perpendicular line is drawn which intersects A's contour line in the middle. The perpendicular line's height is $(dt + 1)$ pixels. The constant 1 ensures that the colors can be blended even when d is 0. For each of B and C, the colors were chosen using a discrete monochromatic color scheme with a perceptually uniform color distance. The number of discrete shades in the scale depends on the number of contour intervals. We blended the colors using the *mix-blend-mode* scheme *Multiply* of CSS, which was chosen in a pilot study with 63 participants on 3 possible candidate schemes: *Multiply*, *Darken* and *Difference*. The participants had to complete 24 value estimation tasks, 8 in each blend mode. *Multiply* had higher mean accuracy than the others.

Pie: In this design (Figure 6(c)), each attribute was allotted a 120° space to create a pie slice. The center of the pie is located on the contour line of A.

Thickness-Shade: This design visualizes B and D on opposite sides of the A's contour lines (Figure 6(d)). The D value is mapped to a perpendicular line of height dt. For B, the value is mapped to $bt + 1$, where a minimum of 1 pixel allows encoding C even if B is 0. The color for C is chosen using a discrete monochromatic color scheme with a perceptually uniform color distance.

Side-by-Side: In this design, the perpendicular lines have the same fixed height t and intersect A's contour lines in the middle (Figure 6(e)). The B, C, and D values were encoded in the corresponding discrete sequential color scheme.

4 Figures from the Study Sections

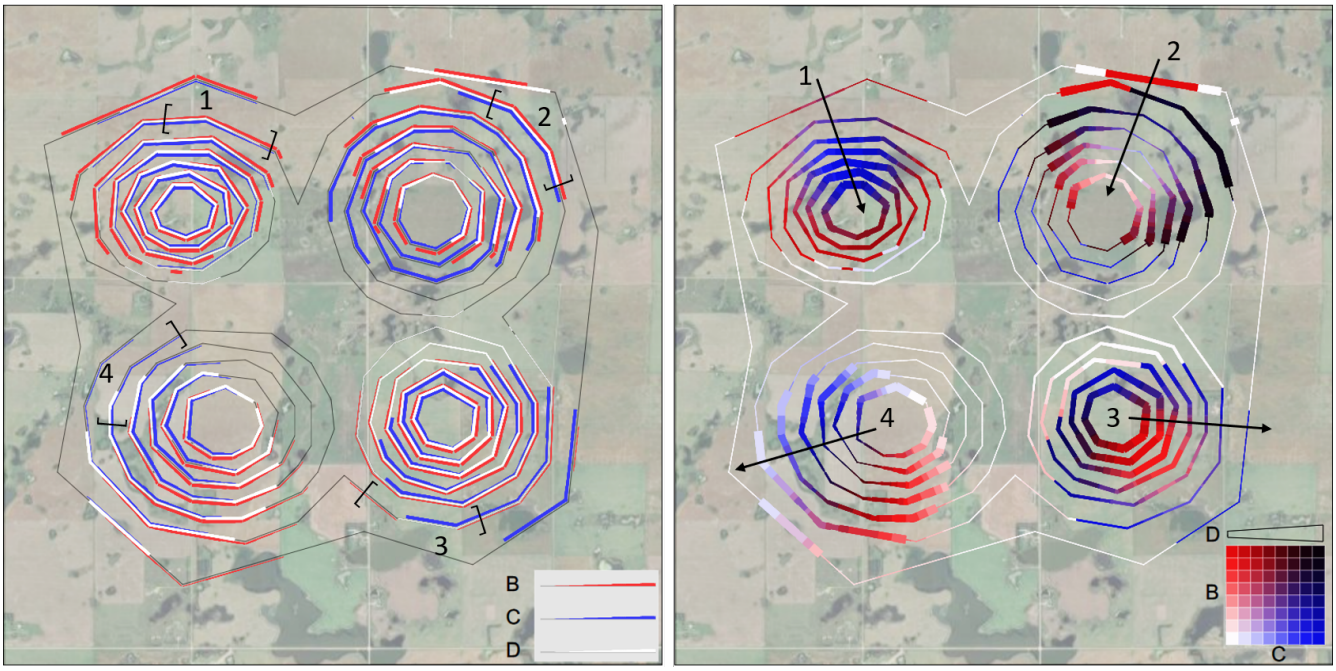


Figure 7: Illustration for tasks.

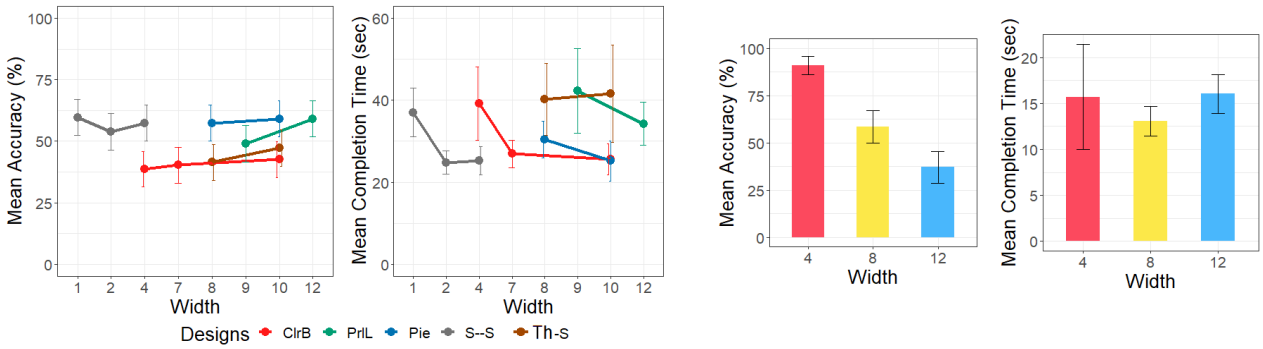


Figure 8: Study 1: (left and middle) Performances of the designs at different width choices for Tasks 1-4. Lines are connected to group the designs, but not to denote continuity of the width. (right) Accuracy and completion time for the icon-counting task.

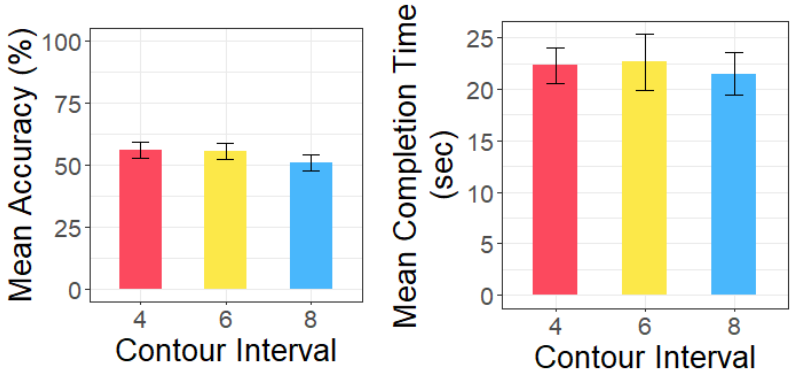


Figure 9: Study 2: Task performance with different contour intervals.

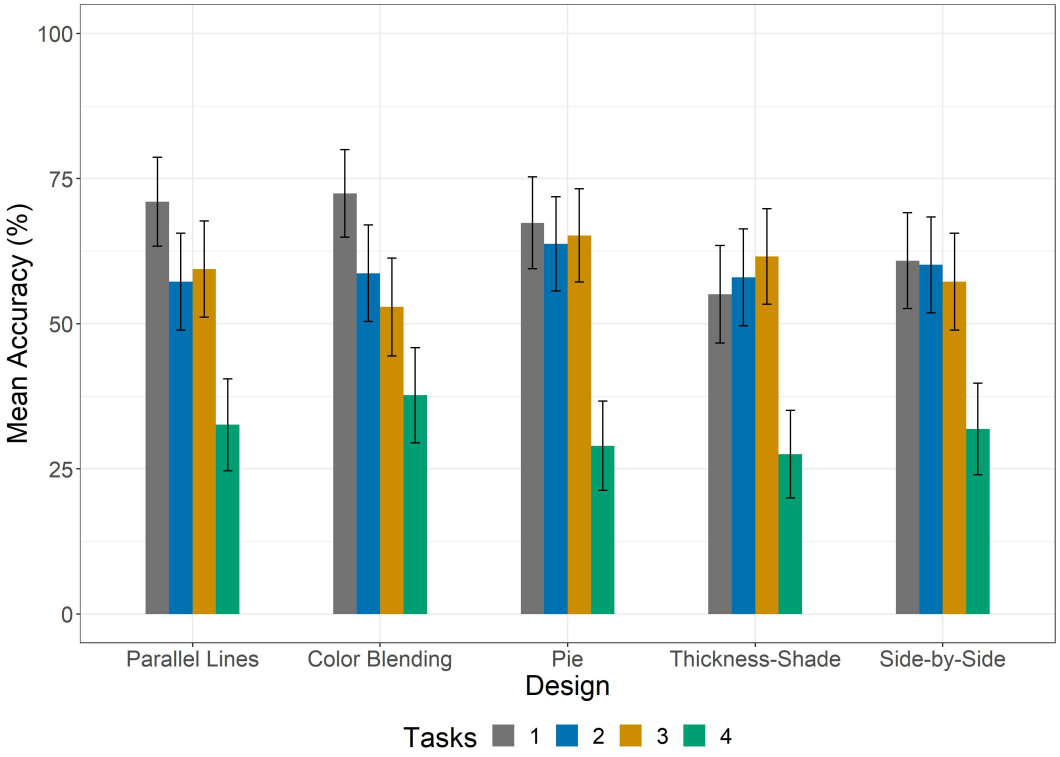


Figure 10: Study 2: Task accuracy for the five designs.

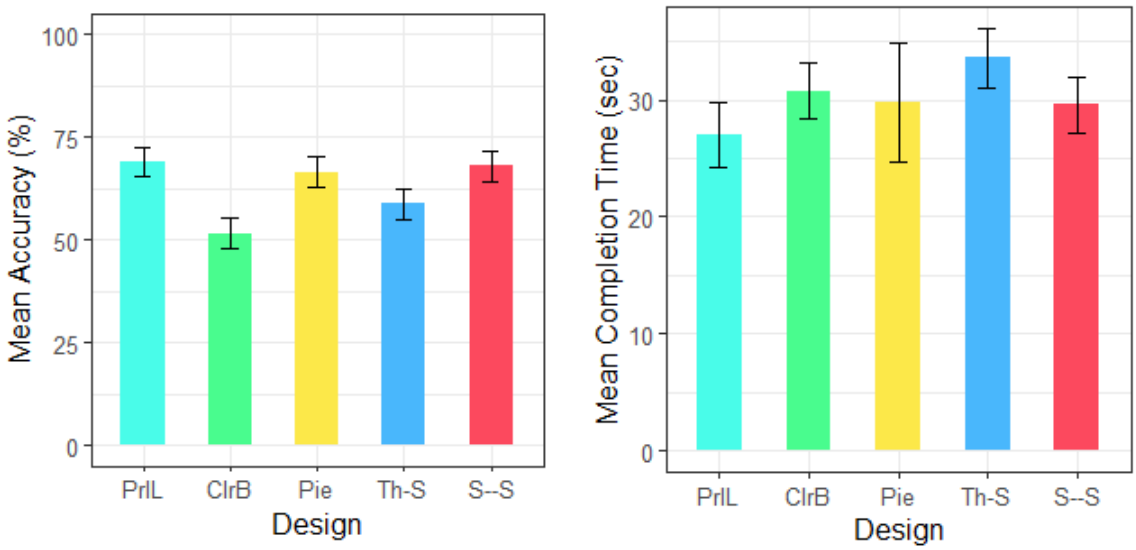


Figure 11: Study 3: Overall performance of the different designs.

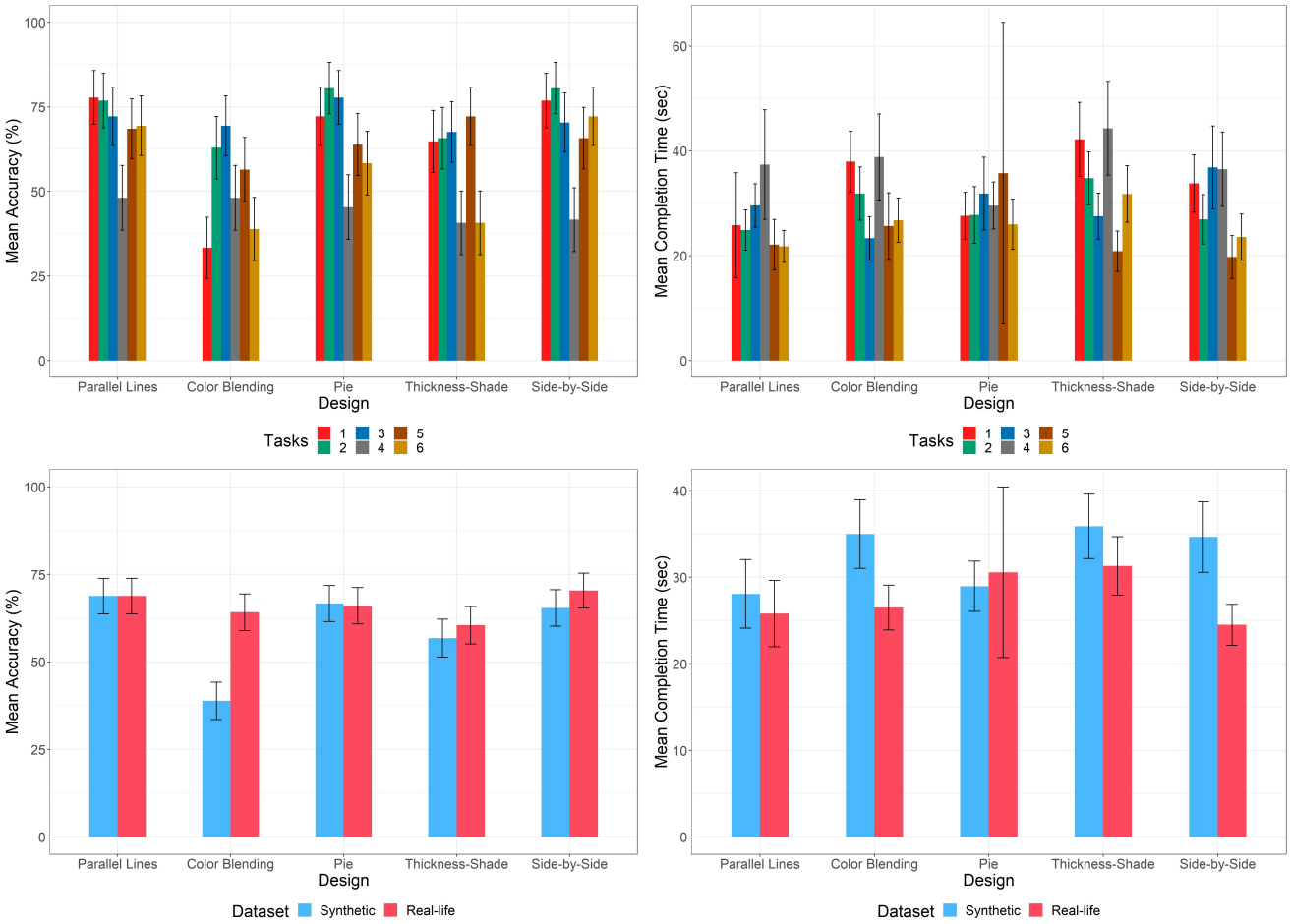


Figure 12: Task performances in Study3, for (top) different designs, and (bottom) different datasets.

5 Design Recommendations

Based on the study results, we formulated a table of design recommendations (Table 1) that summarizes the preferred design choices for various tasks over three environments – general use, time-sensitive interpretation, and high accuracy requirements. The table shows some strengths for particular designs that are different from the overall discussion above: for example, if the task requires quick estimation for trends along a contour line, then *Color Blending* and *Thickness-Shade* may be the best design options. Any comparison with '[*]' means that it is statistically significant.

Table 1: Design recommendation table

	Environment		
Domain	Time Sensitive	Accuracy Sensitive	General
Compare different contour parts	***Parallel Lines*** (mean-23.8 sec; 47%, 64%[*], and 31% faster than *Color Blending*, *Thickness-Shade* [*], and *Side-by-Side* resp.), ***Pie*** (mean-27.6 sec; 38%, 53%[*], and 22% faster than *Color Blending*, *Thickness-Shade* [*], and *Side-by-Side* resp.)	**All except *Color Blending*** (mean-33.3%; 57%[*], 54%[*], 49%[*] and 57%[*] less accurate than *Parallel Lines* [*], *Pie* [*],*Thickness-Shade* [*], and *Side-by-Side* [*] resp.)	All except *Color Blending*
Search for a trend across contour lines	***Parallel Lines*** (mean-24.9 sec; 28%, and 40%[*] faster than *Color Blending* and *Thickness-Shade* [*] resp.),***Pie*** (mean-27.8 sec; 15%, and 25% faster than *Color Blending* and *Thickness-Shade* resp.), ***Side-by-Side*** (mean-24.9 sec; 18%, and 28% faster than *Color Blending*, and *Thickness-Shade* resp.)	***Parallel Lines*** (mean-76.9%; 22% and 17% more accurate than *Color Blending* and *Thickness-Shade* resp.), ***Pie***(mean-80.6%; 28%[*] and 23%[*] more accurate than *Color Blending* [*] and *Thickness-Shade* [*] resp.), ***Side-by-Side***(mean-80.6%; 28%[*] and 23%[*] more accurate than *Color Blending* [*] and *Thickness-Shade* [*] resp.)	*Parallel Lines*,*Side-by-Side*,*Pie*
Search for a trend along a contour line	***Color Blending*** (mean-23.4 sec; 26%, 36%, and 81%[*] faster than *Parallel Lines*, *Pie* and *Side-by-Side* [*] resp.), ***Thickness-Shade*** (mean-27.6 sec; 7%, 13%, and 35% faster than *Parallel Lines*, *Pie* and *Side-by-Side* resp.)	All	All
Identify rate of change of a variable along a contour line	**All except *Pie*** (mean-35.8 sec; 62%, 39%, 71% and 84% slower than *Parallel Lines*, *Color Blending*, *Thickness-Shade*, and *Side-by-Side* resp.)	All	All
Identify the value difference on a contour part	***Parallel Lines*** (mean-21.8 sec; 23%, 19% and 46%[*] faster than *Color Blending*, *Pie* and *Thickness-Shade* [*] resp.), ***Side-by-Side*** (mean-23.6 sec; 12%, 10% and 35% faster than *Color Blending*, *Pie* and *Thickness-Shade* resp.)	***Parallel Lines***(mean-69.4%; 78%[*], 19% and 71%[*] more accurate than *Color Blending* [*], *Pie* and *Thickness-Shade* [*] resp.), ***Side-by-Side*** (mean-72.2% ; 86%[*], 24% and 77%[*] more accurate than *Color Blending* [*], *Pie* and *Thickness-Shade* [*] resp.),	*Parallel Lines*,*Side-by-Side*,*Pie*

References

[1] R. Borgo, J. Kehrer, D. H. Chung, E. Maguire, R. S. Laramee, H. Hauser, M. Ward, and M. Chen. Glyph-based visualization: Foundations, design guidelines, techniques and applications. In *Eurographics (STARs)*, pp. 39–63, 2013.

[2] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

[3] D. H. S. Chung. *High-dimensional glyph-based visualization and interactive techniques*. PhD thesis, Swansea University, UK, 2014.

[4] P. Shanbhag, P. Rheingans, et al. Temporal visualization of planning polygons for efficient partitioning of geo-spatial data. In *IEEE Symposium on Information Visualization (INFOVIS)*, pp. 211–218. IEEE, 2005.

[5] M. O. Ward. Multivariate data glyphs: Principles and practice. In *Handbook of data visualization*, pp. 179–198. Springer, 2008.