Dynamic Choropleth Maps - Using Amalgamation to Increase Area Perceivability

Liam McNabb
Visual & Interactive Computing Group
Swansea University
Swansea, Wales
661370@swansea.ac.uk

Robert S. Laramee
Visual & Interactive Computing Group
Swansea University
Swansea, Wales
r.s.laramee@swansea.ac.uk

Richard Fry
National Centre for Population Health and Wellbeing Research, Medical School
Swansea University
Swansea, Wales
r.j.fry@swansea.ac.uk

***Abstract*—Choropleths are a common and useful way of depicting area-coupled data on a geo-spatial map. One advantage they provide is combining area-based data accurately with geo-space. However perceptual problems arise when areas are too small, i.e when they only cover a few pixels or less. This is a very common occurrence when zooming or in densely populated areas like capital cities. We present a novel algorithm that ensures the user is able to observe area-based data coupled to geo-space based on their interactive level of zoom without distorting the original geo-spatial map. This is resolved by building a hierarchical data structure in which each area and its data is merged with one of its smallest neighbor recursively until only one polygon covers each contiguous region. The benefits are that the viewer can always view area-based data contained in the map regardless of how small any individual area becomes during interactive zooming. We break down each step of the algorithm and provide pseudo-code to enable reproducibility. We also discuss unique test cases that challenge the robustness of the algorithm with 30,000 polygons and 4,652,800 vertices as well as the performance.**

***Index Terms*—Information Visualization, Choropleth Maps, Cartographic Generalization, Hierarchical, Zooming, Perceivability, Geospatial**

I. Introduction

Choropleth maps can be defined as displays where data is aggregated using administrative units and normalized values [33]. Choropleths are ubiquitous for conveying area-based data on a geo-spatial map because they are intuitive and preserve geo-spatial information. However, because they do not distort geospatial boundaries, areas may be too small to perceive any data (see Figure 4). This is especially true in the context of zooming where an area may not even cover a full pixel. Area-based data is often too dense to perceive in capital city regions. Ward *et al.* state, "*A problem of choropleth maps is that the most interesting values are often concentrated in densely populated areas with small and barely visible polygons, and less interesting values are spread out over sparsely populated areas with large and visually dominating polygons*" [51].

We focus on maintaining perceivable areas without map distortions by developing an area-merge algorithm that provides a user-controlled parameter, m, to display area units or area unit clusters that meet a minimum screen-space requirement. Rao and Card define such an adjust operation as "*...change the amount of contents viewed within the focus area without changing the size of focus area*" [36]. By introducing a hierarchical representation of the choropleth, we can update the display quickly and enable changes to the level of detail for the best visual experience. We call this a dynamic choropleth map. Our zooming is smooth and continuous. By this we mean there are no jumps, distortions, or disruptions during the zooming. The level of detail changes dynamically and interactively without distorting the geometry. Changes in zoom level must be smooth and not rely on distortion of the geo-space or any areas contained within.

Our contributions include:

- A novel algorithm to interactively zoom smoothly, providing appropriate and perceivable levels of detail for choropleth maps.
- Providing a set of pseudo-code to enable reproducibility of the method.
- The application of our algorithm to complex, real-world shapefiles including those with over 10,000 unit areas and over 4.5 million vertices.

To provide this functionality, challenges must be overcome including developing an algorithm that detects when unit areas become too small, joining boundaries, building an appropriate area hierarchy, and zooming dynamically and continuously whilst preserving the traditional choropleth properties.

In Section II, we review previous work on interactive zooming and choropleth maps. Section III discusses the proposed methodology of the algorithm, a general overview of the procedure and the individual steps required. Section IV discusses results and performance including benefits and limitations. Section VI looks at potential future work and conclusions.

II. Related Work

We examine three main branches of related work which include zooming, choropleth maps, and cartographic generalization. The Survey of Surveys for information visualization [32] identifies one related survey paper on clutter reduction, no related surveys on the topic of choropleths or surveys focused on geo-spatial zooming, and one survey focused on hierarchical aggregation [18]. Ellis and Dix provide a taxonomy of clutter reduction for information visualization and review 11 clutter

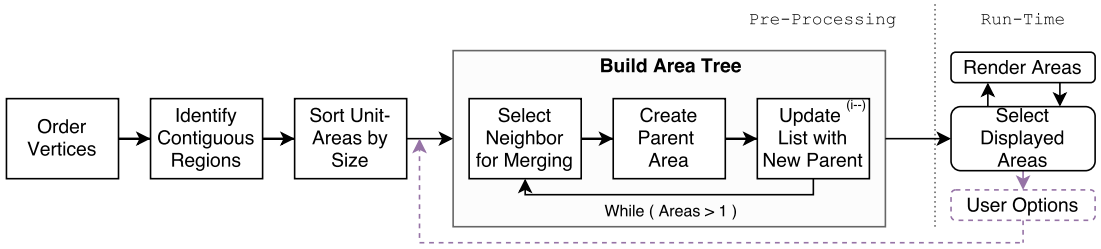


Fig. 1: The pipeline for the area amalgamation algorithm. After loading the shapefile, polygons are partitioned based on area contiguity, and sorted within islands (or land masses) based on their size. A recursive function is then used to identify new parent areas and their boundaries until their are no remaining neighbors to merge. See section III for details.

reduction techniques including clustering, space-filling, and animation [17].

A. Zooming

Cockburn *et al.* review pan+zoom used in over 15 research papers, and examine overview+detail, zoom, and focus+context [13]. Rao and Card discuss the use of zooming for tabular information in the context of interactive manipulation of focus (zoom, adjust, and slide) [36]. We require that the view and geometry are not distorted in any way in our work. Jog and Shneiderman present the zoom bar and introduce a zooming approach based on zooming towards a fixed line within a starfield visualization [26]. This differs from our work that focuses on choropleths. Van Wijk and Nuij provide an algorithm for smooth and efficient zooming across 2D planes [47] and extend on this idea by looking at non-uniform scaling between two planes [52]. They derive an optimal camera path for smooth zooming and panning. This is likely the previous work most similar to ours. Their work does not consider regions that may be too small to perceive which differs from our work. Also the choropleth map is dynamic in our case. Javed *et al.* present a zooming technique titled PolyZoom where a user progressively builds a hierarchy of focus regions to zoom between [24]. Polyzoom focuses on different scales of maps separately whereas we endeavor to provide a continuous zooming method. Axelsson *et al.* tackle challenges addressing visualization between large scales of information for astronomical data using scale scene graphs [4] which differs from our work that focuses on a single scene that must be smooth and continuous. Google Maps provides a map of the earth which enables the user to zoom on user-selected areas. Moving between zooming levels comes with sudden, discontinuous transitions between levels of detail which we avoid [22]. Both Akelsson *et al.* and Google Maps process image data broken up into rectangular tiles. Our algorithm processes original unit areas and handles geo-spatial boundaries composed of vertices and edges.

Blanch and Lecolinet provide zoomable treemaps that pan and snap-zoom between different levels within a tree map [6]. Roberts *et al.* extend Van Wijk and Nuij's zooming work applying their smooth zooming algorithm to tree maps, and combine this with a smooth transition between levels of detail [39]. Our work differs from Roberts *et al.* as our approach maintains a smooth and continuous transition between zoom levels, and selects what to display based the zoom level and a user-specified parameter. In addition, our work handles much more complex area-unit boundaries because it processes choropleths.

B. Choropleths

Digital choropleth maps have been produced prior to 1970 with the U.S Department of Commerce citing 10 choropleth mapping systems [1]. From our related work literature search we find previous work on choropleths focus on class intervals (or systems) rather than zooming. A class is defined as a mutually exclusive and non-overlapping set of grouped data whilst a class interval is defined as the selected width (or range of data) of each class [23]. Tobler questions the use of class intervals within choropleth maps by reviewing the use of inked area vs. white area to display values [46]. Brewer and Pickle provide a qualitative study on class intervals for choropleth maps comparing seven different methods [10]. Zhang and Maciejewski detect critical boundary cases within choropleth maps where statistical measures fall near the selected classification bounds [54]. This informs them of optimal selection of class intervals for data representation. Pickle presents a guideline for map design including color selection, legend design and smooth transition between color within area-units [35]. Slocum *et al.* provide a full chapter on Choropleth Mapping which includes 58 references [41] spanning 1957 [43] to 2006 [3]. They discuss decision-making behind classed and un-classed maps, appropriate color schemes, and designing the legend of the map [41]. Dykes and Brunsdon introduce new techniques for geographically weighted visualization using scalograms [16]. Each of these papers places emphasis on class intervals, whilst our paper focuses on perceivable individual areas on a dynamic map.

Andrienko and Andrienko briefly survey the overall spatial distribution of data with diverging color scales in choropleth maps, and provide an example of animated choropleth map displays with small multiples [2]. We do not review color scales or the use of temporal data in choropleth maps.

Jern *et al.* use linked views to observe regional development data using both a choropleth map and tree map [25]. Our paper focuses on adding a new dynamic feature to choropleth maps rather than combining them with other techniques. Dang *et al.* present a generalized map-based information tool for dynamic queries and brushing on choropleth maps [14]. Our

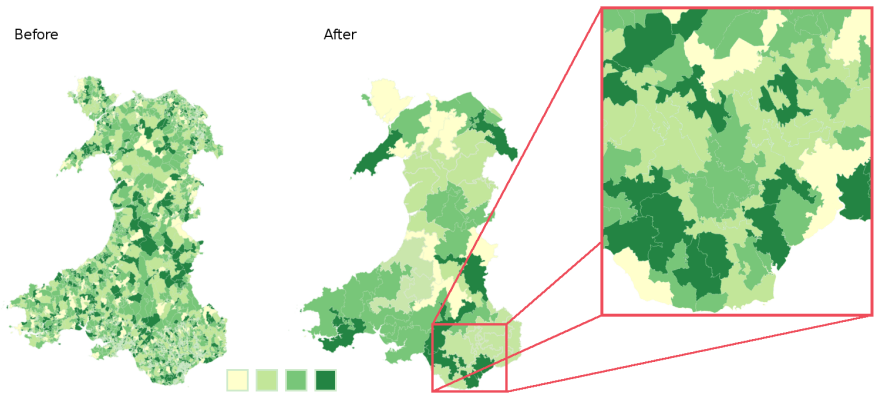


Fig. 2: Example of the procedure applied to Wales [50]. The left image shows the original image with over 10,000 output areas having $4,652,800$ vertices [49], where we can see a dense clutter of indistinguishable areas in the south-east section. The right images shows the effects of the procedure at two different zoom levels (indicated by the red box), where m is 2%. Areas are color-mapped using colorbrewer color palette [9].

work focuses on zooming rather than brushing. Li and Han look at applying the Lorenz curve to choropleth mapping to identify numerical trends [29]. We focus on user perceivability rather than new trends in data. Johansson *et al.* present a web-based visualization tool that combines the use of choropleth maps with dashboard functionality in order to review multifaceted information on climate change and adaption measures [27]. We focus on perceivability of unit areas, rather than the use of a choropleth map for climate change data. Speckmann and Verbeek present necklace maps which present choropleth maps with juxtaposed proportional symbol maps that allow the user to understand size data without distorting the topological view [48]. We develop interactive, smooth zooming in order address similar issues.

Rittschof and Kulhavy present a user-study which includes a comparison of choropleth maps and cartograms. Cartograms are a different class of related work considering a wide range of techniques (Gastner-Newman [21], Dorling [15], etc.) which use distortion to convey data. We want to avoid introducing geo-spatial error into the map in our technique. Their results found choropleth maps were associated with greater recall of information [38]. Kasper review the effectiveness of Gastner-Newman diffusion cartograms [21] for the representation of population data, which includes a comparative experiment against thematic maps (choropleth with overlayed circle maps). The results report that the thematic maps are more efficient and effective, specifically with complex tasks [28]. Sun and Li review the effectiveness of cartograms for the representation of spatial data, which includes a comparative experiment against thematic maps including choropleths. The results indicate that the thematic maps are more effective representing quantitative data, whilst cartograms were more effective with qualitative data [44].

To the best of our knowledge, no previous work focuses on dynamic and continuous zooming of choropleth maps while maintaining perceivable area units without distortion.

C. Cartographic Generalization

Slocum *et al.* provide a full chapter on Cartographic scale and generalization [41]. The chapter defines generalization as: *"the process of reducing the the information content of maps because of scale change, map purpose, intended audience, and/or technical constraints"*, and reviews models of generalization include the models of Robin *et al.* [40] and McMaster and Shea [31]. Slocum *et al.* define the fundamental operations of generalization as simplification, smoothing, aggregation, amalgamation, collapse, merging, refinement, exaggeration, enhancement, and displacement. Our algorithm uses recursive amalgamation on a per-area basis.

Elmqvist and Fekete provide a survey on hierarchical aggregation for information visualization [18]. The survey only provides one spatial aggregation techniques by Andrienko and Andrienko (discussed below). Andrienko and Andrienko briefly discuss aggregation with earthquake occurrences in Turkey [2]. They use a density map to aggregate the occurrences per rectangular grid cell. Andrienko and Andrienko's generalization approach looks at point data, whilst we focus on areas. Zhang *et al.* present a novel visualization technique titled 'TopoGroups' [53] used to group spatial data into hierarchical clusters to minimize visual clutter. Boundaries are used to present data topics as a stipple line, where the ratio of a stipple represent that of the data. We focus on polygon unification rather than point data.

Regnauld and Revell discuss their automatic amalgamation method used in producing the ordnance survey's scale maps [37]. The paper uses a number of generalization techniques to select clusters (triangulation, proximity, and edge filtering) and manipulate the clusters to give a visually clear representation of amalgamated buildings. Our paper looks at areas rather than buildings and is used for only contiguous areas. Li *et al.* review amalgamation of buildings based on the Gestalt principles of design [30] which include separation, length, and area thresholds as well as similarities in shape, size and orientation. Our amalgamation technique does not allow for any separation and unites two areas instead.

III. Methodology

We begin with an overview of our methodology before discussing each step in detail. The algorithm is based on the premise that each area, starting with the unit areas, can be merged with its closest neighbor from smallest to largest to create a smooth and continuous transition for perceptible areas.

A. Method Overview

In order to effectively enable smooth and continuous zooming at run-time, we use pre-processing. We build a hierarchical data structure before displaying the choropleth. For this we have created a pre-processing pipeline shown in Figure 1. We first load each unit area represented by a polygon, p. A polygon p is a list of vertices: $p = \{v_0, \ldots, v_n\}$. We then update the order of each unit-area's list of vertices to ensure that they are in clockwise order. The next step is to identify contiguous regions. Here we separate contiguous regions into islands (or land masses) which enforces topological continuity. Once each contiguous region is identified, each unit-area within the same contiguous region is sorted by size since scale is an important part of the algorithm. It is more efficient to sort before building the hierarchical data structure.

The hierarchy construction is a recursive algorithm broken down into three sub-routines. As the regions are pre-sorted from smallest to largest, we know the first area merge candidate (p_1) is at the front. We must then find the second merge candidate (p_2) by selecting one of p_1's neighbors using a distance function. When we have found a merge pair (p_1, p_2) we identify both the shared (b_s) and non-shared (b_{ns}) boundary of each, and combine such that p_1 and p_2 unite using only their shared boundary to create a new area P.

$$P = (p_1 \cup p_2) - (p_1 \cap p_2) \tag{1}$$

This is stored as a parent in the hierarchical data structure. When this is done, we can then remove the p_1 and p_2 from the merge candidates list and insert the new parent P into the list preserving sorted order (by size). When there are no remaining neighbor candidates, the hierarchy is complete. When this is done for each contiguous region, we have the necessary hierarchical data structures for smooth zooming and clustering.

With the hierarchies built, display is relatively simple. By specifying a desired minimum screen space, m, using the current zoom level and comparing that to each tree node's size using a depth-first search (DFS) in the hierarchy, we can select the appropriate polygons to display. An example of the results can be found in Figures 2 and 4.

B. Order Area Polygon Vertices

Our first step is to order the original vertex data from the shape file. This is important in order to reduce complexities in later stages. It allows us to simplify the identification of and unification of boundaries ($p_1 \cup p_2$). For this we use the shoelace formula (also known as Gauss's Area Formula or Surveyor's Formula), which allows us to derive both the area (useful for later) and the orientation [8].

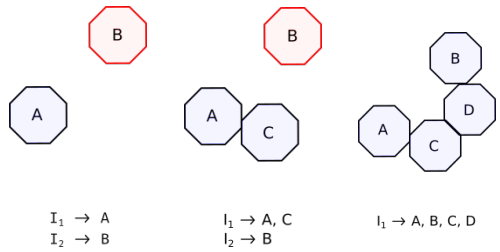


Fig. 3: Visual example of the contiguous regions procedure. This shows how a potential contiguous region can be derived over three steps. See Section III-C.

$$a = \frac{1}{2}\left|\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i - x_1 y_n\right| \tag{2}$$

The notation x and y refer to the coordinates of each vertex and n refers to the number of vertices in p. If we remove the absolute value, we can deduce that if the area is negative, the vertex list is counter-clockwise, and we can reverse the list order. Unit-area's with multiple contiguous regions are also split up to enforce topological continuity. We process these islands (or land masses) as individual areas. We must also test for uncommon inner rings or any other vertices related to the shape. These can be saved in a separate list to aid in rendering, however these must also be searched during boundary processing, as a ring found in unit-areas is usually formed as a result of a fully surrounded unit-area. In our Wales example (Figure 2) we find 31 instances of inner rings out of 30,000 polygons.

C. Identifying Adjacent Neighbors & Contiguous Regions

After ordering each unit-area's vertex lists, we can identify the contiguous regions. This is important for us in order to prevent a merge of two islands. The most important consideration is identifying what is classified as a neighbor. We provide pseudo-code for this in Algorithm 1 in the Supplementary Material (refer to Section V).

We first test p_1 and p_2's bounding boxes for overlap. By comparing Axis Aligned Bounding Boxes (AABB's) which use the maximum and minimum values for each axis of the areas p_1 and p_2 [19], we ensure the in-depth neighbor checking is applied to as few areas as possible.

If p_1 and p_2's AABB intersect, we test their vertex lists for common points. Algorithm 1 in the Supplementary Material (refer to Section V) uses a simpler approach where we assume that all points have a matching point in a neighbor's vertex list. If areas with long straight edges (like some US states) are used to define unit-areas, we find cases where we need to use a second test to identify whether a point intersects a boundary edge (examples of this include T-junctions). We define neighbors as two polygons with at least two unique common vertices. We do not consider one common vertex as a boundary edge. The start and end of a shared boundary b_s must

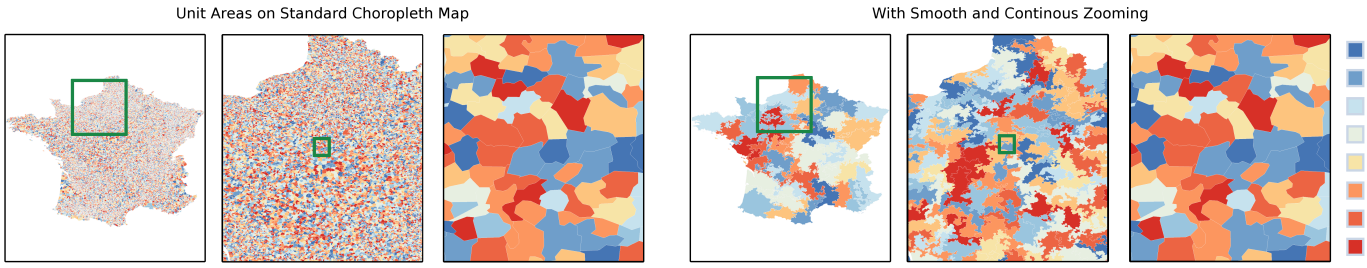


Fig. 4: A comparison between a shape file representing France with over 30,000 administrative units and 729,565 vertices before and after the implementation of smooth zooming at 3 different levels of zoom, with minimum required screen space (m) of 1%. Mapped colors from colorbrewer color palette [9].

also be considered the end and start of a non-shared boundary $b_n s$ to enforce topological continuity of the unit areas.

Now that we can identify adjacent neighbors, we identify the contiguous regions. Pseudo-code is provided in Algorithm 2 in the Supplementary Material (refer to Section V). We assume that our first unit-area is an island and test this against every other island. If an island contains a neighboring unit-area, we know that every other region on that island is also linked. Knowing this, we can merge the two polygon lists and continue our search. See Figure 3. It is important that we do not finish the search here as our new unit-area may connect multiple islands together. Once this is done for each unit-area, we have identified each contiguous region and each of these can be sorted based on their size. Figure 3 provides an example of the procedure, whilst Figure 2 in the Supplementary Material (refer to Section V) shows a visual result of this step.

D. Building the Hierarchical Data Structure

We use a recursive procedure to create a hierarchical data structure. A hierarchy is created for each contiguous region, where each area (p_1) is merged with it's closest neighbor (p_2). Distance is measured using a general and flexible metric described in Section III-E. We start with a merge candidate list filled with the sorted unit-areas (for one contiguous region). The list is sorted by size. As mentioned in Section III-A, there are three main sub-routines: neighbor selection, creating the parent area (P), and updating the merge candidate list. If only a single unit-area remains in the merge candidate list, no further merges can be processed and we have finished the procedure. Here we denote p_1 as the first area merge candidate, p_2 as the second merge candidate and parent P (Equation 1).

E. Boundary Neighbor Selection & Amalgamation Criteria

In order to select an appropriate neighbor to join, we use a general and flexible distance metric for amalgamation evaluated between neighboring areas. We use this to measure a distance where the closest distance is considered the optimal selection for a neighbor. The measure consists of four constituents: Smallest area (a), euclidean distance between centroids (d), value variance (α), and shared boundary resolution (b_s). We formulate the measure as:

$$D = w_a.\frac{a}{a_{max}} + w_d.\frac{d}{d_{max}} + w_\alpha.\frac{\alpha}{\alpha_{max}} + w_{b_s}.(1 - \frac{b_s}{b_{s_{max}}}) \quad (3)$$

The distance metric includes weight co-efficients which enable the user to customize the importance (w) of each criteria as an option, with a default weight 0.5 for a, and a $\frac{50}{3}$ weight for d, α, and b_s. We define the criteria as:

- Smallest area (a). The criteria tests the size of a neighbor. Searching for small areas is the primary objective of the procedure and it is therefore important to take this into account during the distance measure. By doing this we reduce the number of small areas at a faster rate. We discuss how the area is calculated in detail in Section III-B (Equation 2). a_{max} is considered the area of the canvas' bounding box.
- Euclidean distance (d). This represents the shortest distance between two centroids. By taking the distance between centroids into account, we can enable more natural polygon formations to form. To calculate this we can use $(\sqrt{(|p_1(c_x) - p_2(c_x)|)^2 + (|p_1(c_y) - p_2(c_y)|)^2})$. The term d_{max} is the largest distance between all centroids.
- Data Value Similarity (α). Data is an important aspect of cartography and is considered when agglomerating areas. In order to factor it in the distance metric we look at the variance between the values of p_1 and p_2 $(|p_1(\alpha) - p_2(\alpha)|)$. α_{max} is the largest data value in the data range.
- Shared Boundary Resolution (b_s). Unlike the other criterion, we favor a larger shared boundary resolution. The shared boundary resolution refers to the topological length of a shared boundary, where a larger shared boundary defines a closer unification between two areas. This is calculated by running our merge algorithms early (refer to Section III-F for more detail) and normalizing it over the largest resolution area in the tree ($b_{s_{max}}$). Once this is done, we subtract the normalized value from 1 to impose a stronger weight for larger shared boundaries.

Using these criteria, we can select an optimal amalgamation candidate. We also provide the user the freedom to modify the criteria by using weighted coefficients. These can be modified after the procedure has been completed. This is a general and flexible distance metric because the distance metric itself is not a focus of the paper. Many such metrics have been studied in great detail [18].

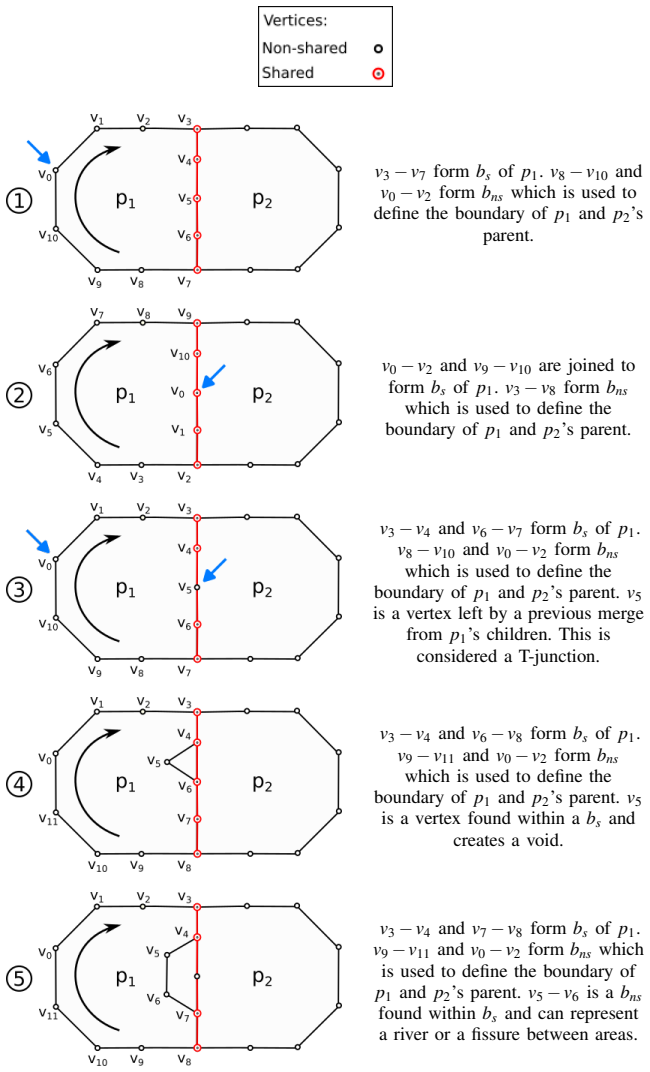


Fig. 5: Different cases for b_s and b_{ns} identification. Case 1 displays the basic case where a whole boundary is found in contiguous order. Case 2 provides a contiguous order, but is split due to the location of p_1's vertex list start index. Case 3 displays a T-junction which splits b_s into two segments. This could be resolved by point-line intersection testing. Case 4 and 5 represent voids and fissures which cannot be resolved by point-line intersection, with the fissure having a possible size of $b_s.length - 2$. We look at the length of common vertex chains to determine the start and end of b_s detailed in Section III-F

F. Creating Parent Area

Creating P includes 3 steps: (1) identify b_s and b_{ns} of each area's merge pair, (2) combining b_{ns} of the p_1 and p_2 for the boundary of the parent area P, (3) linking p_1 and p_2 to P for use in the rendering stage.

There are configurations which can cause unexpected challenges with the boundary identification. Firstly, the vertex list of each area is ordered but there is no given information about shared boundaries. This means that b_s can be found at any point within a vertex list, and can also start at any point with a vertex list. If our boundary search starts on b_{ns} and we search the vertices in clockwise order, as in case 1 of Figure 5, we can assume that the first common vertex is the boundary start. This is not the case for a first vertex found on b_s. In order to render the boundary correctly, we must not only identify b_s but also identify the start and end points of the boundary. Figure 5 illustrates various cases identified for b_s identification between two neighboring areas p_1 and p_2.

Due to voids and fissures representing by rivers or other geographical features, finding the start and end points of b_s can become complicated even when testing the entire vertex list. For example, if a vertex list begins on b_s that includes a fissure of n vertices, the selection of the b_s' beginning and end indexes becomes less obvious.

We provide our boundary identification process in Algorithm's 3 and 4 in the Supplementary Material (refer to Section V) which identify the start and end vertices of b_s. Firstly, we search and identify every common vertex between the area neighbors. As discussed in Section III-C, we assume that every common vertex has a matching vertex in their neighbor's vertex list, whilst shape files with simpler boundaries may need an additional point to line intersection test (T-junctions). From these vertices we can identify the beginning and end indexes of b_s (a common boundary between p_1 and p_2) by looking at the length of each common vertex chain. We use a heuristic that any voids and fissures found on b_s will be smaller in length compared to b_{ns} and therefore the longest chain between two common vertices signifies the chain between the end and the start of b_s. Figure 5 provides a visual presentation of boundary identification on some test cases encountered. This method handles cases with voids and fissures between neighboring polygons, as well as complications that can be caused by the T-junctions that may arise. For our Wales example in Figure 2 with over 10,000 unit areas (over 20,000 merges) and 4.5 million vertices we found 11,112 individual error cases caused by voids, fissures, and T-junctions. This means a non-trivial case is found in over 55% of the merges between p_1 and p_2.

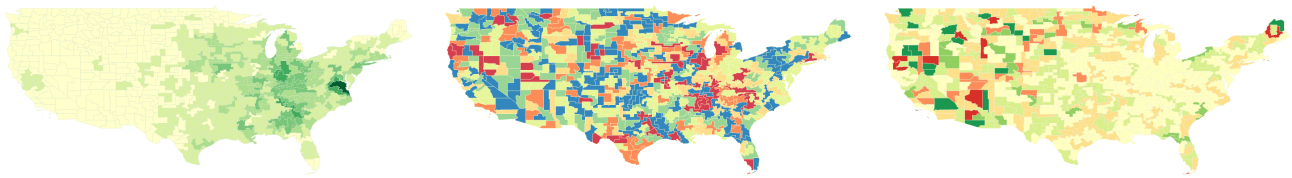
Knowing b_s's start and end indexes, we can easily separate the boundaries into b_s and b_{ns}. We can then combine the b_{ns} of an p_1 and p_2 in clockwise order to create the new parent area P. Once P's vertex list is updated, we create pointers that enable P to find it's children. This is important to enable traversal and selection within the hierarchical data structure. Algorithm's 3 and 4 in the Supplementary Material (refer to Section V) detail this process.

G. Updating the Sorted List with the Parent

We update the list preserving the sorted areas. We first remove the p_1 and p_2 from our merge candidates list as each area can only be merged with one other area. Then we can insert P into the list in sorted position based on its size. The procedure for building the hierarchical structure is found in Algorithm 5 in the Supplementary Material (refer to Section V).

H. Selecting Visible Boundaries

We select visible areas and boundaries based on a minimum area requirement, m, relative to the current screen space. As the screen space coverage changes based on the movement of the dynamic zoom level, we render different areas based on



(a) Sum (b) Frequency (c) Average

Fig. 6: 1 value-set displayed using 3 different base-calculation types using US counties (m=0.3%). (a) Represents using the sum to calculate the new values (sums). (b) Uses the highest frequency to represent values (qualitative data). (c) Uses the average of the value from all leaf nodes. See Section III-I.

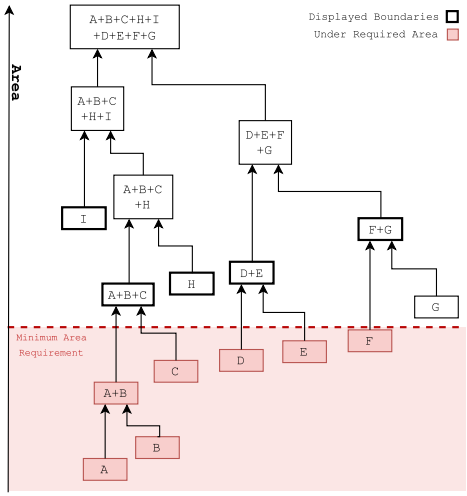


Fig. 7: An example of areas being selected and rendered. An area is only rendered if one or both child nodes are smaller than the minimum area requirement, m. Otherwise, perform a depth-first search until a leaf node is identified. In this example, **I, A+B+C, H, D+E,** & **F+G** are selected to be rendered. See Section III-H.

a zoom level and area size. The DFS identifies the smallest nodes in the tree that meet the minimum area size requirement. If any parent node is larger than the m, we test two criteria. (1) If the area is a leaf node, we can render the current node. (2) If either the left child or right child is smaller than m, then the current parent area is the smallest unit that meets the area requirement and is rendered. Completing the DFS will render only the smallest area within each branch that is larger than m. An illustrated example of this search can be found in Figure 7.

I. Storing Values of Amalgamated Areas

The Modifiable Areal Unit Problem (MAUP) [34] is an important aspect to consider when discussing the modification of boundaries or values. We address this by providing the user options to modify calculation of aggregated values as well as the weighted distance metric discussed in Section III-E. The data is linked to the administrative areas during the initial loading of the shape files. Before the area tree is built, the user can select the type of value amalgamation. This enables the user to choose options of sums, qualitative frequencies, and averages. When amalgamating values using sums, the value of P can be calculated as $P(\alpha) = p_1(\alpha) + p_2(\alpha)$. Qualitative values are calculated using frequencies. Using a DFS, P can count the frequency of each value for each leaf node and use the value of the most frequent of the leaf nodes. This is useful for categorical data. The average and weighted average can also be calculated using a DFS, by calculating the sum, $P(\alpha) = \sum_{i=0}^{i=n} \frac{p_{li}(\alpha)}{p_{li}(a)}$, where p_l denotes a leaf node in the tree. Examples are shown in Figure 6.

As well as these value criteria, these can be normalized at the rendering stage. Some examples of these normalization techniques include area ($\frac{P(\alpha)}{P(a)}$), population ($\frac{P(\alpha)}{P(\kappa)}$), as well as any ratio ($\frac{P(\alpha)}{P(\delta)}$).

Although the normalization can be turned on and off after the area tree is built. In order to change value representations, the build area tree procedure is re-run.

IV. Results and Performance

The desktop used to test this implementation features an Intel i5-4460 at 3.2GHz with 16GB of RAM and a GeForce GTX960. The implementation is developed using the Linux Mint 18 environment and the C++ framework of Qt. The software uses the Geospatial Data Abstraction Library to read the Shape File's unit-area information [45] and the OpenGL library to render the results.

We test 5 different shape files of varying resolution including US Counties, Japan, Italy, Wales and Germany found using the Global Administrative Areas website [20]. There is a large variance in the number of areas, average number of vertices, total contiguous regions, and coordinate space range. We know of no closely related previous algorithm that we can compare performance with. See Figures 4 to 9 for results imagery. See the accompanying video for more dynamic results.

The performance is not only reliant on number of unit area's but also the complexity of unit areas, and the total number of contiguous regions. A summary is found in Figure 8. Pre-processing can require a few minutes however it's only a one-time cost.

We found that different shape files for the same region would garner inconsistent topologies, which even includes the contiguity of the unit-areas. This makes it impossible to compare our fully merged areas to already existing shape files as a way of testing the topology preserving nature of our implementation.

V. Supplementary Material

We include a variety of supplementary material including additional images, the referenced pseudo-code is also included

Shape File	Number of Areas	Total Vertices	$\frac{\text{Vertices}}{\text{Area}}$	Average FPS, m = 5%
US Counties	3,134	51,891	16.56	30
Japan	3,223	869,386	269.744	21
Italy	8,946	966,206	108.004	9
Wales	10,355	4,652,800	449.32	5
Germany	12,416	1,934,800	155.779	6
France	37,227	729,556	19.597	4

Fig. 8: The results of performance. We present some attributes of each shape file, performance times broken into separate sections of the procedure, and the average FPS. The FPS is set to a minimum required screen space of 5% for polygon rendering.

to allow the user a more fundamental understanding of the procedures we discussed This can be found at: https://bit.ly/2GGCe6v. Finally, we present a short video discussing the paper in audio-visual format which can be found at: https://vimeo.com/263507801. For the purpose of this paper, we use GADM, as well as United Stated Census Data [11], [20] to test our algorithm. In our video presentation, we use data to present the value calculation aspect of our algorithm found at the United Stated Census Data, and Office for National Statistics [12], [42].

VI. Future Work & Limitations

There are many avenues for future work. Although we use real unit-areas, we would like to test with a wider range of choropleth data. The algorithm still has performance optimizations which could accelerate the speed even further, such as schematization [5] which could be used to enable better optimization with topological continuity being reduced. Other existing formats such as TopoJSON [7] look at reducing geometry redundancy and could be a good subsequent format for the procedure. We can also continue with the idea of pre-processing by adding ways to improve performance on a second pass-through such as saving build instructions to reduce calculation of neighbor and boundaries. We worked with 2D coordinate-spaces. A 3D coordinate space would be an interesting direction to take the the algorithm and could open new applications. The algorithm potentially can apply to any data-sets with geometric boundaries and is open to new data-structures. We can also test the usability by providing user studies on the minimum perceivable screen space using the algorithm.

VII. Conclusion

We introduce a novel method of smooth and continuous zooming by exploiting a hierarchical data structure to merge areas based on their sizes and shared boundary. The shared boundary is found by first comparing the vertex list of two neighboring areas and finding the longest vertex chain between common vertices. We then render only the perceivable areas or area clusters based on the current zoom level and screen space. This method of rendering improves perceptability whilst still providing an understanding of the underlying data without distorting the map. This enables the user to zoom without any distortion to the geometry and enables clear perceivable choropleth data for the user.

VIII. Acknowledgements

We would like to thank KESS for contributing funding towards this endeavor. Knowledge Economy Skills Scholarships (KESS) is a pan-Wales higher level skills initiative led by Bangor University on behalf of the HE sector in Wales. It is partially funded by the Welsh Government's European Social Fund (ESF) convergence programme for West Wales and the Valleys. We also thank GoFore UK for contributing funds to this endeavor. We thank Dylan Rees for proofreading the paper. We also thank Thomas Basketter for testing content comprehension. We thank Amy Mizen for giving us domain expert feedback and advice.

References

[1] *Use of Address Coding Guides in Geographic Coding*. United States Department of Commerce, November 1970.

[2] N. Andrienko and G. Andrienko, *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer Science & Business Media, 2006.

[3] L. Anselin, I. Syabri, and Y. Kho, "Geoda: an introduction to spatial data analysis," *Geographical analysis*, vol. 38, no. 1, pp. 5–22, 2006.

[4] E. Axelsson, J. Costa, C. Silva, C. Emmart, A. Bock, and A. Ynnerman, "Dynamic scene graph: Enabling scaling, positioning, and navigation in the universe," in *Computer Graphics Forum*, vol. 36, no. 3. Wiley Online Library, 2017, pp. 459–468.

[5] T. Barkowsky, L. Latecki, and K. Richter, "Schematizing maps: Simplification of geographic shape by discrete curve evolution," *Spatial Cognition II*, pp. 41–53, 2000.

[6] R. Blanch and E. Lecolinet, "Browsing zoomable treemaps: Structure-aware multi-scale navigation techniques," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1248–1253, 2007.

[7] M. Bostock and C. Metcalf, "Topojson," 2018, . [Online]. Available: https://github.com/topojson/

[8] B. Braden, "The surveyor's area formula," *The College Mathematics Journal*, vol. 17, no. 4, pp. 326–337, 1986. [Online]. Available: http://www.jstor.org/stable/2686282

[9] C. A. Brewer, "Colorbrewer," 2017, accessed 2017/08/10. [Online]. Available: http://colorbrewer2.org/

[10] C. A. Brewer and L. Pickle, "Evaluation of methods for classifying epidemiological data on choropleth maps in series," *Annals of the Association of American Geographers*, vol. 92, no. 4, pp. 662–681, 2002.

[11] U. C. Bureau, "Cartographic boundary shapefiles - us counties," 2018, date Accessed: 2018-03-21. [Online]. Available: http://www.census.gov/geo/maps-data/data/cbf/cbf_counties.html

[12] ——, "Cartographic boundary shapefiles - us counties," 2018, date Accessed: 2018-03-21. [Online]. Available: https://www.census.gov/support/USACdataDownloads.html

[13] A. Cockburn, A. K. Karlson, and B. B. Bederson, "A review of overview+ detail, zooming, and focus+ context interfaces." *ACM Comput. Surv.*, vol. 41, no. 1, pp. 2–1, 2008.

[14] G. Dang, C. North, and B. Shneiderman, "Dynamic queries and brushing on choropleth maps," in *Information Visualisation, 2001. Proceedings. Fifth International Conference on*. IEEE, 2001, pp. 757–764.

[15] D. Dorling, "From computer cartographyto spatial visualization: A new cartogram algorithm," *Proceedings of Auto-Carto*, vol. 11, pp. 208–217, 1993.

[16] J. Dykes and C. Brunsdon, "Geographically weighted visualization: interactive graphics for scale-varying exploratory analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1161–1168, 2007.

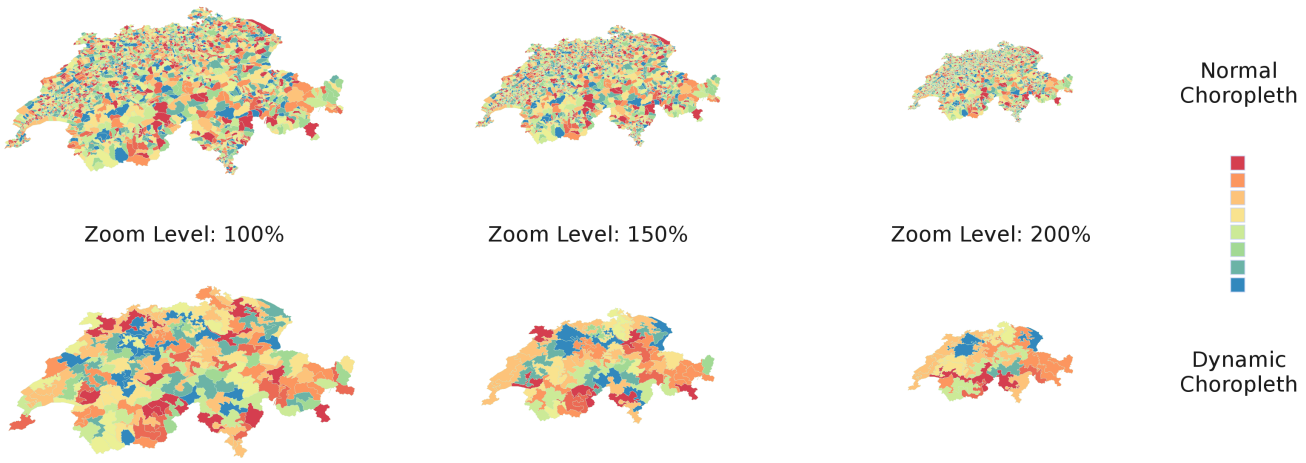


Fig. 9: An example of zooming out of Switzerland's administrative units where $m = 1\%$.

[17] G. Ellis and A. Dix, "A taxonomy of clutter reduction for information visualisation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1216–1223, 2007.

[18] N. Elmqvist and J. D. Fekete, "Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, pp. 439–454, May 2010.

[19] C. Ericson, *Real-time collision detection*. CRC Press, 2004.

[20] GADM, "Global administrative areas," 2017, date Accessed: 2017-10-15. [Online]. Available: http://www.gadm.org/country

[21] M. T. Gastner and M. E. Newman, "Diffusion-based method for producing density-equalizing maps," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 20, pp. 7499–7504, 2004.

[22] Google, "Google maps," 2017. [Online]. Available: https://www.google.co.uk/maps

[23] R. Hooda, *Statistics for business and economics*. Vikas Publishing House, 1994.

[24] W. Javed, S. Ghani, and N. Elmqvist, "Polyzoom: multiscale and multifocus exploration in 2d visual spaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 287–296.

[25] M. Jern, J. Rogstadius, and T. Åström, "Treemaps and choropleth maps applied to regional hierarchical statistical data," in *Information Visualisation, 2009 13th International Conference*. IEEE, 2009, pp. 403–410.

[26] N. K. Jog and B. Shneiderman, "Starfield visualization with interactive smooth zooming," in *Visual Database Systems 3*. Springer, 1995, pp. 3–14.

[27] J. Johansson, T. Opach, E. Glaas, T.-S. Neset, C. Navarra, B.-O. Linnér, and J. K. Rød, "Visadapt: A visualization tool to support climate change adaptation," *IEEE computer graphics and applications*, vol. 37, no. 2, pp. 54–65, 2017.

[28] S. Kaspar, S. Fabrikant, and P. Freckmann, "Empirical study of cartograms," in *25th International Cartographic Conference*, vol. 3, 2011, p. 5.

[29] H. Li and J. Han, "Discovery of population distribution knowledge visually through lorenz curve and choropleth map," in *Information Science and Engineering (ICISE), 2010 2nd International Conference on*. IEEE, 2010, pp. 3657–3660.

[30] Z. Li, H. Yan, T. Ai, and J. Chen, "Automated building generalization based on urban morphology and gestalt theory," *International Journal of Geographical Information Science*, vol. 18, no. 5, pp. 513–534, 2004. [Online]. Available: https://doi.org/10.1080/13658810410001702021

[31] R. B. McMaster and K. S. Shea, "Generalization in digital cartography." Association of American Geographers Washington, DC, 1992.

[32] L. McNabb and R. S. Laramee, "Survey of surveys (sos)-mapping the landscape of survey papers in information visualization," in *Computer Graphics Forum*, vol. 36, no. 3. Wiley Online Library, 2017, pp. 589–617.

[33] I. Meirelles, *Design for information: an introduction to the histories, theories, and best practices behind effective information visualizations*. Rockport publishers, 2013.

[34] S. Openshaw, "The modifiable areal unit problem," *Concepts and techniques in modern geography*, vol. 38, 1984.

[35] L. W. Pickle, "Usability testing of map designs," in *Proceedings of Symposium on the Interface of Computing Science and Statistics*, 2003, pp. 42–56.

[36] R. Rao and S. K. Card, "The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1994, pp. 318–322.

[37] N. Regnauld and P. Revell, "Automatic amalgamation of buildings for producing ordnance survey 1:50 000 scale maps," *The Cartographic Journal*, vol. 44, no. 3, pp. 239–250, 2007. [Online]. Available: https://doi.org/10.1179/000870407X241782

[38] K. A. Rittschof and R. W. Kulhavy, "Learning and remembering from thematic maps of familiar regions," *Educational Technology Research and Development*, vol. 46, no. 1, pp. 19–38, Mar 1998. [Online]. Available: https://doi.org/10.1007/BF02299827

[39] R. C. Roberts, C. Tong, R. S. Laramee, G. A. Smith, P. Brookes, and T. D'Cruze, "Interactive Analytical Treemaps for Visualisation of Call Centre Data," in *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*, G. Pintore and F. Stanco, Eds. The Eurographics Association, 2016.

[40] A. Robinson, R. Sale, and J. Morrison, *Elements of Cartography*. Wiley, 1978. [Online]. Available: https://books.google.co.uk/books?id=QknctEDueRcC

[41] T. A. Slocum, R. B. McMaster, F. C. Kessler, and H. H. Howard, *Thematic cartography and geovisualization*. Pearson Prentice Hall Upper Saddle River, NJ, 2009.

[42] O. F. N. Statistics, "Lower super output area mid-year population estimates," 2018, date Accessed: 2018-03-21. [Online]. Available: https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates/datasets/lowersuperoutputareamidyearpopulationestimates

[43] S. S. Stevens and E. H. Galanter, "Ratio scales and category scales for a dozen perceptual continua." *Journal of experimental psychology*, vol. 54, no. 6, p. 377, 1957.

[44] H. Sun and Z. Li, "Effectiveness of cartogram for the representation of spatial data," *The Cartographic Journal*, vol. 47, no. 1, pp. 12–21, 2010.

[45] G. D. Team, *GDAL - Geospatial Data Abstraction Library, Version 2.2.2*, http://www.gdal.org/, Open Source Geospatial Foundation, 2017.

[46] W. R. Tobler, "Choropleth maps without class intervals?" *Geographical analysis*, vol. 5, no. 3, pp. 262–265, 1973.

[47] J. J. Van Wijk and W. A. Nuij, "Smooth and efficient zooming and panning," in *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*. IEEE, 2003, pp. 15–23.

[48] K. Verbeek *et al.*, "Necklace maps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 881–889, 2010.

[49] D. Vickers and P. Rees, "Creating the uk national statistics 2001 output area classification," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 170, no. 2, pp. 379–403, 2007.
[50] U. G. Wales, "Wales output area boundaries," accessed: 2017-08-20. [Online]. Available: https://data.gov.uk/dataset/output-areas-oa-boundaries
[51] M. O. Ward, G. Grinstein, and D. Keim, *Interactive data visualization: foundations, techniques, and applications*. CRC Press, 2010.
[52] J. J. V. Wijk and W. A. Nuij, "A model for smooth viewing and navigation of large 2d information spaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 4, pp. 447–458, 2004.
[53] J. Zhang, A. Malik, B. Ahlbrand, N. Elmqvist, R. Maciejewski, and D. S. Ebert, "Topogroups: Context-preserving visual illustration of multi-scale spatial aggregates," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017, pp. 2940–2951.
[54] Y. Zhang and R. Maciejewski, "Quantifying the visual impact of classification boundaries in choropleth maps," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 371–380, 2017.