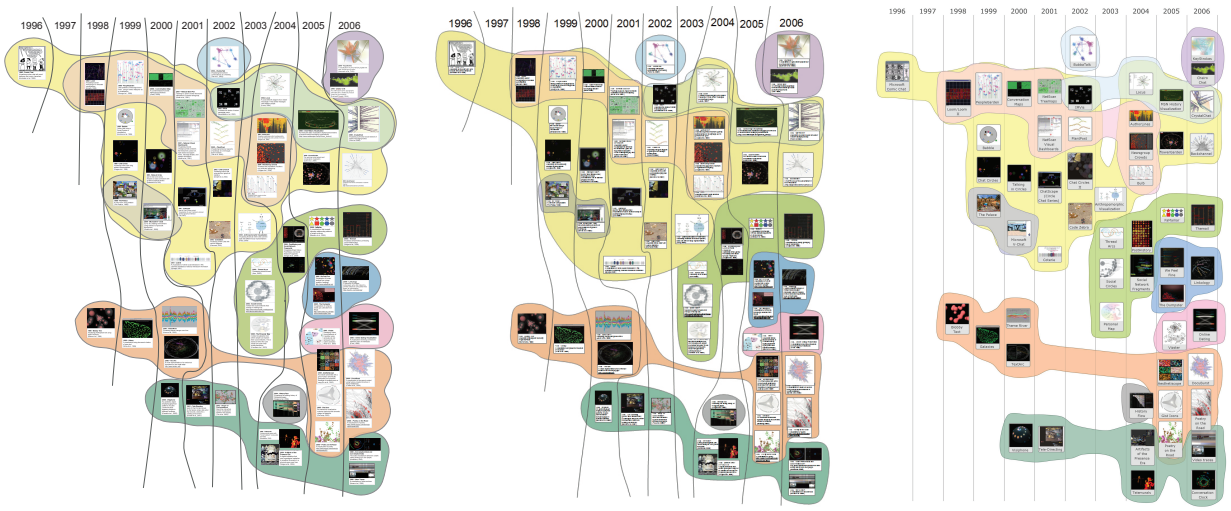
F2-Bubbles: Faithful Bubble Set Construction and Flexible Editing

Yunhai Wang, Da Cheng, Zhirui Wang, Jian Zhang,
Liang Zhou, Gaoqi He, Oliver Deussen



(a) F2-Bubbles (b) Manually Made (c) Bubble Sets

Fig. 1. Using F2-Bubbles, a set overlay visualization can be generated automatically (a) that is comparable to manually made bubble sets (b). In comparison, the visualization of Bubble Sets (c) does not retain a similar style. Note that the curved vertical lines in (a) are manually added, (b,c) are obtained from [10].

Abstract—In this paper, we propose F2-Bubbles, a set overlay visualization technique that addresses overlapping artifacts and supports interactive editing with intelligent suggestions. The core of our method is a new, efficient set overlay construction algorithm that approximates the optimal set overlay by considering set elements and their non-set neighbors. Thanks to the efficiency of the algorithm, interactive editing is achieved, and with intelligent suggestions, users can easily and flexibly edit visualizations through direct manipulations with local adaptations. A quantitative comparison with state-of-the-art set visualization techniques and case studies demonstrate the effectiveness of our method and suggests that F2-Bubbles is a helpful technique for set visualization.

Index Terms—Set visualization, Edge Crossing, Minimal Spanning Tree

1 Introduction

Set visualizations—depicting how elements belong to sets—are often used as overlays on top of other visualizations, since they help to understand set relationships in context. Even though in our case elements have to be drawn at fixed positions, creating faithful visualizations with minimal overlap and edge crossings that accurately depict set membership while being aesthetic is still a challenge [3]. In this paper, we propose F2-Bubbles—a bubble set construction method that creates set visualizations with minimal overlaps close to artistic renditions. Due to its efficiency, the visualization can be edited interactively, on a global scale and for local adaptations.

Our approach was inspired by Bubble Sets [10], a well-received set visualization technique. This method finds a minimum spanning tree for the elements of each set, computes an energy field for each of these trees and derives isocontours from the fields. All isocontours are then blended together to create the final visualization. Some drawbacks come with this elegant technique: it generates overlapping artifacts and areas that do not belong to any set overlap; results become sometimes cluttered due to wide expansions of isocontours; due to its computational requirements the method does not support direct editing; thus, generating a desired look by parameter tuning is difficult.

A different form of set overlay techniques are line-based approaches, such as LineSets [1]. The method uses curves to connect set elements to yield minimal overlaps between sets. Kelp diagrams [12] use circles to highlight set elements and connect them with links, which are further routed for generating aesthetically pleasing diagrams. However, the curves produced by both techniques might not be effective in conveying the sense of grouping for sets with spatially close points [25]; lines may be too thin for being perceived without zooming. KelpFusion [25], a hybrid technique, combines line-based and area-based visualization methods and tries to overcome some of the drawbacks of Bubble Sets. However, KelpFusion results in more visual clutter than LineSets and does not enable users to interactively edit results because of its slow routing.

- *Y. Wang, D. Cheng, and Z. Wang are with Shandong University. Email:* {cloudseawang,sduchd,russellwzrr}@gmail.com.
- *J. Zhang is with CNIC, CAS, China. E-mail:* zhangjian@sccas.cn.
- *L. Zhou is with National Institute of Health Data Science, Peking University, China. E-mail:* zhoul@bjmu.edu.cn.
- *G. He is with East China Normal University, China. E-mail:* gqhe@cs.ecnu.edu.cn.
- *O. Deussen is with Konstanz University, Germany. E-mail:* oliver.deussen@uni-konstanz.de.
- *Y. Wang and D. Cheng are joint first authors.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

With our method, F2-Bubbles, we try to overcome most limitations of the aforementioned systems . We start by identifying limitations of Bubble Set visualizations and formulate objective functions to reduce node overlap and edge crossing as much as possible for supporting data with elements belonging to multiple sets. Since this is an NP-hard problem [9, 18], we heuristically optimize these objective functions by finding the spanning trees of the overlay areas with a joint construction algorithm. Then we generate relation-aware energy fields that remove occlusions as much as possible while producing contours with adaptive widths to mimic human-drawn enclosures. Due to the efficient joint construction of spanning trees, our visualization is fully editable: users can interactively modify intermediate structures, e.g., nodes, spanning trees, and final contours with intuitive interactions. Compared to global parameter tuning, which impacts the whole visualization, our editing allows to modify the visualization locally and, therefore, the user to directly follow his/her intention. F2-Bubbles is available as a web-based online tool[1]. In summary, our method has the following important benefits:

- A faithful and heuristic creation method for bubble sets that heuristically minimizes node overlap and edge crossing;
- A set of intelligent interaction methods that enable users to fully edit the visualization toward desired effects; and
- A quantitative evaluation and several case studies demonstrate the effectiveness and usefulness of our method.

2 Related Work

Sets are a traditional topic in mathematics. Set theory was formulated by Cantor [8] and became a foundation of modern mathematics. By definition, a set is a collection of non-ordered, unique objects called set elements. Grouping data points to different sets based on their attributes is a basic step in data analysis, for example, grouping cars by their manufactures, or grouping wines by their colors.

Set visualization is an active research area. A survey on set visualization techniques is given by Alsallakhi et al. [3]. Different strategies are employed to visualize set memberships, including Euler and Venn diagrams [4, 27, 30, 36–38, 46], node-link diagrams [13, 28, 39], matrix-based methods [22, 26, 34], and aggregation methods [2, 17, 21]. Note that these methods are dedicated to show the relationship between elements and sets without concerning the context of the data, for example, locations of elements on a geological map or a timeline. For this purpose overlay methods have been developed, which place a set membership visualization on top of a different visualization to understand such data in context. Depending on how the overlays are represented, methods fall into three categories: area-based, line-based, and hybrid methods.

Area-based methods represent set membership with "bubbles" – contours that tightly enclose set members, whereas early attempts can only handle proximal groupings. Byelas et al. [6, 7] aim to mimic the way humans draw set enclosure by splatting skeletons. Collins et al. [10] introduce Bubble Sets that allow contiguous sets to be drawn over arbitrary layouts. The method finds a minimum-spanning tree for the elements within a set, and then computes an energy field based on the density of the elements to generate iso-contours using marching squares. c Colors and opacity are used to encode sets, which are blended together in case of overlaps. An inverse distance-based potential field as proposed by Vihrovs et al. [41] is able to reduce false overlaps in Bubble Sets. Region-based methods are well perceived when only few overlaps exist since colored regions are pronounced and members within regions can be easily tracked. Another advantage is that regions containing elements intuitively convey the set concept as they are essentially Euler diagrams. Because of these benefits, Gansner et al. [15] and Dinkla et al. [11] create appealing contours to depict node and link groups of the input graph. However, region-based methods tend to produce clutter due to the overlap between sets, which makes them ineffective for dense visualizations.

[1] https://multisetvis.github.io/

Line-based methods use lines to represent set membership and to reduce clutter. The LineSets [1] method computes a path through all members of a set using the Lin-Kernighan's travelling salesman heuristic [23] to approximately minimize path length while preserving geometric simplicity; a smooth curve is generated subsequently. The main advantage of LineSets is that line crossings only happen at common elements of sets, i.e., no false overlaps are produced. However, studies [25] show that the implied sequential order of LineSets cannot effectively convey the group structures. Kelp diagrams [12] use a circle to surround each element and connect them with a graph structure, whereas computing the graph structure is too expensive to be interactive. The major drawback of line-based methods is that lines may be too thin to be perceived for complex visualizations.

Hybrid methods take advantage of area-based and line-based techniques. KelpFusion [25] blends a area-based method (Bubble Sets [10]) and a line-based method (Kelp diagrams [12]) by using lines as well as filled regions. The method computes shortest-path graphs for the linear features and then fills faces in the graphs to create filled convex hulls. By generating the bubble contours with spatially varying widths, F2-Bubbles is also a hybrid method that smoothly combines lines and regions.

Hypergraph supports. Sets with fixed node positions can be regarded as an embedding of a *hypergraph* [3], where each set is a hyperedge connecting an arbitrary number of nodes. Most of existing set visualization techniques are based on *hypergraph supports*, which connect all elements of a set using links in terms of various criteria such as edge length and edge crossing. For example, Bubble Sets [10] need to compute a customized minimal spanning tree for each set in terms of edge length and node-edge crossing; LineSets [1] computes a single line per set in terms of the minima total edge length; Kelp diagrams [12] computes a sparse spanning graph for all sets by considering edge length, edge bend and edge crossing; and KelpFusion [25] builds on the shortest-path graphs. However, Bubble Sets, LineSets and KelpFusion all optimize edge properties by constructing the support for each set individually, which may lead to significant edge crossing between sets. Kelp diagrams jointly construct the supports for all sets based on the tangent visibility graphs [43], which is very expensive to compute. In contrast, F2-Bubbles minimizes edge length and crossings for all sets based on the complete graph of the input data in each step and yields better performance (see Section 6).

Finding an optimal support in a hypergraph is hard—in fact, the problem is NP-hard for minimal total edge length for two sets if planarity is required [9]. Hurtado et al. [18] show that length minimization is NP-hard for more than two sets [18] and provides a polynomial-time algorithm. Therefore, heuristics are used to approximate the optima by existing methods and also by F2-Bubbles. Here, spanning trees of all sets are generated jointly and the minimal solution is approximated in polynomial time.

Evaluation. A number of empirical evaluations for Bubble Sets, LineSets, Kelp diagrams, and KelpFusion are available. Alper et al. [1] compare LineSets with Bubble Sets on tasks such as identifying the number or sizes of sets, set intersections, and set memberships. They show that LineSets is faster and more accurate than Bubble Sets suggesting that less cluttered visualizations are more effective. Another study by Meulemans et al. [25] suggests that KelpFusion is never outperformed by Bubble Sets or LineSets for the same tasks, while it is aesthetically more pleasing than the other methods. However, a study on group information finds that Bubble Sets is more effective than LineSets for group tasks [19]. Last but not least, a task-based evaluation [31] of Bubble Sets, EulerView [36], KelpFusion, and LineSets, in conjunction with network visualization tasks for social network data analysis finds no significant differences between Bubble Sets, KelpFusion, and LineSets. These studies suggest that the effectiveness of these techniques is data-dependent, and accordingly, we conducted a large scale quantitative evaluation on 30 data sets by using aesthetic criteria such as overlap ratios, the number of edge crossing, edge length and number of bends.

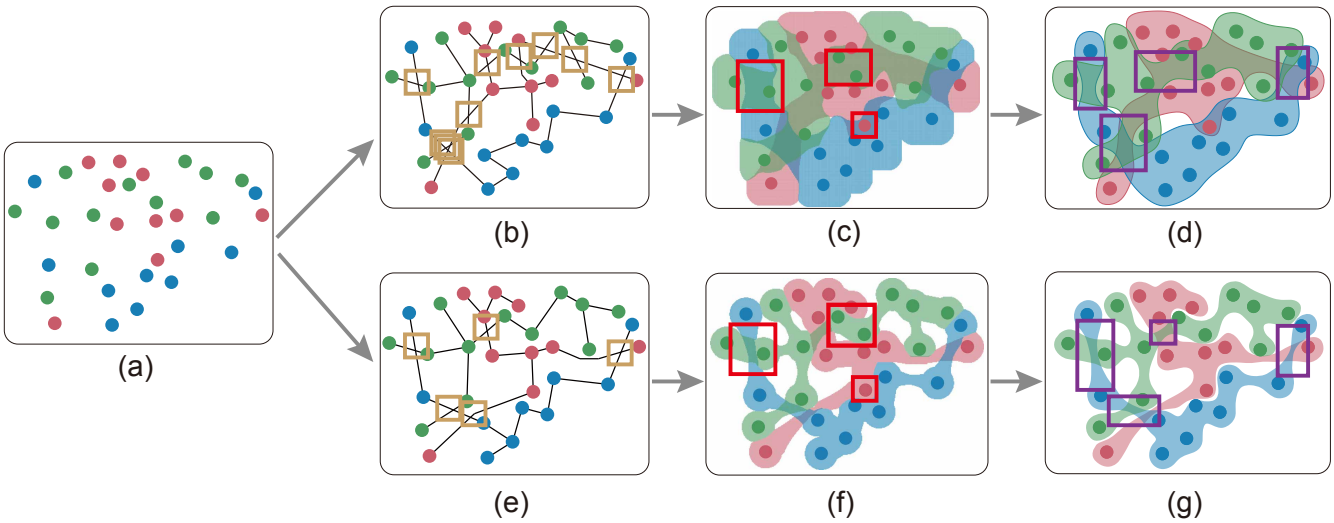


Fig. 2. Components involved in the pipelines of the Bubble Sets (top row) and our approach (bottom row) using the same input with three sets of points (a). (b,e) Spanning trees, the brown boxes indicate the crossing regions; (c,f) Energy fields, the overlapping regions in (c) misrepresent set relations (red boxes) while our method does not have such overlaps; (d,g) final contours, our method (g) exhibits waisted contours with a more pronounced separation.

Editing. Overlay techniques inherently do not support for direct manipulation of parts of the visualization and can only achieve that by re-parameterization and rerunning the entire layout algorithm, e.g., [10, 12, 25]. However, it is difficult, if not impossible, to achieve a desired and satisfactory visualization with re-parametrization, since the results might not be predictable. Such a lack of direct editing could discourage users from using them as has been pointed out for other types of visualizations such as tag-clouds [20], or edge bundling [42]. Our F2-Bubbles fills this gap by providing flexible control over set visualizations.

3 Background: Bubble Sets

Given an array of 2D points $\{\mathbf{p}_1, \cdots, \mathbf{p}_n\}$ with k sets, each point can be associated with more than a single set. Bubble Sets attempt to reveal set memberships by using continuous bounding contours with four design requirements:

(i) **R1:** all members of a set should be enclosed by a continuous and connected contour;
(ii) **R2:** all non-set members should be excluded from the contour
(iii) **R3:** visual and interactive hints should clarify membership where non-set members are enclosed; and
(iv) **R4:** flexible manipulation of the end result must be possible.

In the following, we briefly review how the algorithm of Bubble Sets [10] meets these requirements, but then discuss its limitations.

3.1 Algorithm Pipeline

Collins et al. [10] proposed an implicit surface-based algorithm, which involves three steps: spanning tree construction, energy field calculation and contour extraction. The top row in Fig. 2 illustrates the corresponding pipeline.

Spanning Tree Construction. Given the $s-$th set, all its members are first sorted by their distance to the set center defined as the mean position of all members. For each member $\mathbf{p}_i$, its optimal neighbor $\mathbf{p}_j$ is the one with the least cost, defined as:

$$\text{cost}(\mathbf{p}_i, \mathbf{p}_j) = \text{pd}(\mathbf{p}_i, \mathbf{p}_j) * \text{nn}(\mathbf{p}_i, \mathbf{p}_j)\,, \tag{1}$$

where $\text{pd}(\mathbf{p}_i, \mathbf{p}_j)$ is the Euclidean distance between $\mathbf{p}_i$ and $\mathbf{p}_j$, and $\text{nn}(\mathbf{p}_i, \mathbf{p}_j)$ is the number of non-set members that have intersecting bounding boxes with the straight edge between $\mathbf{p}_i$ and $\mathbf{p}_j$. Incorporating nn() helps for satisfying **R2** and reducing crossings between contours and non-set members. After selecting a node j from the sorted list, an edge routing algorithm [29] is applied to find the best route between these two nodes during tree construction. Note that routing might result in multiple additional virtual nodes. Finally, all nodes are connected by a spanning tree (see Fig. 2(b)).

In short, this algorithm first sorts the nodes in ascending order according to their distances to the center and then at a time adds the next node with an edge having the smallest weight. Specifically, the algorithm does not select the edge with the smallest weight from the whole graph. In doing so, a blobby shape can be ensured as demonstrated by the original authors [10].

Energy Field Calculation. Given a set of virtual edges $E = \{e_1, \cdots, e_m\}$ defined in spanning trees and two parameters R_0 and R_1, the energy value $\phi(\mathbf{q})$ at each point $\mathbf{q}$ in 2D space consists of two parts: radius-nearest nodes $\Omega_v = \{\mathbf{p}_i | \text{pd}(\mathbf{p}_i, \mathbf{q}) < R_1\}$ and edges $\Omega_e = \{e_i | \text{ld}(e_i, \mathbf{q}) < R_1\}$, and

$$\phi(\mathbf{q}) = \frac{\sum_{\mathbf{p}_i \in \Omega_v} w_i (R_1 - \text{pd}(\mathbf{p}_i, \mathbf{q}))^2 + \sum_{e_j \in \Omega_e} w_j (R_1 - \text{ld}(e_j, \mathbf{q}))^2}{(R_1 - R_0)^2}\,, \tag{2}$$

where $\text{ld}(e_i, \mathbf{q})$ is the shortest distance from $\mathbf{q}$ to the line segment of the virtual edge e_i, w_i is the weight assigned to e_i or $\mathbf{p}_j$, and R_0 and R_1 are two distances with energy values 1 and 0, respectively. In other words, a rectangular energy field is formed around each edge e_i and a circular one for each point sample $\mathbf{p}_i$. To meet **R2**, positive and negative energies are introduced by adjusting w_i, where w_i is 1 for the nodes and edges coming from the same set, and -0.8 and 0 for the nodes and edges from the other sets, respectively. Fig. 2(c) shows the energy fields of three sets, with a heavy overlap between green and red sets.

Contour Extraction. To meet **R1**, this step iteratively applies the marching squares algorithm [24] to the obtained energy field of each set with different thresholds, until the set center is enclosed by the corresponding contour. Fig. 2(d) is the contour representation of Fig. 2(c). Crossing regions have similar widths as others.

To enable interactive rendering, Collins et al. [10] suggested to compute the energy field in a low-resolution display space but this introduces visual artifacts. By doing so, it allows users to edit the end results, but only provides node-level operations, such as adding, moving, and deleting nodes.

3.2 Existing Limitations

Based on the above brief review and previous evaluations [25, 32], we believe that Bubble Sets completely meets **R1** but have three aspects that might be further improved.

First, Bubble Sets often generate heavy visual clutter and strong overlap, resulting in poor readability and misleading relations. For

example, the contours highlighted by the boxes in Fig. 2(d) make it hard to discern set relations. If there are many such areas, even using additional visual hints cannot clearly depict the underlying relations.

Second, while carefully comparing boundaries of hand-drawn enclosures and contours created by Bubble Sets (see an example in Fig. 1(b,c)) we observe two prominent differences. One is that the contour thickness of hand-drawn enclosures is not fixed but varies in terms of the number of closeby members. For regions far away from the nearest members, contours are often drawn thinner than for other parts (see orange contours in Fig. 1(c)), this results in a smooth combination of lines and regions. However, adjusting R_0 and R_1 of Bubble sets can only globally change the contour thickness. KelpFusion also attempts to meet this goal, however, the resulting boundaries are not as smooth as human-drawings (see Fig. 1(b)), typically caused by routing and smoothing. The other difference is spatial imprecision, humans tend to draw sketchy features [47], where set contours do not strictly follow the border of the boundary members, see the shape of the yellow set in Fig. 1(b). However, such a style is not easy to simulate by automatic methods. One possible solution would be through contour-level editing, which allows users to fine-tune the contour shapes.

Last, editing node positions inevitably involves re-construction of the spanning tree for the corresponding set, because the set center will be changed. Hence, a simple movement might result in completely different spanning trees and finally generate substantially different contours. Hence, such interactions contradict the consistency principle for efficient user interaction design [35].

To address these limitations, we propose two new algorithms for constructing spanning trees and for computing energy fields (Section 4), while providing flexible interactions for users to edit nodes, edges and contours of interest (Section 5).

4 Our Method

To produce faithful and aesthetic Bubble Sets, we introduce two additional design requirements:

(i) **R5:** minimizing contour crossings as much as possible; and
(ii) **R6:** smoothly combining lines and regions for achieving ink minimization [40],

while at the same time aiming for better satisfying R2 and R4. To realize this, we propose to explicitly incorporate edge crossing and node overlapping into the construction of spanning trees and energy fields. As shown at the bottom of Fig. 2(e), F2-Bubbles generates set skeletons with a joint spanning trees algorithm, which explicitly reduces edge crossings. Given the skeletons, relation-aware energy fields are generated to produce hybrid contour representations as hand-drawn enclosures, while reducing as many overlaps as possible, see Fig. 2(f).

4.1 Joint Construction of Spanning Trees

To construct Bubble Sets, a well-constructed spanning tree for each set is indispensable. To fulfill **R5**, we define the weight of the i-th edge as:

$$\omega_i = \mathrm{nec}(e_i) + w_e \mathrm{len}(e_i)\ , \tag{3}$$

where $\mathrm{len}(e_i)$ is the length of the edge e_i, $\mathrm{nec}(e_i)$ is the number of edge crossing between e_i and edges from the other sets, w_e is the weight (default $w_e = 1.0$). To reduce edge crossing, we give $\mathrm{nec}(e_i)$ more weight by normalizing the $\mathrm{len}(e_i)$ with the largest edge length in the whole graph. The set index for an edge is defined as the set index of its nodes (edges can only span between nodes of the same set). Accordingly, we find the optimal forest F with k spanning trees by minimizing the sum of edge weights

$$\min_F \sum_i \omega_i\ , \tag{4}$$

where all nodes in F are the input points $\mathbf{P}$. However, the derivation of an optimal solution of Eq. 4 is an NP-hard problem [5], because calculating $\mathrm{nec}(e_i)$ requires the forest to be available. Therefore, we propose a heuristic algorithm to approximate it, which simultaneously constructs the forest of spanning trees of all sets.

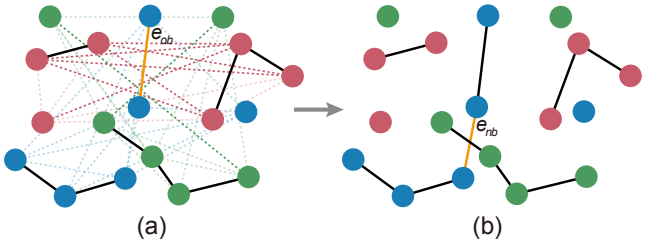


Fig. 3. Selecting a proper edge to construct a spanning tree: (a) the weight of edges that intersect the newly added orange edge are updated (highlighted dotted lines) (b) from all potential edges that connect blue dots in (a) the edge with smallest weight is added to the spanning tree.

Inspired by Kruskal's MST algorithm [44], we initialize the forest by taking each point as an individual tree and then gradually merge them by adding edges one by one. If there are edges that are pre-defined in the input data, we keep them and gradually add new edges. Before constructing the forest, we build a graph G with k complete sub-graphs for each point set in $\mathbf{P}$ and define the edge weight by using Eq. 3, where $\mathrm{nec}(e_i)$ is zero. After initialization, we first select an edge e_{ob} with the smallest weight and add it to the forest F. Then, we repeatedly merge two trees with the following three steps until k (= number of sets) trees are left:

1. update ω_i for all unselected edges in G that intersect with e_{ob},
2. select an edge e_{nb} with minimal weight from the unselected edges in G that does not form a cycle with the edges in F; and
3. add e_{nb} to F and apply the surface routing method proposed by Bubble Sets [10] to the edge e_{nb} if it overlaps with non-set points and set $e_{ob} = e_{nb}$.

Note that only the weights of edges that cross a newly added edge e_{ob} among the to-be-selected edges can change. Therefore, in the update phase, only the number of crossings of these edges needs to be changed. Fig. 3 illustrates these two steps within one iteration.

Algorithm 1 Joint Construction of Spanning Trees

Require: an array of points $\mathbf{P}$ with k sets
Ensure: a forest F with k spanning trees
1: initialize a forest $F = \{\mathbb{E}, \mathbb{V}\}$, where $\mathbb{E} = \emptyset$ and $V = \mathbf{P}$
2: construct a graph G with k complete sub-graphs based on $\mathbf{P}$
3: calculate edge weights in G with Eq. 3
4: find the edge e_{ob} with the minimum weight in G
5: **repeat**
6: update the weights of unselected edges in G with e_{ob}
7: add the edge e_{nb} with the minimum weight and set $e_{ob} = e_{nb}$
8: run surface routing algorithm for the new edge
9: **until** F has only k trees

The most expensive part of Algorithm 1 is to calculate edge weights in line 3 , which involves counting the number of edge crossing in m edges with the time complexity $O(m^2)$. In the worst case, m is kn^2 for each of k overlapped sets with n nodes and thus the time complexity is $O(k^2n^4)$. Since the number of sets k is typically small, the time complexity is further reduced to $O(n^4)$. More details are provided in the supplemental material. Nonetheless, it took less than 1s for 200 points with 6 classes in our experiments. Figs 2(b,e) compares spanning trees constructed by Bubble Sets and our method for three sets, where the numbers of edge crossing in both trees are 10 for BubbleSets and 5 for our method. Since our method is an edge-based solution, it inherently supports edge-level editing (see Section 5.2)

4.2 Relation-Aware Energy Fields

Similar to Bubble Sets, our energy field computation also takes all nodes and edges in each spanning tree as items. To meet **R2** and **R6**, we propose to create energy fields with adaptive radius R_1 and refine

their faithfulness and aesthetics by using two strategies: membership correction and completeness refinement.

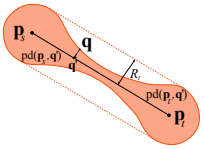
Adaptive Radius. According to the "minimal ink" principle [40], "arms" connecting two areal regions in the isocontour should not be too wide—this inspired the design of Kelp diagrams [12] and KelpFusion [25]. This is also in line with hand-made set overlays (e.g., Fig. 1(b)). Unlike Kelp diagrams and KelpFusion, arms in manually made visualizations typically have smooth transitions between their endpoints. We would like to resemble such a look in F2-Bubbles: arms should exhibit a smooth transition from a large diameter around endpoints to the thinnest in between.

For an edge e_i with two nodes $\mathbf{p}_s$ and $\mathbf{p}_t$, we determine the adaptive radius for any point $\mathbf{q}$ in 2D space by:

$$R(\mathbf{q}) = \frac{R_1}{1 + w_r \cdot f(\mathbf{q}, \mathbf{p}_s, \mathbf{p}_t)}, \quad (5)$$

$$\text{with } f(\mathbf{q}, \mathbf{p}_s, \mathbf{p}_t) = \frac{\min\left(\text{pd}(\mathbf{p}_s, \mathbf{q}'),\ \text{pd}(\mathbf{p}_t, \mathbf{q}')\right)}{\text{pd}(\mathbf{p}_s, \mathbf{p}_t)},$$

where $\mathbf{q}'$ is the projected point (see the inset) of $\mathbf{q}$ on the edge between $\mathbf{p}_s$ and $\mathbf{p}_t$, and w_r is a weight with the default set to 3.



Doing so, the width of the contour at different points on an edge is inversely proportional to the distance from the position to the closer endpoint, which lets the energy field gradually attenuate from the endpoint to the midpoint of the edge (see the inset). Note that we only apply this adaptive contour to the edges whose length is larger than a threshold. Incorporating the adaptive radius into Eq. 2, we can obtain energy fields with spatial varying widths (see Fig. 2(f)).

Unlike Bubble Sets, we only compute the positive energy generated by each item in the set, where w_i in Eq. 2 is 0 for the nodes and edges from the other sets. However, the resulting energy fields still have occlusions between sets and thus we have to further refine them to better respect set relations.

Membership Correction. The pixel-based representation of the energy fields consists of item-based pixels and non-item-based pixels, where items refer to the nodes or edges in the spanning trees. To meet **R1** and **R2**, an item should always be covered by the energy field of its own set, and non-set items should be excluded from this energy field. However, there are two cases that might result in membership ambiguities: i) edges of different sets could cross each other; and ii) the energy fields of different sets could overlap. Therefore, individually computing the energy field of each set is infeasible. An example of the first case is shown in the red box in Fig. 4(a). To address this issue, we loosen the aforementioned restrictions and only ensure that set items are always included in their energy field so that the generated bubble set always contains all elements and its connectivity is ensured. For the second case, the middle area between two or more items might have non-zero energy values for each individual energy field, resulting in membership uncertainty, see the yellow box in Fig. 4(a). Accordingly, we aim to determine certain memberships for such pixels, similar to probability-based image segmentation [16].

Based on the computed k energy fields, we adjust the energy value of each pixel. The procedure is summarized in Algorithm 2 with time complexity of $O(HW)$ (for a field of $H \times W$ pixels). Every set is processed successively. If pixel $\mathbf{x}$ is covered by an item of the current set s, we set its energy value to the value of its energy field $E_s(\mathbf{x})$, otherwise, we set its value to 0. If the pixel $\mathbf{x}$ is not covered by any item but other energy fields, we set its energy value to $E_s(\mathbf{x})$, if the value of $E_s(\mathbf{x})$ is greater than the values of the energy fields of all other sets; otherwise to zero. Fig. 4(b) shows the energy fields after the correction, where only one edge intersection remains in the red box and the boundary between the blue and green sets in the orange box is clearly located. In doing so, the occlusions among different sets are minimized as much as possible.

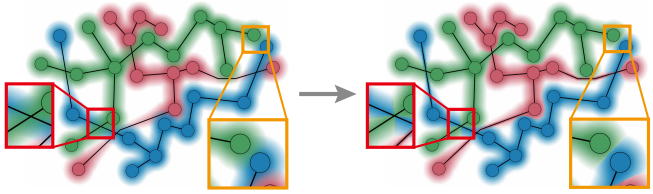


Fig. 4. Energy fields before and after membership correction. (a) The edge-edge overlap and node-node overlap are highlighted by red and orange boxes in (a), respectively; (b) the energy field after applying membership correction.

Algorithm 2 Membership Correction

given the initial energy field E_s for each set $s \in S$
for each set $s \in S$ **do**
  initialize E'_s with a copy of E_s
  for each pixel $\mathbf{x} \in E'_s$ **do**
    if $\mathbf{x}$ is covered by an item of s **then**
      $E'_s(\mathbf{x}) = E_s(\mathbf{x})$
    else if p is covered by an item of another set **then**
      $E'_s(\mathbf{x}) = 0$
    else
      if $\forall i \in S$ and $i \neq s$, $E_s(\mathbf{x}) > E_i(\mathbf{x})$ **then**
        $E'_s(\mathbf{x}) = E_s(\mathbf{x})$
      else $E'_s(\mathbf{x}) = 0$
return E'_s for each set $s \in S$

Continuity Refinement. The membership correction ensures faithfulness, but it might result in two problematic cases: isolated small regions and discontinuities between components, see examples in Fig. 5. Small regions result from crossings between a node and an edge from another set, while discontinuities result from crossings of two edges from different sets. Therefore, we further have to refine the energy fields to improve the aesthetic appearance of the final bubble set.

To solve the first case, we find the connected components of the energy field of each set and remove any connected component that does not belong to any item. Fig. 5(a) shows an example, the region below the horizontal edge of the red set is to be removed since it does not belong to any component of the green set.

To solve the second case, the membership correction created overlap reduced energy fields around regions of edge crossings that appear to be hourglass-shaped (see Fig. 5(b)). Such shapes weaken the continuity of contours; therefore, we define a rectangular energy field centered at the crossing point with the width R_1 (see the dash rectangles in Fig. 5(b)) and fill it with the original energy values. To prevent false memberships, we ensure that the filling is only done for the area without non-set members. Doing so, the final contours of this region will not result in false memberships.

Using these refined energy fields, we generate smooth bubbles by extracting isocontours with marching squares and then fitting the contours with B-Splines. As shown in Fig. 1(a), such smooth and connected contours correctly reveal set relations, while resembling hand-drawn enclosures.

Node duplication. Our method can work on elements contained in multiple sets. This is achieved by duplicating the node $ts-1$ times if it belongs to ts different sets so that each duplicated node is contained by one set. Once it is done, we compute an energy field for each set.

5 Flexible Editing

The efficiency of the above algorithms allows us to provide new or refined types of interactions enabling users to create visually pleasing Bubble Sets. During editing, the user has full control over set points, intermediate structures (spanning trees) and the resulting contours with flexible yet easy-to-use interactions. To make the interaction more effective, F2-Bubbles provides suggestions to users as guidance, while locally updating spanning trees and energy fields for consistent

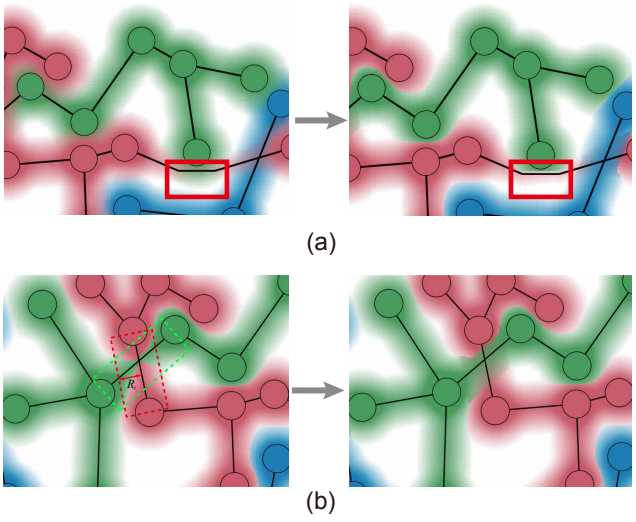


Fig. 5. Continuity refinement for membership correction: (a) introducing artificial gaps for corrected edge-node regions; (b) two dashed rectangles defined for filling the gaps introduced by corrected edge-edge crossings.

editing [35].

5.1 Node-Level Editing

Similar to Bubble Sets, users are allowed to add new points to existing sets or newly created sets, and move and delete existing points.

Adding Points. For a new point added to one of existing sets, we update the complete graph G by connecting the point to all points in the corresponding set and compute the weights for all new edges. Then, we find the edge with the smallest cost starting from this point (see Eq. 3) and add it to the forest F. If the point is inserted into a new set, we wait for one more point to compute its Bubble representation.

Deleting Points. When the user deletes a point, all edges in F related to this point will be removed, the corresponding spanning tree becomes a few disconnected trees. We take these trees as the input of our joint construction of spanning trees and compute a new forest F while keeping all existing edges.

Moving Points. Users can move a point to any position in 2D space. Doing so, the spanning trees should be updated to keep them optimal. However, doing the spanning tree construction from scratch might generate completely different tress, which is typically not desirable since small changes of nodes are expected to lead to small changes in the resulting trees (see Fig. 6(b)).

Thanks to our edge-based construction of the spanning trees, we can heuristically optimize the trees by only modifying the affected parts so as to minimize the overall changes. Specifically, we update the forest F for a moved point in three steps. First, we delete all edges connected to the point at the original position, which disassembles the corresponding spanning tree into a few connected components. Then, we select a region Θ with a radius of 10% of the overall visualization around the new position and remove all edges from the same set in this region. Last, we take the remaining connected components and the points in Θ as inputs of our joint construction algorithm, which computes a new spanning tree while keeping all existing edges.

When the user moves a node to a new position, the method will connect it to a nearby point, at the same time, only the local neighborhood part is changed during re-optimization of the spanning tree. The effect of moving a node is shown in Fig. 6. Compared to Bubble Sets, the induced change of the spanning tree with F2-Bubbles is much smaller. Because this local editing relies on the current status of the forest F, moving a node that is not newly added to a new position and then back to its old position might not yield the same forest.

5.2 Edge-Level Editing

Like node-level editing, users can add and delete edges to change the connectivity between elements. They can also adjust edge routing by

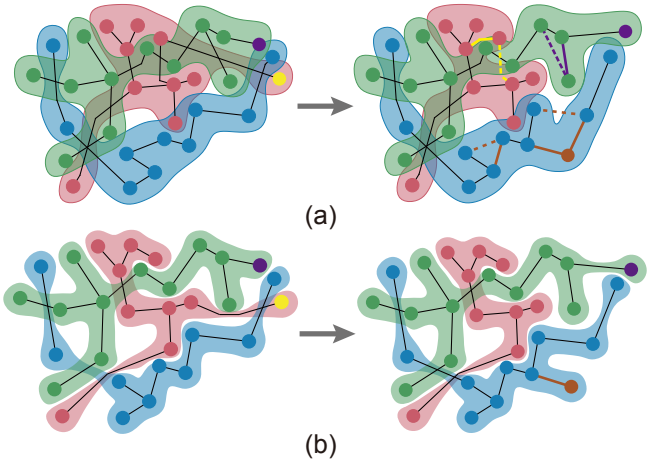


Fig. 6. Node-level editing operations: moving the purple node, deleting the yellow node and adding a brown node with Bubble Sets (a) and F2-Bubbles (b). Bubble Sets removes four unrelated edges (shown as dotted lines) and adds four edges in (a), while our method preserves the input spanning tree in (b).

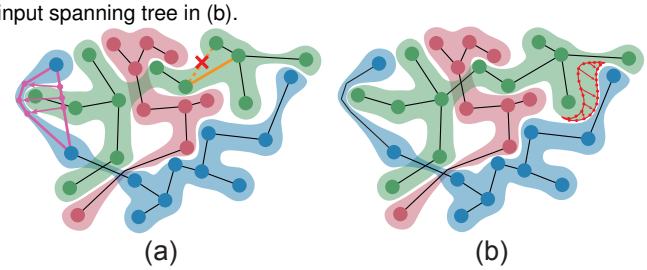


Fig. 7. Editing contours with edge- and contour-level editing operations: (a) The solid orange edge is a user-added edge, while the dotted orange edge is suggested for deleting because of a cycle. The dotted pink edge on the left is moved by adjusting its control points; (b) contour control points are moved to change its boundary.

adding control points to one of existing edges or delete them.

Adding Edges. Users can add an edge by connecting two selected points from the same set. This, however, might form a loop in the spanning tree. To address this issue, we first detect whether there is a cycle, calculate the costs of the loop-involved edges, and suggest the one with the highest cost for deletion. Once an edge is added, we apply surface routing to let the edge bypass all data points from the other sets. To always keep user-added edges during the tree construction, we set their costs to -1.

Deleting Edges. After deleting an edge, the corresponding spanning tree becomes disconnected, and another edge needs to be added elsewhere. We re-calculate the edge weights for all edges that could be added in the current state and suggest the edge with the lowest cost for adding. In Fig. 7(a), the dotted orange edge is deleted by the user and the solid orange line is suggested for adding. If the user is not satisfied with the suggested edge, the system will suggest the edge with the second lowest cost until the user selects one. For the selected edge, we apply surface routing to bypass any obstacles. To ensure that deleted edges will not be re-created later again, we set their costs to $+\infty$.

Editing Control Points. Users can select an edge and then add control points by clicking a position along the edge and dragging it to a desired new position. Likewise, users can also delete control points from a selected edge. This form of editing with bend edges allows to form a desired look for the entire visualization. An example is the leftmost edge in Fig. 7(a), which are routed to resolve the edge crossing by moving its control points towards the left border.

5.3 Contour-Level Editing

We allow users to directly interact with the final generated contours to change their shape. In addition, the width of specific "arms" within the

isocontours can be modified interactively, which is not possible with Bubble Sets.

Contour Spline Editing. After running marching squares, B-spline interpolation is used to smooth the contours. To change the contour, we allow users to freely edit its control points. When users move a control point outwards of the existing contour, we fill the newly formed area with the energy value at the original position to extend the energy field. If the control point is moved inwards, we fill the removed regions with zero energy. Fig. 7(b) shows local contour editing of Fig. 7 (a) by dragging control points.

Local Radius Editing. Adjusting parameters R_1 and R_0 in Equation 2 can change the amount of energy generated by an edge, and modifies the isocontour thereof. However, adjusting these values changes the energy fields globally, whereas the user may only want to change the contour width near a certain edge in the spanning tree. To do so, we allow users to adjust w_r (see Eq. 5) to increase or decrease the radius of the selected edges.

With the synergy of the aforementioned interactions, the user is able to conveniently achieve visualizations that are comparable to manually made set overlays within a short time. Examples can be seen in Fig. 1 and Fig. 9 in the case studies.

6 Evaluation

We performed a quantitative evaluation of F2-Bubbles against Bubble Sets [10], Kelp diagrams [12], KelpFusion [25], and LineSets [1] to evaluate our automatic adaptive construction method. The comparison against Bubble Sets is done in terms of four measures: the number of edge crossings, the total edge length, the number of bends, and the ratio of overlapping areas to all enclosed regions. The first three measures evaluate the quality of the generated supports, while the last one assesses the faithfulness of the Bubble contours. Overlap ratios of KelpFusion and Line Sets are better than what Bubble Sets that can produce due to their minimal "ink" strategy. We formulate the following hypothesis:

- F2-Bubbles has a lower overlap ratio than Bubble Sets and fewer edge crossings, a smaller total edge length and lowest number of edge bends than all other methods.

Since the code of Kelp diagrams and KelpFusion is not available from the authors nor online, we implemented F2-Bubbles, Line Sets (based on an open-source Java code [45]), Kelp diagrams and KelpFusion in JavaScript, and used a publicly available Java package of Bubble Sets from the original authors for the evaluation. A total of 30 datasets were used in the evaluation: 10 of which are real-world datasets, and 20 are synthetic datasets generated with randomly placed set elements. The number of points in all datasets are less than 200. Default parameter settings of our method, Bubble Sets, and Line Sets were used, no editing was done. Kelp diagrams and KelpFusion do not have a default configuration, so we empirically set the parameters $b_t = 2$, $c_d = 1$, $c_\alpha = 100$, $c_l = 100$ for Kelp diagrams to generate proper links and $t = 3$ for KelpFusion and node radius $r = 15$ for both of them to achieve a "medium" effect that balances linear and areal regions. A comprehensive documentation of the evaluation with visualizations of all datasets can be found in the supplemental material, while a summary statistics of the results on all datasets and real-world datasets are shown as the red and blue boxplots in Fig. 8.

Fig. 8(a) shows that the number of edge crossing in spanning trees with F2-Bubbles is about 1/2 of that of Bubble Sets, Kelp diagrams, and Line Sets, while it is 1/3 of KelpFusion. Since the objective of Kelp diagrams considers edge crossing, it performs slightly better the other methods but still worse than our method. Fig. 8(b) shows that the total edge length in spanning trees with F2-Bubbles is less than about 10% to 25% of that of the other methods. Since the objective of Line Sets only considers edge length, it performs slightly worse that our method. Fig. 8(c) shows that KelpFusion results in the largest number of bends and LineSets performs the best, while our method is comparable to Bubble Sets, and is less than 10% to 25% of that of Kelp Diagrams. In contrast, the overlap ratio with F2-Bubbles is typically 1/3 to 1/5 of Bubble Sets as shown in Fig. 8(d). Comparing the red and blue boxplots

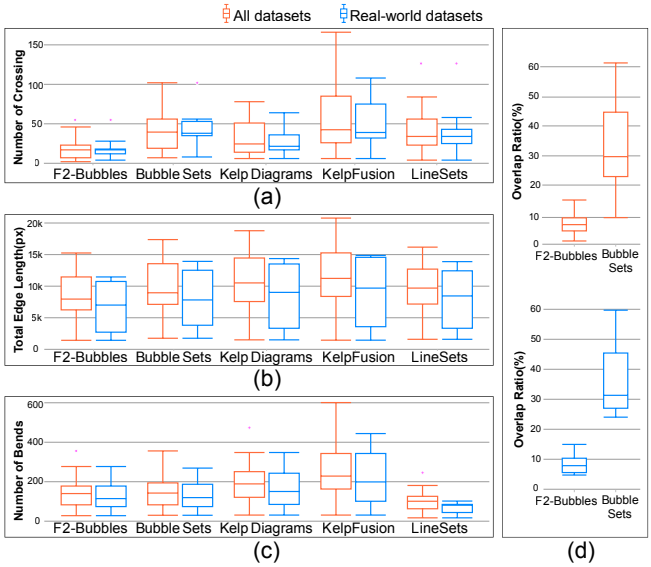


Fig. 8. Performance evaluation for set overlay techniques on all datasets (red boxplots), and on real-world datasets (blue boxplots) with four different measures: numbers of edge crossings (a), total edge lengths(b), numbers of bends(c), and the overlap ratios(d). Outliers are shown as dots in the boxplots.

indicates that the distributions of the first two measures for all methods on real-world datasets is more concentrated, while F2-Bubbles and Bubble Sets both result in a slightly larger overlap ratios on real-world datasets. On the other hand, F2-Bubbles resulted distributions of three measures are less spread than that of others.

We then used statistical inference to check if significant differences exist between our method and the other approaches. The Wilcoxon signed-rank test—a non-parametric method—was used for testing statistically significant differences. Such differences were found between F2-Bubbles and Bubble Sets on the overlap ratio ($p < 0.0001$), the number of edge crossings as well as the total edge length between F2-Bubbles and Bubble Sets, Kelp diagrams, KelpFusion, and Line Sets. Tests on real-world datasets show that our hypothesis still holds ($p < 0.05$). For the number of bends, our method is significant lower than Kelp diagrams and KelpFusion and is significantly higher than LineSets, but its difference from Bubble Sets is not significant. Namely, F2-Bubbles performs only worse than Line Sets among all methods. Therefore, our hypothesis can be partially accepted.

Moreover, the computation of our method is fast (within 2 seconds for all data) and comparable to that of Bubble Sets and Line Sets. In comparison, KelpFusion typically takes hundreds of seconds (Median = 450.3s, Min = 3.4s, Max = 2804.6s) to generate a result, while Kelp diagrams is even slower. We speculate that it is caused by the expensive computation ofthe shortest paths from the tangent visibility graphs [43], which are based on Voronoi diagrams of the input data. The evaluation provides evidence that F2-Bubbles is able to effectively and efficiently construct set overlays automatically. The construction method of F2-Bubbles generates fewer edge crossings and overlaps in comparison to existing techniques, which provides a good basis for further interactive editing (see Section 7).

It should be noted that using fixed parameters with KelpDiagrams and KelpFusion might not generate good results for every dataset and thus we cannot conclude that they always perform poorly in terms of our used measures. To obtain a more complete view, we show the results yielded by different parameters in the supplementary material. Moreover, our implementations of KelpDiagrams, KelpFusion and LineSets are not optimized and thus running times could be further reduced.

7 Case Studies

To demonstrate the usefulness of our overall method—the set overlay construction and the flexible, intelligent editing thereafter— we

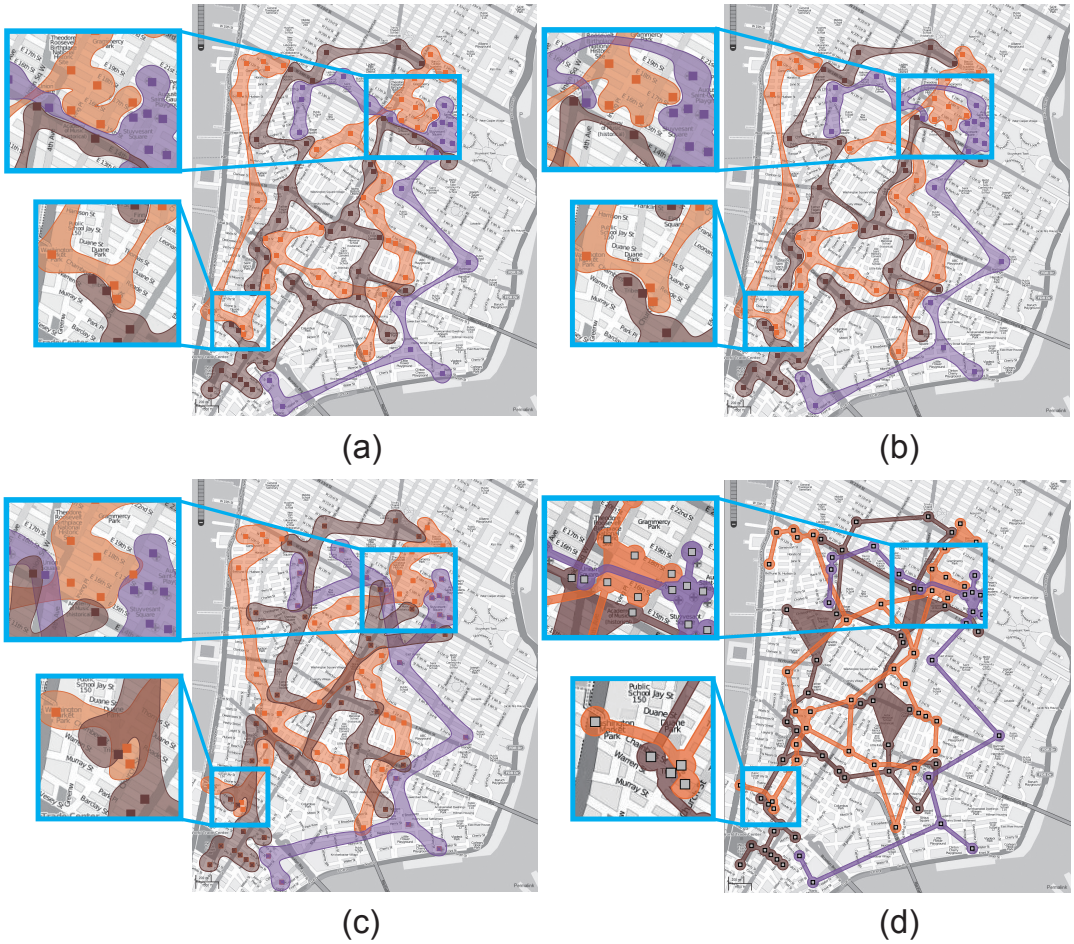


Fig. 9. Set visualization of (a) F2-Bubbles right after adaptive construction, (b) after editing, (c) Bubble Sets [10], and (d) KelpFusion [25]. All sets are visualized over the map of Manhattan (New York). Hotels, subway stations, and clinics are visualized as bubble sets in orange, brown, and purple.

created set overlays on top of a wide range of data types, including a geographical map, scatterplots, and a timeline.

Geographical Map—Manhattan. Sets of hotels (orange), subway stations (brown), and clinics (purple) in Manhattan (New York) are visualized in Fig. 9. For Bubble Sets (Fig. 9(c)) the overlaps between "bubbles" make tracking set elements difficult—users have to zoom in and out to correctly identify and track the set of interest which might break their mental map. For example, the brown set overlaps with the orange set around Tribeca, and a brown element is close to an orange element on the boundary of the brown set (the bottom zoom-in); a similar case is seen for the brown point in the overlap of the brown and purple sets around the Union Square (the top zoom-in). Moreover, overlaps may be misleading to users as they may suggest false set relationships, e.g., blended regions of set contours and the underlying map could be misinterpreted as new, isolated sets. KelpFusion (Fig. 9 (d)) clearly separates each set with the line and polygon-based rendering, and the memberships of set elements in the overlapped area of Bubble Sets are clearly seen (top zoom-ins). However, the lines of KelpFusion in Fig. 9 (d) share a similar style with the subway lines of the underlying map, which might might confuse viewers, especially on colored maps. Moreover, polygonal areas may mislead people to suggest more set elements since they are pronounced perceptually (for example, the brown areas). Note that such effect might not exist when using different t values. By contrast, F2-Bubbles creates a clutter-reduced visualization with an automatic adaptive construction (Fig. 9 (a)), no ambiguity is present for the problematic regions of Bubble Sets (both zoom-ins of Figs. 9(a, c)). However, F2-Bubbles might yield contours that are too thin to be identified, which might result in membership mis-estimations. The part of the purple contour between the brown and orange sets in the top zoom-in of Fig. 9(a) is an example. To address this issue, we delete the corresponding edge in the forest and add a new edge as well as control points to form new contours. After interactive editing, a further fine-tuned visualization is achieved (see Fig. 9 (b)). Tracking of each set is easy as few overlaps occur, set information can be perceived without zooming-in, and, therefore, a good focus-and-context effect is achieved without a high mental load of the user. In addition, our version looks more pleasing than Bubble Sets due to the smoother long arms and less "ink"; our result looks more faithful than KelpFusion and is less likely to be misinterpreted as part of the map.

Furthermore, the Bubble Sets and KelpFusion cannot be edited directly, whereas our result in Fig. 9 (a) was interactively edited for the regions around the Union Square with edge editing (the red box), and close to Soho with isocontour editing (the blue box).

Scatterplots. Fig. 10 shows set visualizations over 2D scatterplots for total fertility (horizontal axis) and life expectancy (vertical axis) of countries and regions during the year 1981 from Gapminder [33]. The sets are color-coded based on geographic regions. The Bubble Sets visualization in Fig. 10 (a) is cluttered due to many overlaps. This is especially the case for high-density regions (e.g., the red and gray sets are almost fully covered by others). Moreover, overlaps are misleading and cause confusions, in particular, when close-by data points are from different sets (see zoom-ins). For example, elements of gray and blue sets are close-by (the central zoom-in) and their set contours overlap. Thus, users are typically unsure if these are from the data or caused by a bug; moreover, blending generates new colors close to set colors (the brown points on the boundary of the red contour, seen in the top left zoom-in) making a correct perception difficult. In comparison, the automatic adaptive construction of F2-Bubbles (Fig. 10 (b)) already creates a much clearer visualization with no severe overlaps. However, there are still edge crossings in the overlapping area (see the blue zoom-in in Fig. 10 (b)). To alleviate such crossings, we route the gray contour by deleting its original edge and add a new curved edge. The same holds

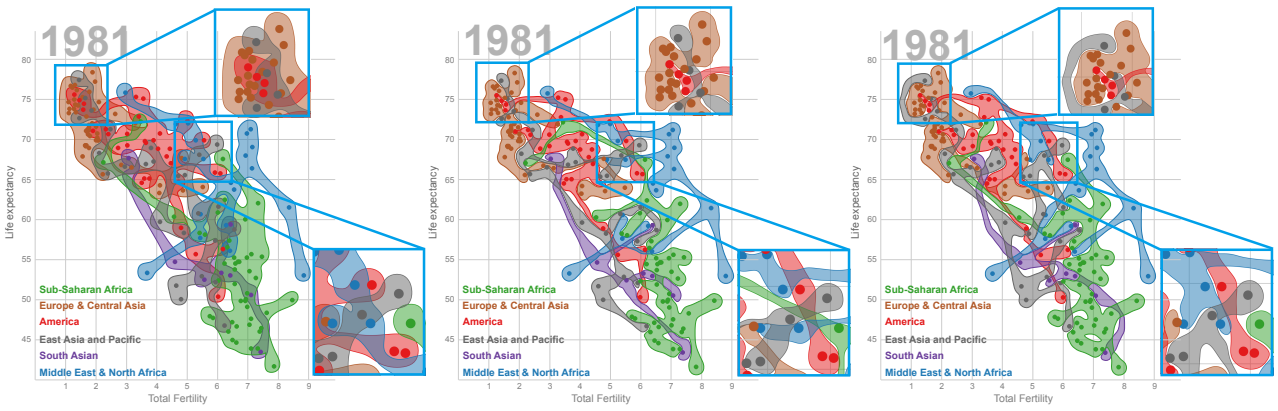


Fig. 10. Set visualization of (a) Bubble Sets, (b) F2-Bubbles, and (c) F2-Bubbles after editing 2D scatterplots of life expectancy versus total fertility of countries and regions of the year 1981. Sets are colored based on geographic regions of the world as shown in the legend.

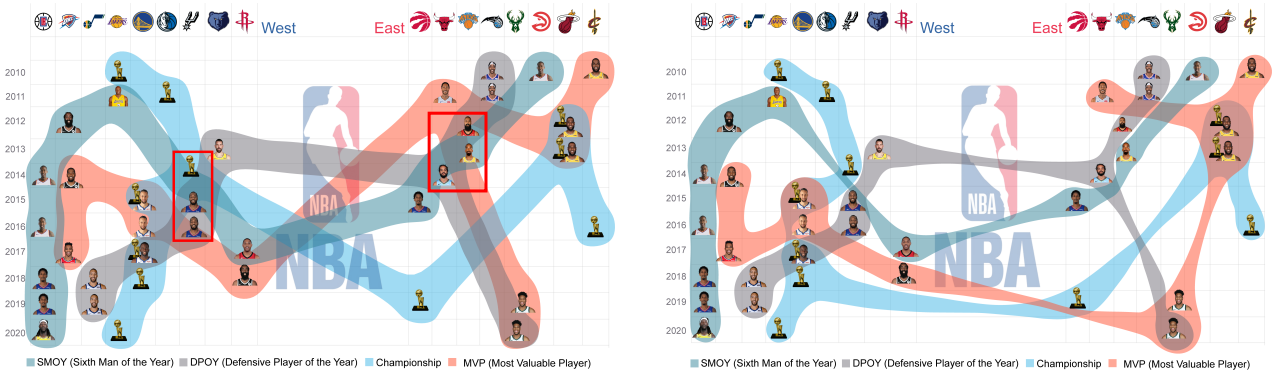


Fig. 11. Visualizations of NBA (National Basketball Association) season awards for the years 2010 to 2020 with (a) Bubble Sets, and (b) F2-Bubbles. The horizontal axis of the timeline shows the teams in which the awarded players served, while the vertical axis is the year of the season. Players are indicated by their photos, the winner teams are shown with winner cups. Sets are color-coded by the award type as shown in the legend.

for the contours in the red zoom-in in Fig. 10 (b). After fine-tuning with intelligent suggestions (Fig. 10 (c)), even better visualization is achieved with ease: optimal edge configurations are suggested after deleting unwanted edges, and control points are added to avoid dense areas.

Timeline—NBA Awards. Fig. 11 shows set overlays of NBA season awards on top of a timeline visualization of years 2010 to 2020 (the vertical axis) versus teams of awarded players (the horizontal axis). Players are shown with their photos, and the winner teams are tagged with winner cups. The sets are color-coded by the type of awards as indicated in the legend. The Bubble Sets visualization (Fig. 11 (a)) conveys the overall set information but generates many overlaps that hinder the understanding of set relationships. In fact, as photos, instead of colored set elements are used in this example, it is impossible to identify which award a player received when his photo is shown in overlapped regions (highlighted in red boxes). Such ambiguities can only be resolved with background knowledge, which reduces the value of the visualization. With F2-Bubbles (Fig. 11 (b)), the visualization is clear and without any ambiguity—sets are clearly separated in the erroneous regions in Bubble Sets (regions within red boxes). The visualization of F2-Bubbles is more accessible and easier to understand than Bubble Sets, and, therefore, can be readily used as visualization for the public, e.g., as an infographic in a news report.

8 Conclusion & Future Work

In this paper, we have introduced F2-Bubbles for the generation of faithful set overlays with interactive editing. We have designed our method to improve faithfulness and enable users to directly edit the visualization with intuitive interactions. The generation of F2-Bubbles has been modeled as an optimization problem, which is approximated by our pipeline consisting of the joint production of spanning trees, structure-aware energy fields, and isocontour smoothing. Intelligent interactions allow users to edit visualizations with guided direct manipulation towards desired effects. The effectiveness of our method was evaluated by comparing it to existing methods on various datasets with quantitative comparisons. Through case studies, we demonstrated the capability of F2-Bubbles in achieving artistically-made set overlays within several interactions.

However, our method still has several limitations. First, our adaptive radius based thinning strategy may prevent users from noticing cluster patterns or areas of the underlaying data that are largely covered by a single set. An example is the purple contour in top zoom-in of Fig. 9(a), which can be improved by further interactive edting. In the future, we would like to automatically detect such cases and adapt surface routing to avoid them. Second, our results seem to be more naturally looking than other methods, whereas the perceived quality is not explicitly measured. Hence, we would like to conduct a user study to learn how well F2-Bubbles works for different set-related tasks and how much they are preferred by users. Last, our method still takes around one minute for data with more than 1000 points, which results in a latency for further interactive editing. In the future, we will accelerate F2-Bubbles by exploring advanced MST optimization methods [14] and using the GPU to further reducing construction time.

Acknowledgments

This work is supported by the grants of the National Key Research & Development Plan of China (2019YFB1704201), NSFC (61772315, 61861136012), the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (No.VRLAB2020C08), CAS grant (GJHZ1862), Beijing Advanced Discipline Construction Project (BMU2019GJJXK001), PKU-Baidu Fund (2019BD017), and Deutsche Forschungsgemeinschaft (DFG) Project DE 620/26-1.

REFERENCES

[1] B. Alper, N. Riche, G. Ramos, and M. Czerwinski. Design Study of LineSets, a Novel Set Visualization Technique. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2259–2267, 2011. doi: 10.1109/TVCG.2011.186

[2] B. Alsallakh, W. Aigner, S. Miksch, and H. Hauser. Radial Sets: Interactive Visual Analysis of Large Overlapping Sets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2496–2505, 2013. doi: 10.1109/TVCG.2013.184

[3] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers. The State-of-the-Art of Set Visualization. *Computer Graphics Forum*, 35(1):234–260, 2016. doi: 10.1111/cgf.12722

[4] M. E. Baron. A Note on the Historical Development of Logic Diagrams: Leibniz, Euler and Venn. *The Mathematical Gazette*, 53(384):113–125, 1969. doi: 10.2307/3614533

[5] B. Bollobas. *Graph Theory: An Introductory Course*. Springer-Verlag, New York, 1979. doi: 10.1007/978-1-4612-9967-7

[6] H. Byelas and A. Telea. Visualization of Areas of Interest in Software Architecture Diagrams. In *Proceedings of the ACM Symposium on Software Visualization*, pp. 105–114, 2006. doi: 10.1145/1148493.1148509

[7] H. Byelas and A. Telea. Towards realism in drawing areas of interest on architecture diagrams. *Journal of Visual Languages Computing*, 20(2):110–128, 2009. doi: 10.1016/j.jvlc.2008.09.001

[8] G. Cantor. Beiträge zur Begründung der transfiniten Mengenlehre. *Mathematische Annalen*, 46(4):481–512, 1895. doi: 10.1007/BF02124929

[9] T. Castermans, M. van Garderen, W. Meulemans, M. Nöllenburg, and X. Yuan. Short plane supports for spatial hypergraphs. In *International Symposium on Graph Drawing and Network Visualization*, pp. 53–66. Springer, 2018. doi: 10.1007/978-3-030-04414-5_4

[10] C. Collins, G. Penn, and S. Carpendale. Bubble Sets: Revealing Set Relations with Isocontours over Existing Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009. doi: 10.1109/TVCG.2009.122

[11] K. Dinkla, N. H. Riche, and M. A. Westenberg. Dual adjacency matrix: exploring link groups in dense networks. 34(3):311–320, 2015. doi: 10.1111/cgf.12643

[12] K. Dinkla, M. J. van Kreveld, B. Speckmann, and M. A. Westenberg. Kelp Diagrams: Point Set Membership Visualization. *Computer Graphics Forum*, 31(3pt1):875–884, 2012. doi: 10.1111/j.1467-8659.2012.03080.x

[13] M. Dörk, N. Henry Riche, G. Ramos, and S. Dumais. PivotPaths: Strolling through Faceted Information Spaces. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2709–2718, 2012. doi: 10.1109/TVCG.2012.252

[14] R. Dorrigiv, R. Fraser, M. He, S. Kamali, A. Kawamura, A. López-Ortiz, and D. Seco. On minimum-and maximum-weight minimum spanning trees with neighborhoods. *Theory of Computing Systems*, 56(1):220–250, 2015. doi: 10.1007/s00224-014-9591-3

[15] E. R. Gansner, Y. Hu, and S. G. Kobourov. Gmap: Drawing graphs as maps. In *International Symposium on Graph Drawing*, pp. 405–407. Springer, 2009. doi: 10.1007/978-3-642-11805-0_38

[16] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985. doi: 10.1016/S0734-189X(85)90153-7

[17] H. Hofmann, A. P. J. M. Siebes, and A. F. X. Wilhelm. Visualizing Association Rules with Interactive Mosaic Plots. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 227–235, 2000. doi: 10.1145/347090.347133

[18] F. Hurtado, M. Korman, M. van Kreveld, M. Löffler, V. Sacristán, A. Shioura, R. I. Silveira, B. Speckmann, and T. Tokuyama. Colored spanning graphs for set visualization. *Computational Geometry*, 68:262–276, 2018. doi: 10.1016/j.comgeo.2017.06.006

[19] R. Jianu, A. Rusu, Y. Hu, and D. Taggart. How to Display Group Information on Node-Link Diagrams: An Evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 20(11):1530–1541, 2014. doi: 10.1109/TVCG.2014.2315995

[20] K. Koh, B. Lee, B. Kim, and J. Seo. ManiWordle: Providing Flexible Control over Wordle. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1190–1197, 2010. doi: 10.1109/TVCG.2010.175

[21] R. Kosara, F. Bendix, and H. Hauser. Parallel Sets: interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, 2006. doi: 10.1109/TVCG.2006.76

[22] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister. UpSet: Visualization of Intersecting Sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1983–1992, 2014. doi: 10.1109/TVCG.2014.2346248

[23] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973. doi: 10.1287/opre.21.2.498

[24] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. vol. 21, pp. 163–169. ACM New York, NY, USA, 1987. doi: 10.1145/37401.37422

[25] W. Meulemans, N. H. Riche, B. Speckmann, B. Alper, and T. Dwyer. KelpFusion: A Hybrid Set Visualization Technique. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1846–1858, 2013. doi: 10.1109/TVCG.2013.76

[26] L. Micallef, P. Dragicevic, and J. Fekete. Assessing the Effect of Visualizations on Bayesian Reasoning through Crowdsourcing. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2536–2545, 2012. doi: 10.1109/TVCG.2012.199

[27] L. Micallef and P. Rodgers. eulerAPE: Drawing Area-Proportional 3-Venn Diagrams Using Ellipses. *PLOS ONE*, 9(7):1–18, 2014. doi: 10.1371/journal.pone.0101717

[28] K. Misue. Drawing Bipartite Graphs as Anchored Maps. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60*, pp. 169–177, 2006. doi: 10.1145/1151903.1151929

[29] D. Phan, L. Xiao, R. Yeh, and P. Hanrahan. Flow map layout. In *Proc. IEEE Symposium on Information Visualization.*, pp. 219–224. IEEE, 2005. doi: 10.1109/INFVIS.2005.1532150

[30] P. Rodgers, J. Flower, G. Stapleton, and J. Howse. Drawing Area-Proportional Venn-3 Diagrams with Convex Polygons. In A. K. Goel, M. Jamnik, and N. H. Narayanan, eds., *Diagrammatic Representation and Inference*, pp. 54–68. Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-14600-8_9

[31] P. Rodgers, G. Stapleton, B. Alsallakh, L. Michallef, R. Baker, and S. Thompson. A task-based evaluation of combined set and network visualization. *Information Sciences*, 367-368:58–79, 2016. doi: 10.1016/j.ins.2016.05.045

[32] P. Rodgers, G. Stapleton, B. Alsallakh, L. Michallef, R. Baker, and S. Thompson. A task-based evaluation of combined set and network visualization. *Information Sciences*, 367:58–79, 2016. doi: 10.1016/j.ins.2016.05.045

[33] H. Rosling, R. A. Rosling, and O. Rosling. New software brings statistics beyond the eye. *Statistics, knowledge and policy: Key indicators to inform decision making*, pp. 522–530, 2005.

[34] R. Sadana, T. Major, A. Dove, and J. Stasko. OnSet: A Visualization Technique for Large-scale Binary Set Data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1993–2002, 2014. doi: 10.1109/TVCG.2014.2346249

[35] B. Shneiderman. *Designing the user interface: strategies for effective human-computer interaction*. Pearson Education India, 2010. doi: 10.1007/BF01934418

[36] P. Simonetto, D. Auber, and D. Archambault. Fully Automatic Visualisation of Overlapping Sets. *Computer Graphics Forum*, 28(3):967–974, 2009. doi: 10.1111/j.1467-8659.2009.01452.x

[37] G. Stapleton, J. Flower, P. Rodgers, and J. Howse. Automatically drawing Euler diagrams with circles. *Journal of Visual Languages Computing*, 23(3):163–193, 2012. doi: 10.1016/j.jvlc.2012.02.001

[38] G. Stapleton, P. Rodgers, J. Howse, and L. Zhang. Inductively Generating Euler Diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 17(1):88–100, 2011. doi: 10.1109/TVCG.2010.28

[39] J. Stasko, C. Görg, and Z. Liu. Jigsaw: Supporting Investigative Analysis through Interactive Visualization. *Information Visualization*, 7(2):118–132, 2008. doi: 10.1057/palgrave.ivs.9500180

[40] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 2 ed., 2002. doi: 10.1075/idj.4.3.12cos

[41] J. Vihrovs, K. Prūsis, K. Freivalds, P. Ručevskis, and V. Krebs. An inverse distance-based potential field function for overlapping point set visualization. In *2014 International Conference on Information Visualization Theory and Applications (IVAPP)*, pp. 29–38, 2014.

[42] Y. Wang, M. Xue, Y. Wang, X. Yan, B. Chen, C. W. Fu, and C. Hurter. Interactive Structure-aware Blending of Diverse Edge Bundling Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):687–696, 2020. doi: 10.1109/TVCG.2019.2934805

[43] R. Wein, J. P. Van den Berg, and D. Halperin. The visibility–voronoi

complex and its applications. *Computational Geometry*, 36(1):66–87, 2007. doi: 10.1016/j.comgeo.2005.11.007

[44] M. A. Weiss. *Data structures and algorithm analysis*. Benjamin-Cummings Publishing Co., Inc., 1995.

[45] D. Welch. An implementation of linesets in processing/java. `https://github.com/dtwelch/Line-Sets`. [Online; accessed 12-Jul.-2021].

[46] L. Wilkinson. Exact and Approximate Area-Proportional Circular Venn and Euler Diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 18(2):321–331, 2012. doi: 10.1109/TVCG.2011.56

[47] J. Wood, P. Isenberg, T. Isenberg, J. Dykes, N. Boukhelifa, and A. Slingsby. Sketchy rendering for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2749–2758, 2012. doi: 10.1109/TVCG.2012.262