EUROGRAPHICS 2017 / J. Heer, T. Ropinski and J. van Wijk
(Guest Editors)

*Volume 36 (2017), Number 3*

# Generating Tile Maps

Graham McNeill[1] and Scott A. Hale[1,2]

[1]Oxford Internet Institute, University of Oxford, UK
[2]Alan Turing Institute, UK

**Abstract**

*Tile maps are an important tool in thematic cartography with distinct qualities (and limitations) that distinguish them from better-known techniques such as choropleths, cartograms and symbol maps. Specifically, tile maps display geographic regions as a grid of identical tiles so large regions do not dominate the viewer's attention and small regions are easily seen. Furthermore, complex data such as time series can be shown on each tile in a consistent format, and the grid layout facilitates comparisons across tiles. Whilst a small number of handcrafted tile maps have become popular, the time-consuming process of creating new tile maps limits their wider use. To address this issue, we present an algorithm that generates a tile map of the specified type (*e.g. *square, hexagon, triangle) from raw shape data. Since the 'best' tile map depends on the specific geography visualized and the task to be performed, the algorithm generates and ranks multiple tile maps and allows the user to choose the most appropriate. The approach is demonstrated on a range of examples using a prototype browser-based application.*

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

## 1. Introduction

*Tile maps* (also known as *grid maps* and *tile grid maps*) such as those in Figure 1 are used to display data about geographic regions. Each region is represented by a tile of the same shape and size. The tiles are arranged so as to retain local relationships and also, so that the overall shape of the tile map resembles that of the original map. These constraints help to preserve the viewer's mental model, allowing them to locate regions quickly. Tiles are typically square, but as shown in Figure 1d, other shapes can be used. The uniform size and shape of the tiles prevents large regions from dominating and makes it easy to find small regions. Furthermore, complex data can be shown on each tile in a consistent format, allowing comparisons across tiles. Square and rectangular tile maps are particularly well-suited to this task (*e.g.* [Ric16]) and the rigid grid layout of tile maps further aids comparison across tiles. From an aesthetic perspective, the clean, simple appearance of tile maps is notable. Of course, tile maps have their limitations. In particular, they preserve neither region shape nor region size, and hence, should not be used when these features are directly relevant to the analysis.

Tile maps are growing in popularity, but remain much less common than related approaches. Choropleth maps—where the regions of the map are colored to represent a thematic variable—are very popular and retain the geometric properties of the original map. However, they suffer from the problem that large regions dominate and small regions are hard to find/see [SMKH09].

Proportional and graduated symbol maps are another common choice for displaying geographic data. For data concerning the region as a whole, a symbol can be placed on the region's centroid with the size (area) of the symbol representing the thematic variable. Symbol maps are easy to construct, but are susceptible to symbol overlap and congestion. In addition, viewers often misjudge the relative areas of the symbols often underestimating the size of larger symbols [Fla71, Cra73, tC80]. It is difficult to correct for this perception error as there is variation amongst and within subjects [Gri85].

Cartograms are a final common choice. For these, each region is transformed so that its area is equal to the variable of interest. The transformed regions can be colored to display a second variable or simply to allow the audience to distinguish between regions. Traditional contiguous cartogram algorithms preserve neighborhood relationships and attempt to preserve region shape, but cartograms are most effective when the audience has good prior knowledge of the original geography; yet even so the shapes of the regions may make them difficult to identify [SMKH09, p. 402]. Alternative cartogram approaches use the same shape for each region—typically circles (Dorling cartograms [Dor96]) or rectangles (Demers cartograms [BDC02]), but these do not preserve the overall shape of the entire map and suffer from the area perception problems of proportional symbol maps.

Assessing the demand and suitability of different techniques is beyond the scope of this paper. However, given the distinctive benefits of tile maps we suggest that their relative scarcity is *not* due to a lack of demand, but rather, the effort required to

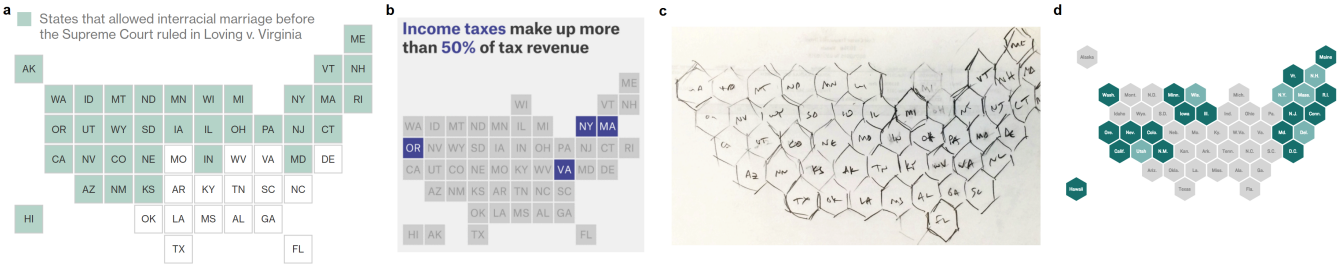


**Figure 1:** *Tile maps: (a) Bloomberg.com [TC15], (b) FiveThirtyEight.com [CM15], (c,d) NPR [DeB15].*

create them. Indeed, the growing popularity of tile maps can be largely attributed to the reuse of existing, handcrafted tile maps (Figure 1c) rather than the creation of new ones. There are many interesting blog posts that discuss tile maps and their construction [Sha16, Ric16, Won16a, DeB15, Zac15], but relatively little research on automating this procedure. In contrast, researchers have been investigating cartograms for decades and the best-known algorithms are available as plugins for popular GIS software [Wul] enabling their widespread use. Furthermore, choropleths and symbol maps require no layout algorithm and hence, are easily created programmatically or with GIS software. In this paper, we seek to level the playing field by presenting an algorithm that generates tile maps from raw shape data and a prototype browser-based application that implements the algorithm.

It is worth considering if a dedicated tile map algorithm is required. One possible approach is to use an existing algorithm to get close to a tile map and then complete the process manually [Won16a]. This semi-automated approach seems reasonable since it is likely that the 'optimal' tile map found by an algorithm may be suboptimal in the user's opinion and hence, require some tweaking. For example, the algorithm may fail to sufficiently preserve the shape of a perceptually salient peninsula or some adjacency relations that are important for the given context or task to be performed. Unfortunately, the intuitive notion that the automated part of the procedure will take us 'almost to the optimal solution' is overly optimistic due to the interdependent nature of the tiles: moving one tile affects both the global shape and the local relationships. This necessitates changes to neighboring tiles, which in turn requires changes to their neighbors. This chain reaction is frustrating for maps with a small number of tiles and becomes increasingly hard to handle as the number of regions increases.

## 2. Related Work

Eppstein *et al.* [EvKSS15] considered the general problem of finding a one-to-one assignment of region centroids to a given point set. For the specific case of tile maps, their experiments focused on examples where the prespecified point set is rectangular or almost rectangular. Eppstein *et al.* discuss three intuitive optimization criteria: preserving location, adjacency and relative orientation. For computational efficiency, they chose to optimize with respect to location or relative orientation (using a simplified, quadrant-based approach); all three criteria were then used as metrics to assess the suitability of the produced tile map. Both the location and orientation-based approaches outperformed an adapted spatial treemaps algorithm [WD08]. Their experiments suggested that preserving location by minimizing the sum of squared distances between region centroids and the corresponding tile centroids (allowing for a global translation) was the optimal approach. Meulemans *et al.* [MDS*17] extend the work of Eppstein *et al.* by considering a wider range of optimization criteria/metrics. Their heuristic optimization algorithm allows them to optimize with respect to any (or multiple) criteria—though local minima can be a problem. The resulting tile map can be scored on all criteria enabling a detailed analysis of how the different criteria interact.

The different approaches for handling empty grid cells is noteworthy. Eppstein *et al.* have various ways of handling empty cells, but note that *"none of these ways is obviously the right one, and each option to deal with empty cells gives different results, making the choice rather arbitrary."*. In contrast, the heuristic algorithm of Meulemans *et al.* selects which grid cells to use as part of the optimization. However, their focus is on understanding the interdependence of the optimization criteria and the number of empty grid cells; it is not clear what tile maps produced by their approach look like. Another potential solution to the empty grid cells problem is to discard cells whose centroids are furthest from any of the region centroids [WDS10]. This approach is reasonable when the region density of the original map is approximately uniform. When this is not the case, some deformation in global shape will be required to preserve local properties such as region adjacency. Hence, fixing the global shape of the tile map (by discarding grid cells) prior to considering the region-to-tile assignments is overly simplistic.

## 3. Rationale and Overview

Combining ideas from previous work on tile maps [EvKSS15, MDS*17] and cartograms [AKV*15, KN16], we suggest that a tile map should preserve the following properties of the original map:

***Local Relationships*** Neighbor relationships should be preserved—Egypt should be adjacent to and east of Libya in a tile map of Africa.

***Global Position*** Regions should be in the correct position relative to the global shape—California should be in the southwest of a USA tile map.

***Global Shape*** The boundary of the tile map should be similar to the boundary of the original map.

Existing approaches focus on finding the 'best' tile map for a

given geography with respect to criteria such as those listed above. However, there are important limitations to this approach. Firstly, it is unclear whether the importance of each criterion is fixed or depends on the geography. Secondly, physical geography is not the only factor that determines the perceptual salience of individual geographic features and relationships. Task-specific issues as well as the end users' cultural and demographic knowledge of the area also contribute to the significance of (and hence, importance of preserving) specific features of the original map. Naturally, case-specific considerations such as these cannot be incorporated into a fully-automated algorithm that reliably produces the 'best' tile map in any context. With this in mind, we create and rank many candidate tile maps (guided by the criteria listed above), then allow the designer to browse the ranked list to choose the most appropriate map for the given task. To help the designer choose from the candidates, we display the *total cost* associated with each tile on each map. Furthermore, the designer can choose to view the cost for a specific criteria only (*e.g.* adjacency) or adjust the weights assigned to different criteria and rerank the maps. Note that this semi-automated approach does not suffer from the chain reaction problem discussed earlier. Here, the automatic step *does* do all the hard work and the user need only browse the candidate tile maps. This approach fits with the conclusions and further work discussions made by Meulemans *et al.* [MDS*17]. In particular, they note that *"algorithms for computing layouts may need to take multiple metrics into account, weighing them depending on the spatial aspects of the input data."* and also that case-specific considerations challenge *"the assumption of a single-map solution: solutions may need to be developed or algorithms be applied on a case-by-case basis..."*.

Our algorithm for generating a single tile map can be summarized as follows:

1. Add Gaussian noise to region centroids.
2. Transform centroids so that (ideally) neighbors are equidistant (more gridlike), but the relative orientation of neighbors is preserved.
3. Transform boundary polygon (global shape) to reflect changes in centroids—think of the centroids near the boundary dragging the boundary with them.
4. Fit tiles (any shape) inside transformed boundary polygon—same number of tiles as regions.
5. Assign transformed centroids to tile centroids so as to minimize the sum of squared distances.

To generate multiple tile maps, we split each existing candidate into multiple candidates at each of stages 1–4 of the above procedure. Once created, the candidates are ranked based on how well they preserve location, adjacency and relative orientation as well as their boundary roughness. The prototype browser-based application presented in Section 4.6 allows users to select the number of candidates created at each step and also, to rerank candidates by adjusting the importance attached to each criterion.

Before describing the algorithm in detail, we consider how different map properties are preserved and assessed. In terms of *Local Relationships*, we optimize with respect to location (shown to be effective in [EvKSS15, MDS*17]) and use relative orientation and adjacency (as well as location) as ranking costs. Since boundary points are dragged along by nearby centroids, we assume that *Global Position* is roughly preserved and do not assess this quantitatively. Since tiles are fitted to the transformed boundary polygon, the extent to which *Global Shape* is preserved depends on the shape similarity of the original and transformed boundary. The change in *Global Shape* can be limited (or prevented) by limiting the extent to which the centroids and hence the boundary is transformed (Section 4.1). *Global Shape* is not assessed quantitatively. Whilst this is an appealing idea, it is problematic since the boundary of the tile map *should* be significantly different to the original boundary in many cases. For example, we intuitively expect a tile map of the USA to expand the north-east of the country to make room for the densely packed states in that area (Figures 1-5). Despite the obvious differences between the boundary of the USA and the boundary of a ('good') USA tile map, they are still perceived as similar—people with a reasonable knowledge of US geography will be able use their mental model when analyzing the tile map. Simple shape similarity measures such as the Fréchet, Hausdorff and Procrustes distances [VH01,DM98,MV05] are not equipped to assess this type of shape similarity. Whilst more sophisticated measures (*e.g.* [MV06, FS07]) are possibly more appropriate, they are still not able to capture any task-specific and demographic issues which affect the relative importance of shape features. In contrast, assessing global shape similarity (with or without context) is a simple task for a human designer visually scanning candidate maps.

## 4. The Algorithm

We focus on problems where the set of regions is contiguous. More complex problems naturally decompose into multiple problems of this type; the associated tile maps can easily be combined manually—*e.g.* positioning Europe above Africa.

The input to the algorithm is a polygon (or multiple polygons) for each region. From these, we compute the region centroids, the adjacency matrix and the global boundary polygon. The algorithm then proceeds as described in the following sections.

### 4.1. Transforming the Centroids

We start by adding Gaussian noise to the centroids. The standard deviation of the noise is specified as a proportion of the mean distance of a centroid from its neighbors. In other words, the noise is proportional to the local centroid density. Even a small amount of centroid noise can have a significant impact on the shape (Figure 2) and the region-to-tile assignments of the associated tile maps. Adding unique noise to the original centroids allows us to create as many candidate sets of noisy centroids as desired.

The second type of shape variation is more substantive: the centroids are transformed so that (ideally) neighbors are equidistant (more gridlike), but the relative orientation of neighbors is preserved. To accomplish this, we update the position of each centroid $\mathbf{c}_i$ based on the current position of its neighbors using:

$$\mathbf{c}_i^{new} = \frac{1}{|N(i)|} \sum_{j \in N(i)} (\mathbf{c}_j^{old} + s\hat{\mathbf{u}}_{ji}) \tag{1}$$

where $N(i)$ is the set of neighbors of region $i$, $s$ is the estimated grid step size (Eq.(5)) and $\hat{\mathbf{u}}_{ji}$ is the unit displacement vector between

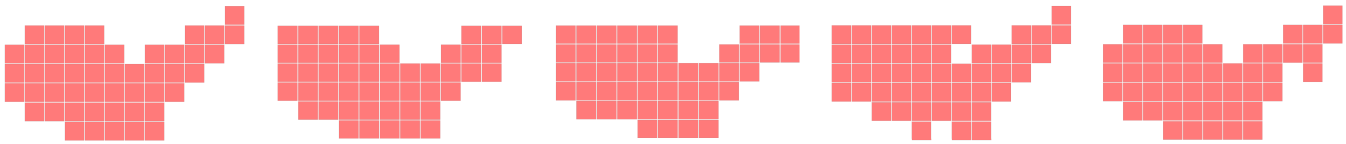

**Figure 2:** *Unlabeled tile maps for US states. The first map is based on the original centroids; other maps are based on noisy centroids. Noise standard deviation: 0.1; other parameters fixed (fully transformed centroids, boundaries smoothed–30 basis functions, no grid shift).*

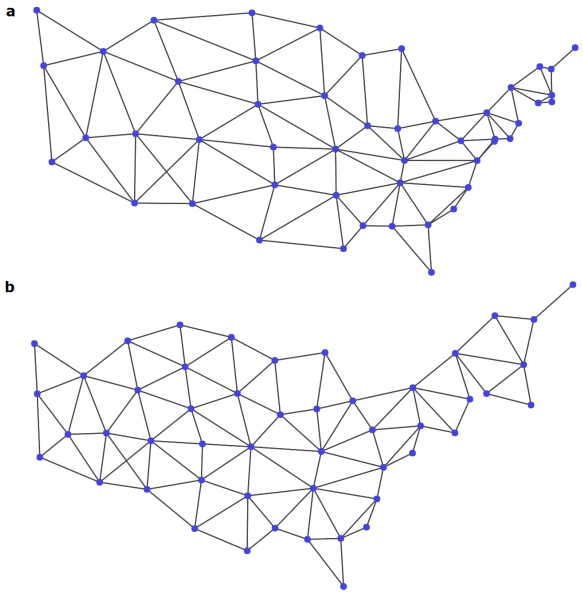


**Figure 3:** *States of the USA. (a) Original centroids. (b) 'Fully transformed' centroids.*

the original/noisy centroids $\mathbf{c}_j$ and $\mathbf{c}_i$:

$$\hat{\mathbf{u}}_{ji} = \frac{\mathbf{c}_i - \mathbf{c}_j}{\|\mathbf{c}_i - \mathbf{c}_j\|}.$$

The centroids are iteratively refined using Equation (1) until convergence. We use 30 iterations in all experiments—this is rather generous and the additional (albeit negligible) computation could be avoided by monitoring for convergence. Figure 3 shows the original centroids and the transformed centroids for the states of the USA. Note that the distances between neighbors (*i.e.* the lengths of the lines) are more uniform after the centroids have been transformed, but also that the relative orientation of neighbors appears to be well-preserved.

For a given set of region centroids (possibly with noise added), we create multiple sets of candidate centroids by interpolating between the untransformed and the 'fully transformed' centroids. If centroids remain close to their original positions, the intuition is that the resulting tile map will preserve *Global Shape*, but have poor *Global Position* and *Local Relationships* (Sections 3 and 5).

### 4.2. Transforming the Boundary

Given a set of transformed centroids $\{\mathbf{c}_j^{new}\}$, we update the boundary using a similar approach:

$$\mathbf{b}_i^{new} = \left( \sum_{j \in M(i,k)} w_j(i,k) \mathbf{c}_j^{new} \right) + \sqrt{\frac{s}{\|\mathbf{v}_i(k)\|}}\, \mathbf{v}_i(k) \qquad (2)$$

where $M(i,k)$ is the set of indices of the $k$ nearest original/noisy centroids to the original boundary point $\mathbf{b}_i$, $w_j(i,k)$ are weights on the centroids in $M(i,k)$ (see below) and $\mathbf{v}_i(k)$ is the weighted mean of the displacement vectors from the centroids in $M(i,k)$ to $\mathbf{b}_i$:

$$\mathbf{v}_i(k) = \sum_{j \in M(i,k)} w_j(i,k)(\mathbf{b}_i - \mathbf{c}_j). \qquad (3)$$

In the above equation, $\mathbf{c}_j$ is the pre-transformation $\mathbf{c}_j^{new}$. The square root term in Equation (2) pulls the boundary towards nearby centroids if it is greater than $s$ from them, and pushes the boundary away from nearby centroids if it is less than $s$ from them. Equation (2) need only be evaluated once for each boundary point—the values that it depends on are known and fixed. In all experiments, we use $k = 3$ nearest centroids.

The weights $w_j(i,k)$ used in Equations (2) and (3) ensure that centroids nearer the boundary point have a greater impact on the new position of the boundary point. They are computed using:

$$w_j(i,k) \propto \exp\left( -\frac{\|\mathbf{b}_i - \mathbf{c}_j\|^2}{2 \min_{j \in M(i,k)} \|\mathbf{b}_i - \mathbf{c}_j\|^2} \right) \qquad (4)$$

then normalized to sum to 1:

$$\sum_{j \in M(i,k)} w_j(i,k) = 1.$$

We smooth the transformed boundary using standard basis-function regression with periodic basis functions:

$$f(x; \mu_i, \sigma) = \exp\left( -\frac{D(x-\mu_i)^2}{2\sigma^2} \right)$$

where $D(x,\mu_i)$ is the minimum arc length along the boundary between $x$ and the mean of the $i$-th basis function $\mu_i$. The $\mu_i$ are equally spaced around the boundary polygon and the spread parameter $\sigma$ is set equal to the length of the boundary divided by the number of basis functions. The form of the basis functions is not critical and other periodic functions such as wrapped normals, von Mises distributions or trigonometric functions could be used.

For a given boundary, we create multiple versions of it by varying the amount of smoothing, *i.e.* the number of basis functions. Examples of transformed boundaries are shown in Figures 4 and 6.

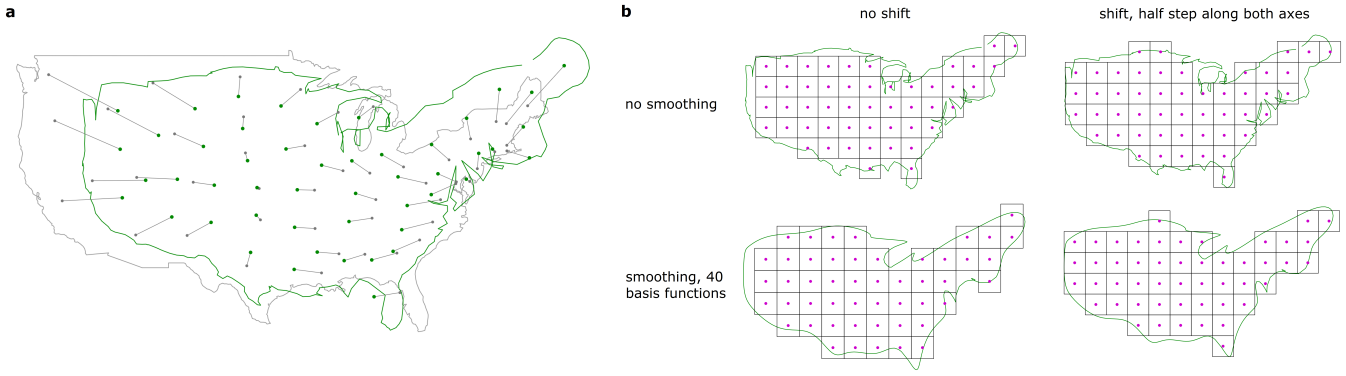


**Figure 4:** *States of the USA. (a) Points are state centroids: gray–original (no noise), green–fully transformed. Lines: gray–original boundary, green–transformed boundary. (b) Creating variation by smoothing the boundary and shifting the grid origin.*

### 4.3. Fitting Tiles to the Boundary

Given $R$ centroids and an associated boundary polygon of area $A$, we wish to find a set of $R$ tiles whose centroids fit inside the boundary. To achieve this, we lay a grid with step size:

$$s = \sqrt{\frac{A}{R}} \tag{5}$$

over the boundary polygon and iteratively resize the distance between grid points (tile centroids) until exactly $R$ of them lie inside the boundary.

For a given boundary polygon, we create multiple candidate sets of tiles by shifting the grid's origin (horizontally and/or vertically) prior to fitting the grid points to the boundary (Figure 4b).

### 4.4. Assigning Regions to Tiles

Given a set of $R$ transformed region centroids $\{\mathbf{c}_i^{new}\}$ and a set of $R$ tiles, we seek a 1-to-1 assignment that minimizes changes in centroid position. Specifically, we seek a permutation $\phi$ of $\{1,2,\ldots,R\}$ that minimizes the cost function:

$$f(\phi) = \sum_{i=1}^{R} \|\mathbf{c}_i^{new} - \mathbf{t}_{\phi(i)}\|^2 \tag{6}$$

where $\mathbf{t}_i$ is the centroid of tile $i$. We solve the assignment problem (*i.e.* find the optimal $\phi$) using the Hungarian method (the Kuhn-Munkres algorithm). [Mun57].

### 4.5. Ranking Tile Maps

Given any map, we can now generate candidate tile maps using the approach described in Sections 4.1-4.4. The final step is to rank the candidate maps. This is done by evaluating a cost function that measures the extent to which *Local Relationships* are preserved as well as penalizing boundary roughness:

**Location cost** The distance between the region centroids and the tile centroids should be as small as possible. However, some displacement is inevitable (Section 3) and the transformation described in Section 4.1 is designed to move centroids in an acceptable way. Hence, we do not penalize displacement due to the transformation. The location cost is the mean distance between the transformed region centroids and the tile centroids to which they are assigned; this is divided by the approximate grid step $s$ (Equation (5)) so that the cost is measured in grid steps.

**Adjacency cost** The mean fraction of missing neighbors for each region. Two tiles are considered neighbors if they share a side or a corner in the tile map. The cost lies in $[0,1]$.

**Relative orientation cost** The change in the angles of the lines from a region centroid to its neighbors for the tile map compared to the original map. To ensure that each region contributes equally (some regions have more neighbors than others), we compute the mean for each region then take the mean of these. The cost lies in $[0,\pi]$.

**Roughness cost** The boundary of a tile map may be rough even if the tiles were fitted to a smoothed boundary polygon. Roughness is penalized since tile maps with overly rough boundaries look messy and unattractive. On the other hand, care is required since penalizing roughness too aggressively can smooth out genuine boundary features. The roughness cost is based on the number of shared tile edges. In general, fewer shared edges indicates a rougher tile map. A cost based on shared edges has the added advantage of penalizing holes in the tile map—which are typically undesirable. A disadvantage of using shared edges is that elongated boundary shapes are more heavily penalized than compact shapes. Formally, we define the roughness cost as:

$$q_{rough} = \frac{nR - m - P(n,R)}{P(n,R)} \tag{7}$$

where $n$ is the number of edges of each tile, $R$ is the number of tiles, $m$ is the number of shared edges in the tile map and $P(n,R)$ is the minimum possible perimeter of a shape with the same area as the tile map—when computing $P(n,R)$, we assume that tiles are regular polygons with edges of length 1. Since $P(n,R)$ corresponds to a circle, it is easy to show that:

$$P(n,R) = 2\sqrt{\pi a_n R}$$

where $a_n$ is the area of a tile with unit length sides. Subtracting $P(n,R)$ in Equation (7) shifts the roughness cost towards zero whist ensuring that it remains positive. Dividing by $P(n,R)$ means $q_{rough}$ is measured in units of $P(n,R)$.

We combine the above costs into a total cost using:

$$Q = \sum_{j \in costs} \lambda_j q_j \tag{8}$$

where the $q_j$ are the costs described above and the weights $\lambda_j \geq 0$ specify the relative importance of each cost. $Q$ is used to rank the candidate tile maps. Note that the costs are defined in such a way that the $q_j$ will be of the same order of magnitude. Hence, we initially set the $\lambda_j$ to 1, and then allow the tile map designer to adjust them as required to rerank the candidates.

### 4.6. Prototype Application

Figure 5 shows a screenshot of our prototype HTML-based application. The user specifies the data, tile shape (square, hexagon-flat-base, hexagon-flat-side, triangle-flat-base, triangle-flat-side) and options that dictate the number of candidate tile maps and the extent to which they vary (Sections 4.1–4.3). The application generates the tile maps and displays the top 20 (by default) in rank order based on the current cost weights. These weights can be adjusted and the tile maps will be re-ranked and displayed.

A plot showing the contribution of the individual costs helps the user to adjust the weights appropriately if required. Furthermore, each region of each plot is colored to show its contribution to the total cost for that tile map. Clicking on a tile map allows the user to view the original map, details of the transformation (Figures 4a and 6a), as well as the individual costs by region. This region-level information is useful when a particular subarea of the map has contextual importance and hence must be accurate.

The prototype application is written in HTML and JavaScript, and all computation takes place in the browser (client-side). Currently, the application can only import 6 test maps that have been partially pre-processed (polygons simplified and boundary polygon computed) using the Mapshaper software [Blo13]. Planned extensions to the application are discussed in Section 5.

### 4.7. Performance and Complexity

The runtime of the algorithm increases with $R$, the number of regions. As an example, creating 192 tile maps for the 50 countries of continental Africa ($R = 50$) takes 2.2 seconds whereas the same task for the 94 departments of mainland France ($R = 94$) takes 5.2 seconds; creating a single tile map for the 374 local authorities on the island of Great Britain ($R = 374$) takes approximately 4 seconds (times based on Chrome 55, no web workers, Intel Core i7-6700HQ, 2.6GHz). Note that improvements to the prototype such as optimizing the JavaScript code or moving the computation to the server could have a significant impact on performance.

The algorithm does not require (nor does it benefit from) high resolution shape data so we simplify the region polygons if appropriate before computing the required features: boundary polygon, adjacency matrix, region centroids. Computing the region-tile assignments is the most expensive step of the algorithm. The Hungarian method used for this has complexity $\mathcal{O}(n^3)$, *i.e.* it is cubic with respect to the number of regions. Once the candidate maps and their costs have been computed, the candidates can be reranked very quickly when the cost weights are adjusted—we simply reevaluate Equation (8) for each candidate and sort the results.

## 5. Examples and Discussion

Figures 5–9 show examples of tile maps created using the prototype application. In all figures, the color of each region shows the total cost. The colors are normalized so that the lowest cost region on the map is white and the highest cost region on the map is red. In all cases except Figure 5, the costs correspond to the default weights (all 1). Note that costs are only comparable for the same geographic map and tile shape.

Figure 5 shows the top-ranked tile map for various weights. The tile maps in Figures 7, 8 and 9b were all ranked first or second (using the default options and weights) except for the flat-sided hexagon map in Figure 7 which was ranked sixth. The Africa tile maps in Figure 9a required a little more effort. In particular, we investigated non-default values of the smoothness option and reranked the maps using a high roughness weight to find the triangular maps shown (though the costs shown are for the default weights).

Typically, we find that the highest ranked tile maps are based on global boundaries that have been fully or almost fully transformed—a "transform" value of 1 in the screenshots indicates fully transformed. This suggests that the transformations described in Sections 4.1 and 4.2 have the desired effects. It also suggests that simply fitting tiles to the original (untransformed) boundary is suboptimal. This is not surprising given the large variation in region size and shape for most maps and that it is common for regions to have neighbors of a similar size. As discussed in Section 3, the north-east of the USA has relatively small states. When a tile map is fitted to the original boundary, eastern states are pushed to the center (Figure 10). In this case, the sacrifice of *Local Relationships* and *Global Position* to preserve *Global Shape* is unacceptable—there will be poor agreement between Figure 10 and peoples' mental models for the east of the country. In general, we need extra space in areas of high region density and less space in areas of low density. This allows regions to stay in the 'right place' at the cost of deforming the global shape. The best tile map is likely to be a balance between making/removing space where necessary and retaining the global shape. The correct balance will depend on the specific geography and task, but the examples demonstrate that our approach can be used to create good tile maps for a range of challenging geographies.

In future work, we will explore the theoretical and empirical behavior of the centroid and boundary transformations in more detail. The iterative algorithm for transforming the centroids works well in our experiments, but we would like to establish if there are degenerate cases or point sets that demonstrate slow convergence. The current use of $k = 3$ nearest centroids when transforming the boundary is somewhat arbitrary. Increasing $k$ leads to a smoother boundary, but also makes the transformation less sensitive to local—possibly visually salient—boundary features. Further investigation is required to find the optimal value of $k$ and to decide if it is even reasonable to use the same value of $k$ for all maps. If it is not, possible solutions include estimating $k$ from the geographic

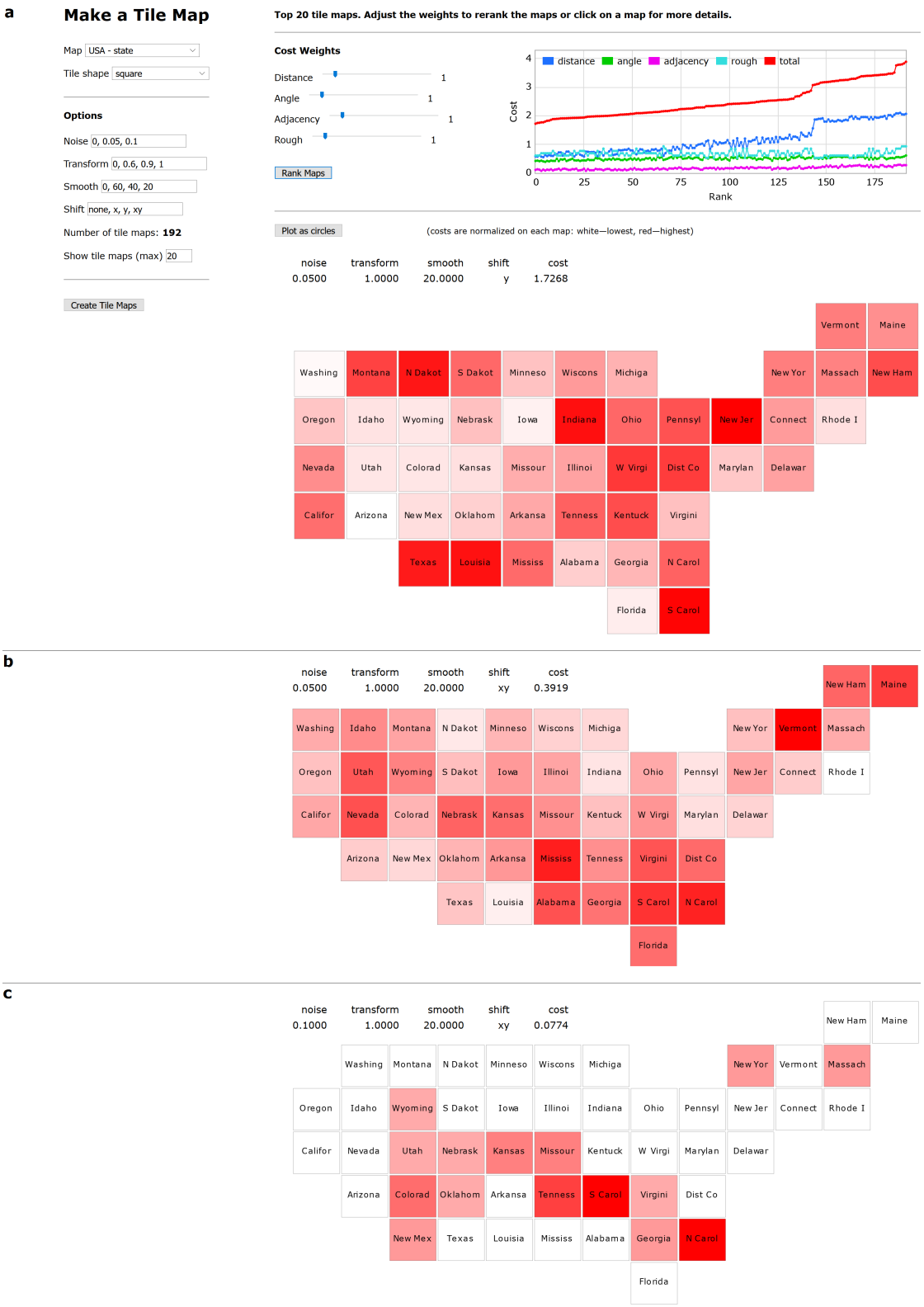


**Figure 5:** *States of the USA; screenshots from browser-based prototype with default options (left side panel). Top-ranked map for: (a) default weights (all equal to 1); (b) relative orientation (angle) weight 1, other weights 0; (c) adjacency weight 1, other weights 0.*

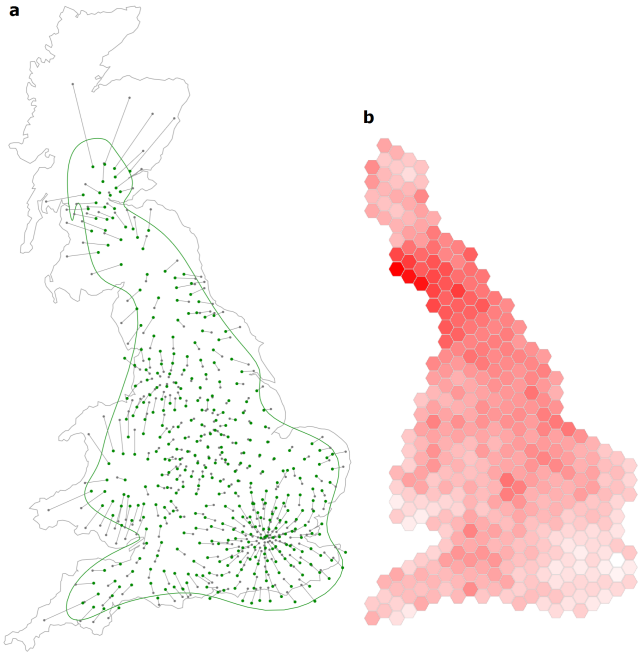


**Figure 6:** *Local authorities of the UK. (a) Points are local authority centroids: gray–original (no noise), green–fully transformed. Lines: gray–original boundary, green–transformed boundary (smoothed, 30 basis functions). (b) Hexagonal tile map corresponding to (a) (grid shift: half step along x-axis).*

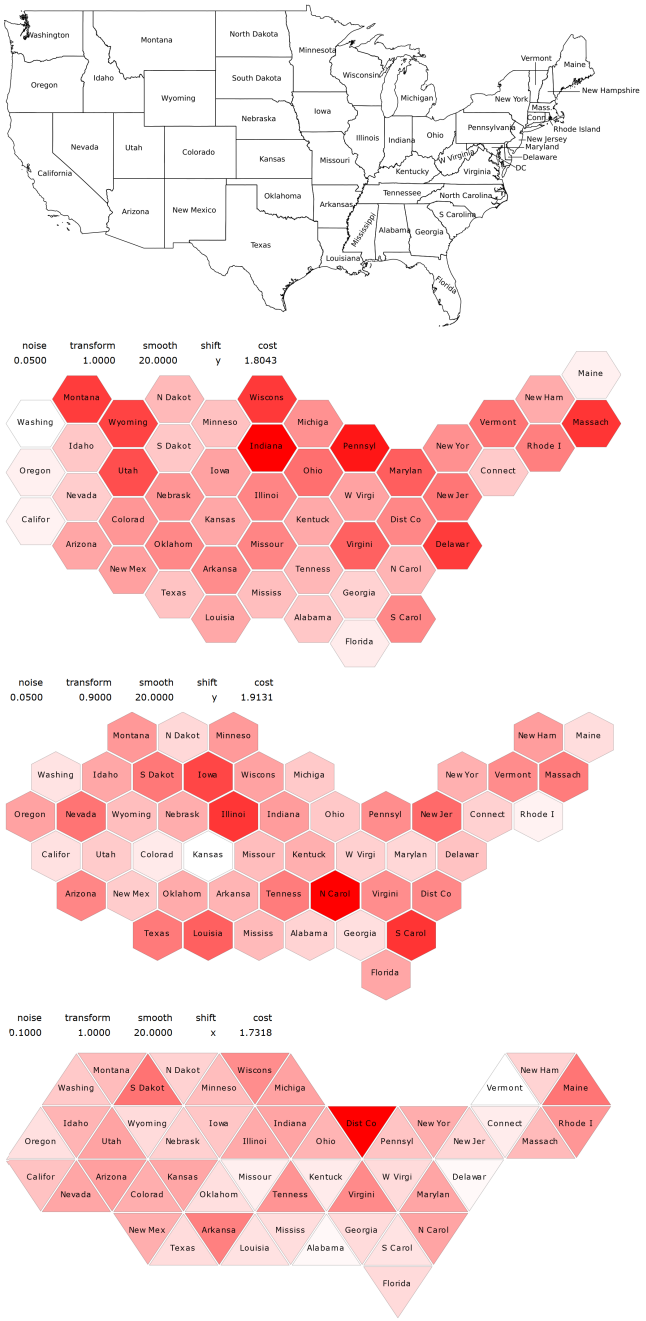


**Figure 7:** *States of the USA; hexagonal and triangular tile maps.*

shape data or generating candidate tile maps for multiple values of $k$.

Focusing on the concrete applications of our work, we are currently extending the browser-based prototype to enable import and basic preprocessing of shape data, import of a thematic variable (a value for each region) and the export of a selected tile map. Longer term, there are many features that could easily be added. For example, we could allow the user to choose the aspect ratio (width-to-height ratio) of tiles or choose the aspect ratio automatically based on the geography. Including alternative/additional costs is another option. In particular, we would like to experiment with a more complex measure of tile adjacency (currently this is binary and tiles are considered adjacent if they share a corner). Other interesting possibilities include allowing the user to specify important regions (easily incorporated into the costs) or to specify that given regions must remain on the boundary. Finally, we would like to investigate alternative methods for displaying the candidate tile maps. For example, using small multiples would allow users to filter many candidates quickly based on global shape.

In this paper, we have focused on the algorithmic challenge of creating tile maps. However, the complementary problem of quantitatively assessing the quality of a tile map is also important. In particular, user studies are required to understand how properties such as adjacency, relative orientation and global shape correlate with human assessments of tile map accuracy and their ability to use a tile map for common tasks. This is a complicated issue since people will weigh the importance of map features and properties based on their personal preferences, experiences and prior knowledge of the geography. For example, Wongsuphasawat [Won16b] discusses six different tile maps of the contiguous USA. The fact that six designers (or teams) produced different tile maps for a well-known geography of only 48 regions highlights the subjective nature of the problem.

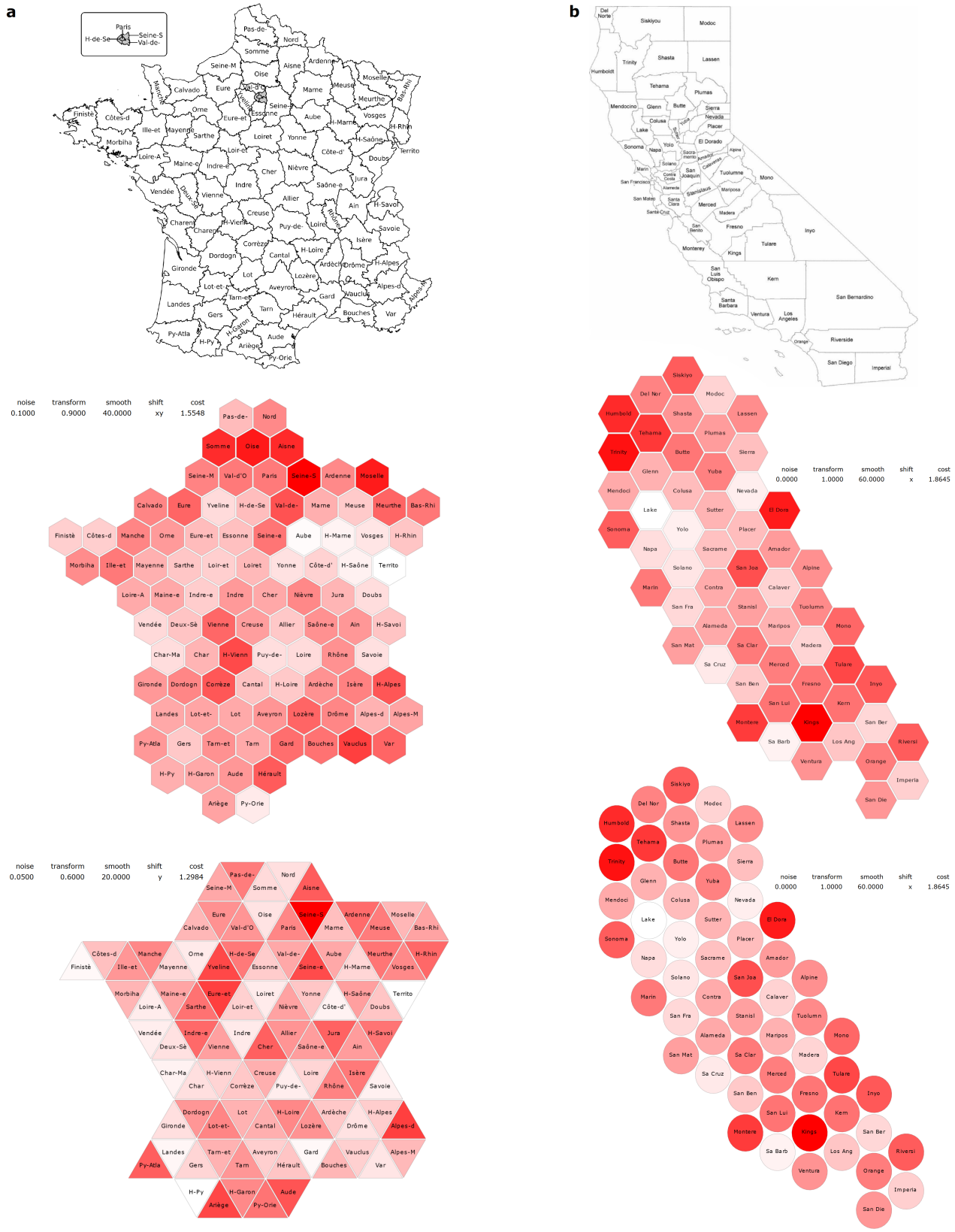


**Figure 8:** *(a) Departments of France. (b) Counties of California (The bottom map shows the same tile assignments plotted as circles).*


**Figure 9:** *(a) Countries of continental Africa. (b) Boroughs of London.*

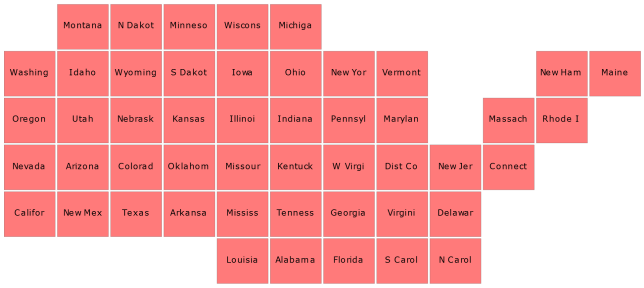


**Figure 10:** *A poor tile map. States of the USA; no centroid noise, centroids (and hence boundary) not transformed, boundary smoothed (30 basis functions), no grid shift.*

## 6. Acknowledgments

We would like to thank our academic colleagues and our reviewers who provided helpful feedback on this research. This publication was supported by Innovate UK as part of the NEXUS project (grant reference NE/N00728X/1) and The Alan Turing Institute (EPSRC grant EP/N510129/1).

## References

[AKV*15] ALAM M., KOBOUROV S. G., VEERAMONI S., ET AL.: Quantitative measures for cartogram generation techniques. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 351–360. 2

[BDC02] BORTINS I., DEMERS S., CLARKE K.: Cartogram types. http://www.ncgia.ucsb.edu/projects/Cartogram_Central/types.html, 2002. Accessed: 2016-12-10. 1

[Blo13] BLOCH M.: Mapshaper. http://mapshaper.org/, 2013. Accessed: 2016-12-10. 6

[CM15] CASSELMAN B., MCCANN A.: Where your state gets its money. http://fivethirtyeight.com/features/where-your-state-gets-its-money/, 2015. Accessed: 2016-12-10. 2

[Cra73] CRAWFORD P. V.: The perception of graduated squares as cartographic symbols. *The Cartographic Journal 10*, 2 (1973), 85–88. doi:10.1179/caj.1973.10.2.85. 1

[DeB15] DEBELIUS D.: Let's tesselate: Hexagons for tile grid maps. http://blog.apps.npr.org/2015/05/11/hex-tile-maps.html, 2015. Accessed: 2016-12-10. 2

[DM98] DRYDEN I. L., MARDIA K. V.: *Statistical Shape Analysis*. Wiley, Chichester, 1998. 3

[Dor96] DORLING D.: Area cartograms: their use and creation. In *Concepts and techniques in modern geography* (1996), Citeseer. 1

[EvKSS15] EPPSTEIN D., VAN KREVELD M., SPECKMANN B., STAALS F.: Improved grid map layout by point set matching. *International Journal of Computational Geometry & Applications 25*, 02 (2015), 101–122. 2, 3

[Fla71] FLANNERY J. J.: The relative effectiveness of some common graduated point symbols in the presentation of quantitative data. *The Canadian Cartographer 8*, 2 (1971), 96–109. doi:10.3138/J647-1776-745H-3667. 1

[FS07] FELZENSZWALB P. F., SCHWARTZ J. D.: Hierarchical matching of deformable shapes. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (2007), IEEE, pp. 1–8. 3

[Gri85] GRIFFIN T.: Group and individual variations in judgment and their relevance to the scaling of graduated circles. *Cartographica: The International Journal for Geographic Information and Geovisualization 22*, 1 (1985), 21–37. doi:10.3138/0X46-G750-6261-R826. 1

[KN16] KOBOUROV S., NUSRAT S.: The state of the art in cartograms. In *Computer Graphics Forum* (2016), WILEY-BLACKWELL. 2

[MDS*17] MEULEMANS W., DYKES J., SLINGSBY A., TURKAY C., WOOD J.: Small multiples with gaps. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 381–390. 2, 3

[Mun57] MUNKRES J.: Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics 5*, 1 (1957), 32–38. 5

[MV05] MCNEILL G., VIJAYAKUMAR S.: 2d shape classification and retrieval. In *Proceedings of the 19th international joint conference on Artificial intelligence* (2005), Morgan Kaufmann Publishers Inc., pp. 1483–1488. 3

[MV06] MCNEILL G., VIJAYAKUMAR S.: Hierarchical procrustes matching for shape retrieval. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (2006), vol. 1, IEEE, pp. 885–894. 3

[Ric16] RICHARDS N.: Where are the africa grid/tile maps? https://questionsindataviz.wordpress.com/2016/09/01/where-are-the-africa-gridtile-maps/, 2016. Accessed: 2016-12-10. 1, 2

[Sha16] SHAW T.: Good data visualization practice: Tile grid maps. https://forumone.com/ideas/good-data-visualization-practice-tile-grid-maps-0, 2016. Accessed: 2016-12-10. 2

[SMKH09] SLOCUM T. A., MCMASTER R. B., KESSLER F. C., HOWARD H. H.: *Thematic Cartography and Geovisualization*, 3rd ed. Pearson Education, 2009. 1

[tC80] TSUNG CHANG K.: Circle size judgment and map design. *The American Cartographer 7*, 2 (1980), 155–162. doi:10.1559/152304080784523107. 1

[TC15] TRIBOU A., COLLINS K.: Interracial marriage. http://www.bloomberg.com/graphics/2015-pace-of-social-change/, 2015. Accessed: 2016-12-10. 2

[VH01] VELTKAMP R. C., HAGEDOORN M.: State of the art in shape matching. In *Principles of visual information retrieval*. Springer, 2001, pp. 87–119. 3

[WD08] WOOD J., DYKES J.: Spatially ordered treemaps. *IEEE transactions on visualization and computer graphics 14*, 6 (2008). 2

[WDS10] WOOD J., DYKES J., SLINGSBY A.: Visualisation of origins, destinations and flows with od maps. *The Cartographic Journal 47*, 2 (2010), 117–129. 2

[Won16a] WONGSUPHASAWAT K.: A semi-automatic way to create your own grid map. https://medium.freecodecamp.com/creating-grid-map-for-thailand-397b53a4ecf#.4tbo5jror, 2016. Accessed: 2016-12-10. 2

[Won16b] WONGSUPHASAWAT K.: Whose grid map is better? quality metrics for grid map layouts. https://medium.com/@kristw/whose-grid-map-is-better-quality-metrics-for-grid-map-la.6ghxo6wyw, 2016. Accessed: 2016-12-10. 8

[Wul] WULFF M.:. URL: https://github.com/informeren/qgis-cartogram. 2

[Zac15] ZACHARY R.: Data visualization strategies using tile grid maps. https://www.gislounge.com/data-visualization-strategies-using-tile-grid-maps/, 2015. Accessed: 2016-12-10. 2