KelpFusion : a hybrid set visualization technique

KelpFusion: a Hybrid Set Visualization Technique

Wouter Meulemans, Nathalie Henry Riche, Bettina Speckmann, Basak Alper, and Tim Dwyer

Abstract—We present *KelpFusion*: a method for depicting set membership of items on a map or other visualization using continuous boundaries. KelpFusion is a hybrid representation that bridges hull techniques such as Bubble Sets and Euler Diagrams and line- and graph-based techniques such as LineSets and Kelp Diagrams. We describe an algorithm based on shortest-path graphs to compute KelpFusion visualizations. Based on a single parameter, the shortest-path graph varies from the minimal spanning tree to the convex hull of a point set. Shortest-path graphs aim to capture the shape of a point set and smoothly adapt to sets of varying densities. KelpFusion fills enclosed faces based on a set of simple legibility rules. We present the results of a controlled experiment comparing KelpFusion to Bubble Sets and LineSets. We conclude that KelpFusion outperforms Bubble Sets both in accuracy and completion time, and outperforms LineSets in completion time.

Index Terms—Information visualization, visualization techniques and methodologies.

Teaser Figure: KelpFusion applied to restaurants in Boston (left) and to cities in Europe (right).

1 Introduction

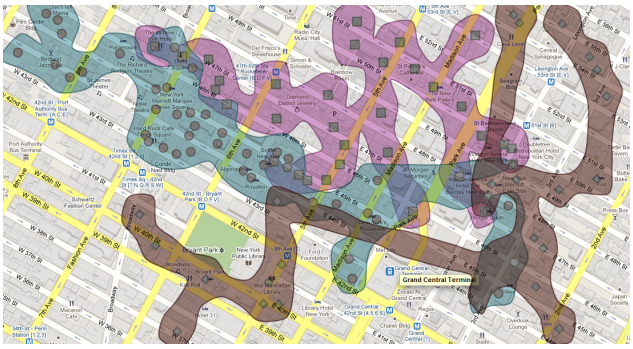
Exploring large information spaces often requires understanding the grouping of elements by their properties. For example, social scientists tackle the analysis of large social networks by grouping individual people into communities and then studying how these groups interact. Similar analysis of groups or sets is a key task in many domains, from identifying related words in linguistics to studying relations between geographic places. Visually representing sets and their elements can lead analysts to effectively identify properties of an element and its relationships to those around it.

There are many ways to represent sets and their elements. Among the oldest representations—and probably most familiar for many people—are Venn diagrams [9] and their generalization, Euler Diagrams. Such visual representations naturally and effectively convey how a small number of sets intersect through simple overlapping regions. When dealing with a larger number of sets and more intricate intersections, using simple shapes such as ellipses is no-longer possible. Common solutions propose to use more complex shape boundaries to enclose set elements, for example, Bubble Sets [7]. However, when representing set membership of places marked on a map, or other visualization where there is no freedom to rearrange the elements, such techniques generate a lot of occlusion. Recent solutions have proposed more minimal enclosing geometries. LineSets [1] reduce the geometry to a single continuous line. Kelp Diagrams [8] use a sparse spanning graph, essentially a minimal spanning tree with some carefully chosen additional edges. Both LineSets and Kelp Diagrams attempt to reduce visual clutter and clarify intersecting regions in different ways. However, when a set contains elements that are spatially close, it may become more effective to generate an enclosing geometry to better convey a sense of grouping.

In this paper we attempt to bridge the two ends of the spectrum, providing a hybrid technique named KelpFusion that uses a mix of hulls and lines and generates fitted boundaries for groups of elements in a given arrangement. Such a hybrid technique is interesting to explore since it falls between two desirable extremes: a minimal spanning tree guarantees to use minimal "ink" (after Tufte's rule [17]) to show a connected boundary for the set, while the convex hull best displays cohesive grouping according to Gestalt theory [19].

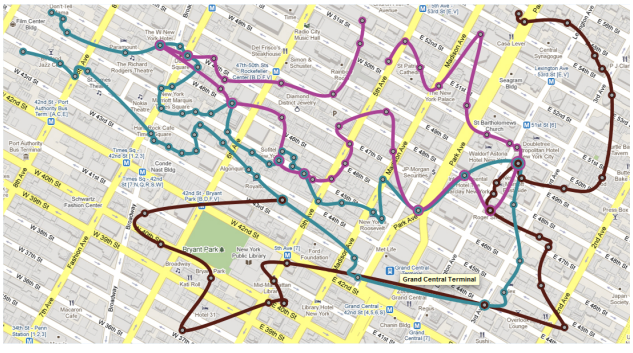
While Kelp Diagrams and KelpFusion are both based

- *Wouter Meulemans is with the TU Eindhoven, E-mail: w.meulemans@tue.nl*
- *Nathalie Henry Riche is with Microsoft Research, E-mail: Nathalie.Henry@microsoft.com*
- *Bettina Speckmann is with the TU Eindhoven, E-mail: speckman@win.tue.nl*
- *Basak Alper is with University of California Santa Barbara, E-mail: basakalper@gmail.com*
- *Tim Dwyer is with Monash University, E-mail: tim.dwyer@monash.edu*

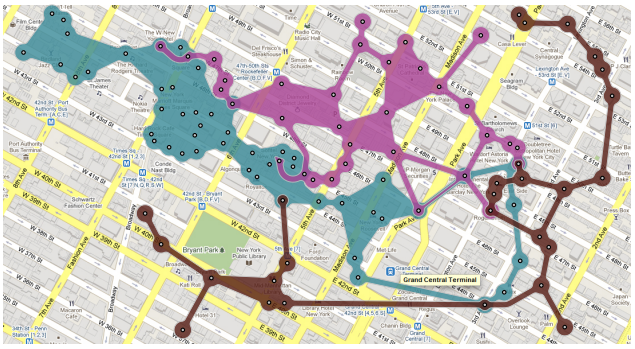


(a) Bubble Sets

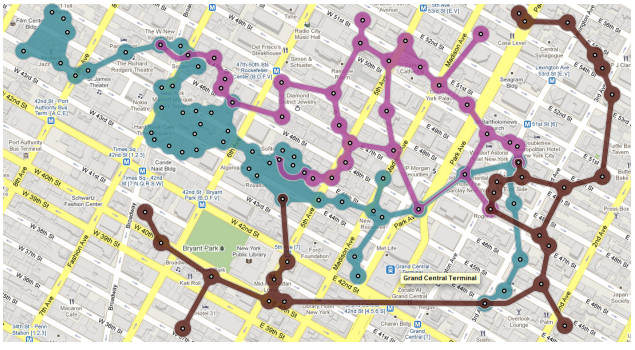
(b) Kelp Diagrams



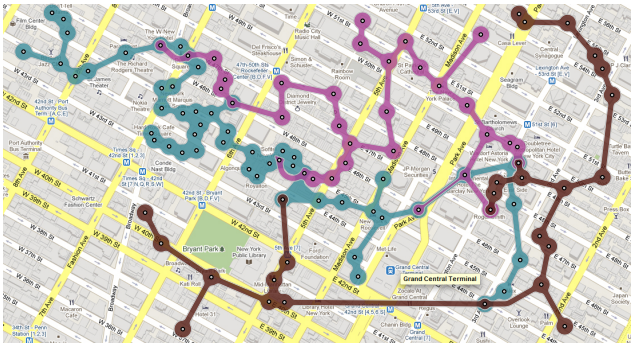
(c) LineSets



(d) KelpFusion (dense)



(e) KelpFusion (medium)



(f) KelpFusion (sparse)

Figure 1. Visualizations using the various methods discussed in this paper. (a) Image generated using the implementation generously provided by the authors of Bubble Sets [7]. (b) Image courtesy of Kasper Dinkla. (c) Image generated using the LineSets implementation described in [1]. (d-f) Images generated by our KelpFusion implementation.

on a spanning graph, KelpFusion introduces the use of a proximity graph, a so-called *shortest-path graph*. In the context of Geographic Information Science, shortest-path graphs have been used to delineate imprecise regions, reconstructing a boundary of a region based on points that are likely inside the intended region [2]. Shortest-path graphs adapt to point sets of varying density and aim to capture the shape and clusters of a point set. In other words, the use of shortest-path graphs allows KelpFusion to fill faces when points are spatially close. Furthermore, we show that the shortest-path graph and its corresponding boundary can be computed efficiently, enabling interactive manipulation of the visualization. Figure 1 illustrates three existing methods, Bubble Sets, LineSets, and Kelp Diagrams, in comparison with our new hybrid technique, KelpFusion.

To understand the advantages and drawbacks of our technique, we performed a controlled experiment with 13 participants, comparing KelpFusion to Bubble Sets [7] and LineSets [1]. We discovered that KelpFusion improved on Bubble Sets, outperforming the technique in accuracy and completion time. We also found that KelpFusion was on par with LineSets in terms of accuracy but yielded faster response times. User preferences and comments also indicated that KelpFusion provides a good sense of grouping and is aesthetically more pleasing than the other methods.

2 Related Work

Venn or Euler diagrams are popular ways to visually represent set intersections. In these diagrams, closed curves correspond to sets and overlaps between the curves indicate intersections. Several papers have explored the problem of automatically drawing Euler diagrams to convey abstract set topology, for example, Simonetto and Auber [14] and Stapleton *et al.* [16]. Other approaches investigated the

possibility of adding labels or glyphs to Euler diagrams to represent individual set members, i.e. data items contained in the set. As discussed by Simonetto *et al.* [15], this requires ensuring that the set regions—and the areas of intersection between them—are large enough to enclose such labels. Readability of highly overlapped regions quickly becomes a concern. In a controlled study Henry Riche and Dwyer [10] found that it is beneficial to show intersections using simple set regions and strict containment, enabled by duplication or splitting of sets.

The methods above all assume that set regions and their members can be freely placed anywhere in the plane. An important variation is the problem of overlaying a given arrangement of points or labels representing data elements with indicators of their set membership. Collins *et al.* [7] presented Bubble Sets, a method based on isocontours to overlay such an arrangement with enclosing set regions. A similar approach was suggested by Byelas and Telea [6]. Again, readability of regions in these methods suffers when they are highly overlapping.

A recent technique called LineSets [1] attempts to improve the readability of complex set intersections and to minimize the overall visual clutter by reducing set regions to simple curved lines drawn through set elements. Using a single continuous line induces a sequential order in which to browse elements of a set, conveying a different concept of grouping. In a method that falls stylistically somewhere between lines [1] and the strict nested containment of [10], Dinkla *et al.* [8] introduced so-called *Kelp Diagrams*. Kelp Diagrams incorporate classical graph-drawing "bubble and stick" style graph or tree spanners over the member points in a set. When multiple sets are shown simultaneously, overlapping regions are drawn with strictly nested containment.

These set visualization techniques focus on representing sets of elements with fixed locations, as does the KelpFusion technique we present in this paper. While preserving the spatial context of the data is essential in many analysis scenarios, it also significantly limits the scalability of these methods. As the number of sets and the complexity of their intersections increases, one should consider designing a dedicated set representation that relaxes the spatial constraint on their elements. One possibility is permutation matrices [13].

Sets defined over points in the plane can be interpreted as an embedding of a *hypergraph* where the points are vertices and each set is a hyperedge connecting an arbitrary number of vertices. Then a Kelp Diagram is a realization of a *hypergraph support*, as discussed by Kaufmann *et al.* [12], Buchin *et al.* [5], and Brandes *et al.* [3]. Though the concepts explored in these papers consider structures similar to Kelp Diagrams and LineSets, they are more concerned with their graph theoretical properties, such as existence theorems for various classes of supports, than with application to practical visualization.

3 KelpFusion

In the design study performed by Alper *et al.* [1], sketch results drawn by participants hinted at a hybrid method, combining edges with filled faces to visualize sets. KelpFusion offers exactly such a hybrid approach by supporting a user-controlled trade-off between minimizing ink in the diagram (a minimal spanning tree) and maximizing the coherence of group boundaries (the convex hull). It builds on the visual appearance of Kelp Diagrams [8], but with some significant diffe rences. The first difference is the addition of filled faces. Filled faces provide a stronger sense of grouping for spatially close elements. To avoid misleading visualizations we use a few simple legibility rules to decide which faces to fill.

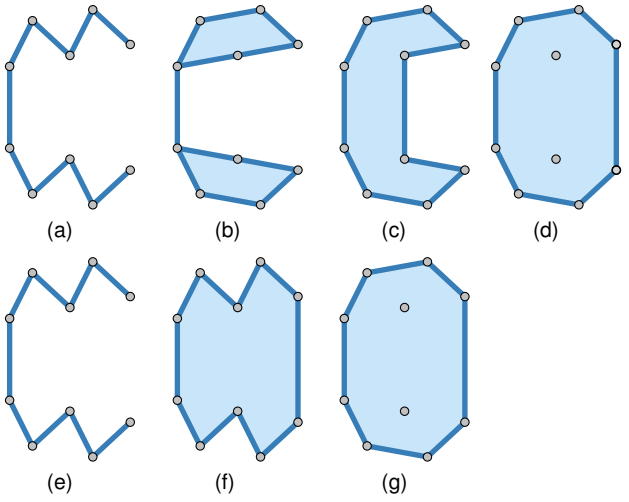


Figure 2. Difference in evolution from spanning tree to convex hull. We consider shortest-path graphs to be more appropriate for capturing shape and filling faces. (a-d) KelpFusion. (e-g) Kelp Diagrams.

The second difference is the graph that defines the edges. Whereas Kelp Diagrams use a detour factor to decide whether an additional edge should be inserted, KelpFusion uses a shortest-path graph. While both methods have a convex hull and a minimal spanning tree as extremes, the intermediate stages of a shortest-path graph identify clusters much more readily and allow us to fill many small faces. In contrast, the detour factor used for Kelp Diagrams mostly adds long edges, creating large faces. Figure 2 illustrates this difference. Our implementation of KelpFusion allows us to interactively vary the sparsity of the visualization.

Preliminaries. We assume the input is given as a set N of *nodes* corresponding to points in $\mathbb{R}^2$. We wish to visualize a number of *sets*, $S_1, \ldots, S_k$, where each set is a non-empty subset of N. The numbering of the sets indicates the front-to-back ordering. Every node v is assigned a unique *allocation area*. This allocation area is used to draw a characteristic nested Kelp "bubble" around v, one for each set that contains v. The allocation area is defined as the intersection of the Voronoi cell of v and a disk centered on v with radius r (a parameter). We link the allocation areas of the nodes with edges. A parameter w controls the width of a rendered edge.

Set order. The sets $S_1, \ldots, S_k$ are specified in their front-to-back ordering. This order affects the quality of the visualization. Since we typically have a small number

of sets, it would be possible to try all orderings. For automated selection, we would need a quality criterion to find the best drawing (e.g. the number of intersections). However, reordering the sets would require recomputing the visualization since the geometry is affected by the order. We find that ordering the sets by increasing size tends to produce the most readable results. That is, the largest sets are in the back, the smallest in front. We apply this heuristic order to all our results.

Algorithm overview. After computing the allocation areas, each set is processed in three phases. In the first phase, the *shortest-path graph* is computed to define the edges that connect the nodes. In the second phase, the shortest-path graph is analyzed to determine the faces of the graph that can be filled. In the third phase, the actual visual representation is computed from the shortest-path graph and its fillable faces. Algorithm 1 summarizes this process. Note that the sets are processed in a back-to-front order. Figure 3 illustrates the effect of each phase.

The first and second phase both introduce one parameter (t and A), to control the density of the shortest-path graph and the size of the fillable faces respectively. These parameters may have different values for different sets, as is illustrated in Figure 4. To compute the fillable faces (phase 2), only the shortest-path graphs of the lower sets need to be known. Therefore, it is also possible to compute the visualizations on a phase-by-phase basis, rather than set-by-set. This opens up possibilities for parallel processing. In the remainder of this section, we discuss each of the phases and some design considerations.

Algorithm 1 `KelpFusion`$(N, S_1, \ldots, S_k)$

Input: N is a node set and $S_1, \ldots, S_k$ are subsets of N
Output: a KelpFusion visualization for $S_1, \ldots, S_k$

1: Compute allocation area for each node in N
2: **for** $S_i = S_k$ **to** S_1 **do**
3: Compute shortest-path graph
4: Determine fillable faces
5: Construct visualization V_i
6: **return** visualizations $V_1, \ldots, V_k$

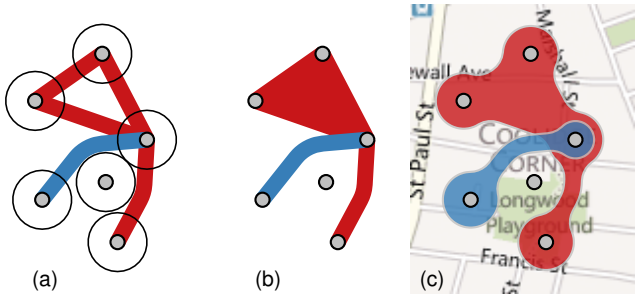


Figure 3. The three phases in KelpFusion. (a) Shortest-path graphs avoiding allocation areas. (b) A face is filled. (c) Final visualization.

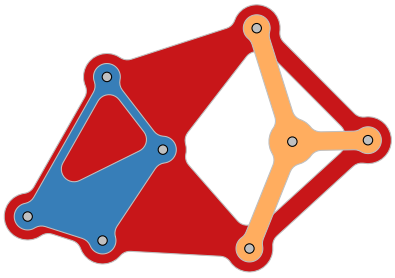


Figure 4. Different parameter settings in one visualization. Back-most set (red) uses a convex hull, filling all possible faces. The left set (blue) also uses a convex hull, but does not fill all possible faces. The right set (orange) uses a spanning tree.

3.1 Phase 1: Shortest-path graph

In the first phase, a shortest-path graph is computed for each set S_i, to determine the edges that visually link the nodes. It is computed as follows.

First, we compute the *reachability graph* G, a superset of the shortest-path graph. G contains an edge for each pair of nodes (u,v) in S_i. This edge is defined as the shortest Euclidean route between u and v, constrained to avoid any allocation area with a minimum clearance of $\frac{1}{2}w$, except those of u and v. Hence, when an edge is drawn with width w, its drawing does not intersect any allocation area. Note that (with $r > 0$ or $w > 0$) such a path need not exist and G may in fact be disconnected, implying a disconnected shortest-path graph. However, for most real-world datasets, G is connected. The edges of G can be computed using a visibility tangent graph [18]. For Kelp Diagrams, edges are selected from the reachability graph G based on a detour factor (called *benefit* [8]), which measures how much an additional edge improves the connectivity of the drawing. All edges that improve by at least a user-controlled factor are added. In contrast, for KelpFusion we compute a *shortest-path graph* of the reachability graph G.

Shortest-path graphs. A shortest-path graph aims to capture the shape of a point set with a graph and naturally adapts to varying point density [2]. Using a single parameter, this graph varies from a minimal spanning tree to the convex hull. In the context of Geographic Information Science, shortest-path graphs have been used to delineate imprecise regions, reconstructing a boundary of a region based on a set of points that are likely inside the intended region [2]. For KelpFusion, the shortest-path graph defines the edges to connect the nodes in a set, capturing the local shape of clusters of nodes.

The shortest-path graph defines a subgraph of the reachability graph G, based on a single parameter $t \geq 1$. If G is connected, the shortest-path graph is a spanning graph of S_i. Let $\mathrm{SPG}(S_i,t)$ denote the shortest-path graph for (the reachability graph of) S_i for a parameter value t. An edge $e = (u,v)$ is included in $\mathrm{SPG}(S_i,t)$ if and only if e is a shortest path between u and v in G, where the weight of an edge is defined as its length raised to the power t. The method to compute $\mathrm{SPG}(S_i,t)$ is given in Algorithm 2. The execution time of this algorithm is $O(n \cdot (|S_i| \log |S_i| + n))$, where n is the number of edges in G. Since n is quadratic in $|S_i|$, the execution time is $O(|S_i|^4)$.

We now briefly discuss the geometric structure of shortest-path graphs. To this end, we consider the case $r = 0$

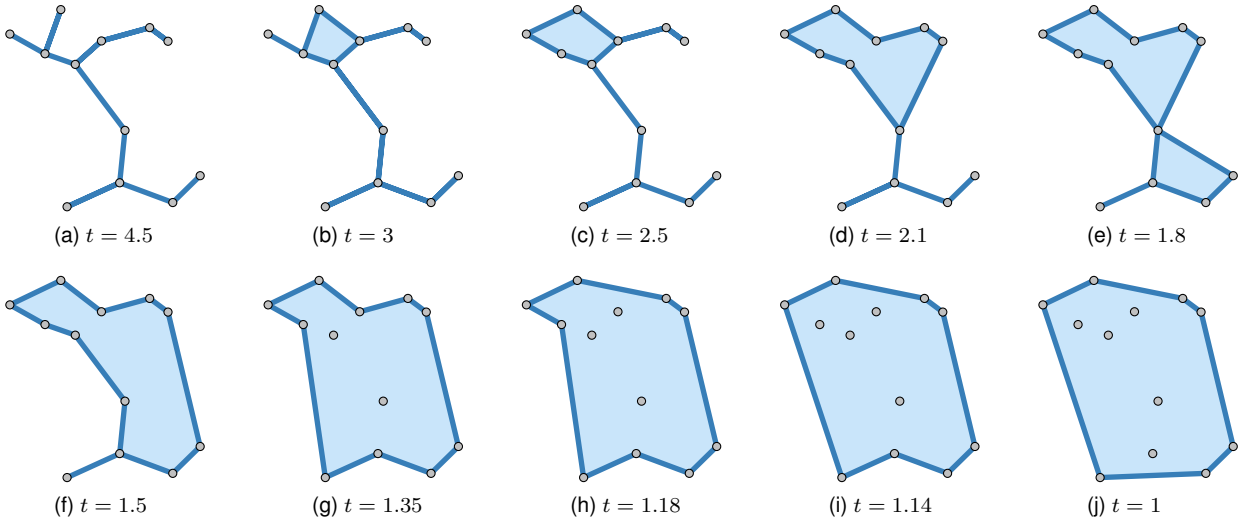


Figure 5. The outline of the shortest-path graph for varying t-values. The outline ranges from the minimal spanning tree to the convex hull.

Algorithm 2 `ShortestPathGraph`(S_i, G, t)

Input: S_i is a node set, G is the reachability graph of S_i, and $t \geq 1$

Output: the shortest-path graph of S_i for parameter t

```
Let e_1, ..., e_n denote the edges of G in increasing order of length
Let SPG be a graph with nodes S_i and no edges
for e = e_1 to e_n do
    Let p be the shortest path in SPG between the endpoints of e
    if p does not exist or weight(p) ≥ |e|^t then
        Add e to SPG with weight |e|^t
return SPG
```

and $w = 0$. Hence, every edge in the reachability graph is a straight line. Due to the triangle inequality, $\mathrm{SPG}(S_i, 1)$ is the complete graph on S_i, its outline being the convex hull of S_i. In contrast, $\mathrm{SPG}(S_i, \infty)$ is the Euclidean minimal spanning tree of S_i. The graph $\mathrm{SPG}(S_i, t)$ is a subset of $\mathrm{SPG}(S_i, t')$ for $t > t'$ and thus varying the value of t allows (the outline of) the graph to range from a spanning tree to the convex hull. This is illustrated in Figure 5. For $t \geq 2$ (and $r = 0$ and $w = 0$), $\mathrm{SPG}(S_i, t)$ is a subset of the Delaunay triangulation. Hence, for this special case, we can restrict the reachability graph G to contain only edges between the pairs of nodes that are neighbors in the Delaunay triangulation.

For $t < 2$, $r > 0$, or $w > 0$, the monotonicity property ($\mathrm{SPG}(S_i, t) \subseteq \mathrm{SPG}(S_i, t')$ for $t > t'$) still holds and $\mathrm{SPG}(S_i, \infty)$ is still a spanning tree. However, the shortest-path graph is no longer necessarily a subset of the Delaunay triangulation. But, if we simply restrict the reachability graph G to the edges of the Delaunay triangulation, our algorithm still works well and the computation time is decreased significantly: since n is now linear in $|S_i|$, the computation time is $O(|S_i|^2 \log |S_i|)$.

Insertion order. When decreasing t from infinity down to 1, the shortest-path graph grows from the minimal spanning tree to the complete reachability graph G. Each edge e of G has an associated t-value at which it is added to the shortest-path graph, that is, the largest value of t such that e is in $\mathrm{SPG}(S_i, t)$. These t-values naturally induce an insertion order of the edges. Computing exact t-values is difficult, but they can be approximated sufficiently well. The t-values give us the actual insertion order, allowing for interactivity. (Note that for certain point configurations the t-value of an edge can be less than 1 and hence such an edge never appears in shortest-path graph. However, such edges are always redundant and can safely be omitted.)

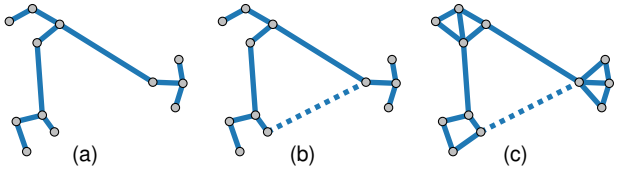


Figure 6. (a) The Euclidean minimal spanning tree ($t = \infty$) on a synthetic point set with three small clusters. (b) Inter-cluster edge is the first to appear for $t < \infty$. (c) Insertion order has changed, extra inter-cluster edge appears after reaching convex hull for each cluster.

The shortest-path graph is by definition scale independent. However, in the case of set visualizations, we may prefer to add short edges early, to converge to a (local) convex hull for clusters more quickly. To modify the insertion order, we need to modify only the weight definition for the shortest-path graph: instead of $|e|^t$, we use $(|e| + C)^t$, where $|e|$ is the length of an edge and C is a non-negative parameter. With this change, shorter edges are preferred over longer edges. Moreover, this change does not affect the "extreme" cases of $t = 1$ and $t = \infty$. Figure 6 illustrates the effect of changing the insertion order.

Penalties to improve readability. Kelp Diagrams can include additional weights (penalties) to alter the route of edges and improve the legibility of the visualization. In particular, crossing edges and angular change can induce penalties. We can use the same ideas for KelpFusion.

However, it is very costly to compute the route of edges with additional penalties, making it impossible to interactively vary parameters. Moreover, in most cases, it seems unnecessary to include penalties since the visual quality of the drawings is already high.

3.2 Phase 2: Fillable faces

Since KelpFusion is based on shortest-path graphs it naturally identifies dense clusters of nodes in a set. To strengthen this effect and convey an additional sense of grouping, we fill enclosed faces of the shortest-path graph, making KelpFusion a hybrid method that combines linear and areal features. However, filling all faces results in visualizations with false memberships, ambiguity, and areas that are too visually dominant. Therefore, we use four simple legibility rules to decide whether a given face may be filled.

The first rule avoids large filled faces. Large faces obscure much of a background map, are too visually dominant, and may suggest a higher density. The second rule avoids false memberships. If a face f formed by nodes of a set S_i is filled but f contains a node v that is not in S_i, then the visualization incorrectly implies that v is actually part of S_i or, alternatively, it may hide the other set memberships of v. This is illustrated in Figure 7 (a-b).

The third rule maintains the visual continuity of sets below S_i by preventing a face from hiding edges of a lower set (Figure 7 (c-e)). Finally, the fourth rule also maintains visual continuity and prevents false memberships. If the boundary of a face f shares one or more edges with a lower set, then filling f may incorrectly imply that the other nodes on f are also part of the lower set (Figure 8).

More formally, let f denote a face in the shortest-path graph of S_i. Let $|f|$ denote the area of f, nodes(f) the nodes in S_i on its boundary, and $|\text{nodes}(f)|$ the number of nodes on its boundary. Face f is filled if and only if all of the following conditions are met:

1) $\frac{|f|}{1+|\text{nodes}(f)|} < A$, where A denotes a parameter;
2) f does not contain a node in $N \backslash S_i$;
3) the interior of f does not overlap an edge of a lower set S_j $(j > i)$;
4) if the boundary of f is shared with an edge of a lower set S_j $(j > i)$, then nodes(f) is a subset of S_j.

3.3 Phase 3: Visualization

After computing the shortest-path graph and fillable faces, we construct the final visualization. To this end, we compute a polygon, potentially containing holes. This polygon is the union of the characteristic Kelp "bubbles" at each node, the thickened edges in the shortest-path graph, and the fillable faces. To this polygon, we apply similar visual postprocessing to that used in Kelp Diagrams. That is, we smooth the polygon boundary for visual continuity, assign a unique color, and give it a thin gray outline for better visual separation between the sets. To avoid false memberships as a result of smoothing, we subtract the allocation area of each node that is not in the set from the smoothed shape. In addition, we use a slight transparency (20%), kept low to avoid color-blending. In visually dense areas, transparency can help to determine set memberships and to track the shapes. In visually sparse areas, it causes the underlying map to be slightly more readable.

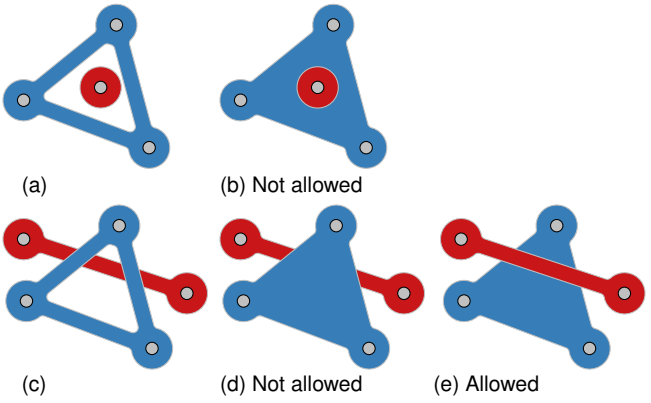


Figure 7. (a-b) Regardless of order, face cannot be filled according to Condition 2. (c-e) Depending on order, face may or may not be filled according to Condition 3.

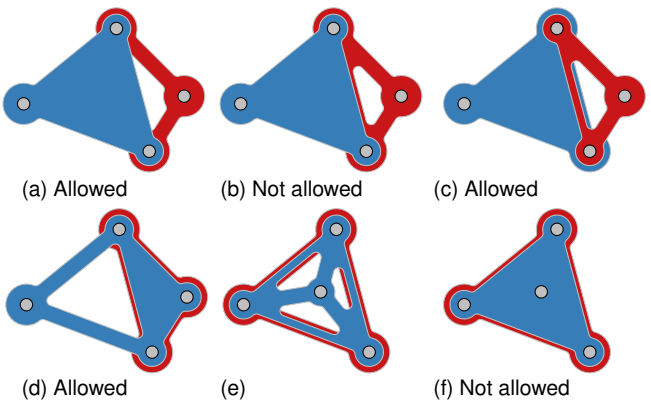


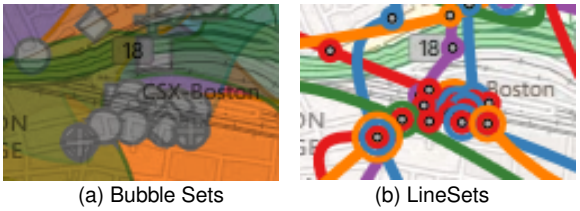
Figure 8. Illustrations for Condition 4. (a) Face can be filled, since no edge is shared. (b) Face cannot be filled, since an edge is shared with lower set. (c) Face can be filled, since shared edge is in a higher set. (d) Right face can be filled since lower set contains all vertices of the face, but left face cannot be filled as it would hide the edge of the lower set. (e-f) Filling the faces of the top set may incorrectly imply that central point is also in bottom set.

3.4 Implementation

Our implementation of the KelpFusion algorithm allows us to vary the t-values and A-values (the parameters of the first and second phase) interactively and independently. We achieve this by omitting crossing and angular change penalties and by restricting the reachability graphs to the Delaunay triangulation, limiting its size (and thus the length of the insertion order) to $O(|S_i|)$ for each set S_i. Omitting penalties makes the reachability graph and the corresponding insertion order for the shortest-path graph of a set independent of the other sets. Hence, we can precompute the insertion order for each set. To vary the t-value of a set, we simply use this precomputed insertion order. Varying the A-values at interactive rates is supported

by storing with each face in the shortest-path graph the overlapped edges in the reachability graph of lower sets. To decide whether a face can be filled or not, we have to check only if the stored edges are in fact drawn (i.e. checking the t-value of the overlapped edges to the value of t configured for the lower set). We used our implementation to generate all KelpFusion figures throughout the paper.

Illustrations. Here we briefly showcase the range of visualizations possible with KelpFusion. Figure 9 zooms in on a cluster in a dataset of Boston restaurants. Despite the densely clustered data points KelpFusion is still able to clearly distinguish between different sets. In Figure 10 we are using a convex hull for the lowest (red) set and spanning trees for all others. This emphasizes the interaction of the red set with the other sets, in particular, it clearly indicates the nodes that are *not* in the red set. Figure 11 shows how KelpFusion deals with a dataset frequently used in [8].



(a) Bubble Sets (b) LineSets

(c) KelpFusion

Figure 9. Area of high density visualized with three different techniques.

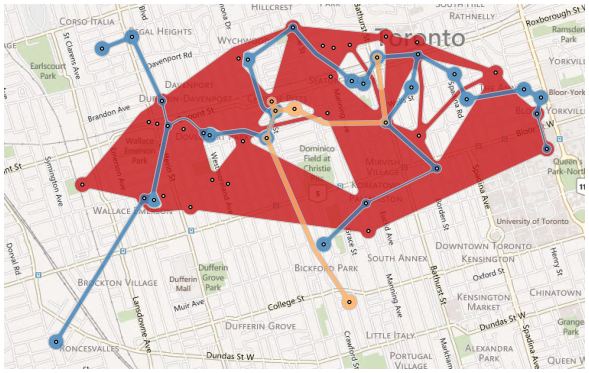


Figure 10. Using a convex hull for the lowest (red) set and spanning trees for the others emphasizes which nodes are *not* in the red set.

3.5 Other design considerations

Here we briefly discuss some design considerations for KelpFusion.

Allocation area. KelpFusion, like Kelp Diagrams, uses a fixed allocation area for each node, scaling the representations of sets to fit within the allocation area. We scale

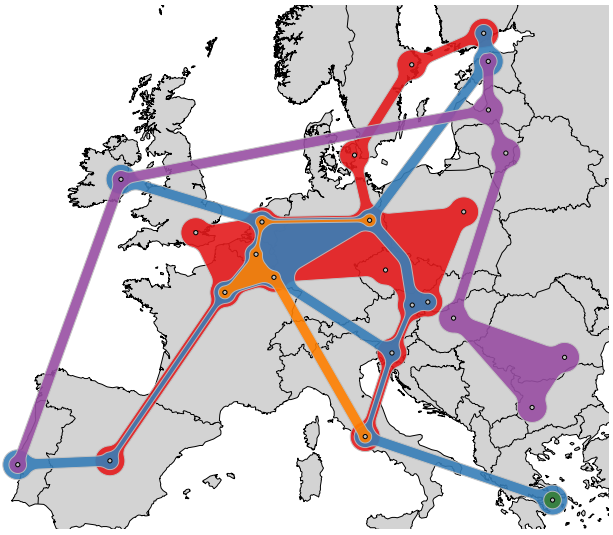


Figure 11. A KelpFusion visualization for a Europe dataset, as used in Figure 12 in [8]. Dataset courtesy of Kasper Dinkla.

such that the full allocation area is used and every set has approximately the same visible area. LineSets handle node area differently. They simply add a ring around a node for each set the node is contained in. Hence a node contained in many sets uses more area than a node contained in only a few (see Figure 12). This draws additional attention to high-degree nodes. KelpFusion might also benefit from having degree-dependent allocation areas. Since large allocation areas can easily disconnect the shortest-path graph, smaller allocation areas for low-degree nodes may benefit connectivity in some cases. Another user study would be advisable to confirm or refute the actual benefit of allocation areas of varying sizes.

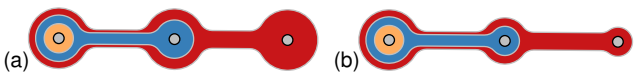


Figure 12. (a) Constant allocation area used by KelpFusion. (b) Allocation area depending on number of sets, similar to LineSets.

Thinning areal features. When a face has long edges, it may draw too much attention. Instead of not filling such a face, we can simply reduce its area by thinning its outline, similar to tapering edges (see Figure 13). Doing so may reduce clutter, but it is unclear if the legibility and the performance of KelpFusion would actually improve.

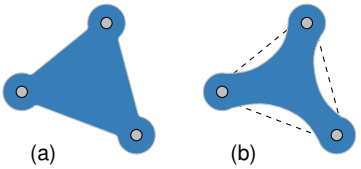


Figure 13. Thinning areal features. (a) Filled triangular feature in KelpFusion. (b) Sketch of thinned version.

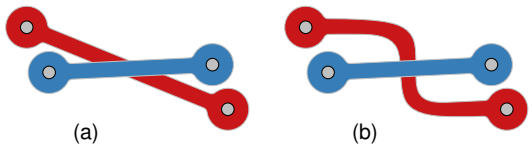


Figure 14. Right-angle crossings may improve legibility. (a) Shallow crossing in KelpFusion. (b) Sketch of right-angle crossing.

Right-angle crossings. Edge-crossings are often unavoidable, even when penalties are imposed. To make crossings as clear as possible, edges should intersect at a large angle, ideally forming a right-angle crossing (see Figure 14). This minimizes the area of overlap between the two edges and thus also minimizes the interruption of the visual continuity of the lower set. One could consider using force-directed methods in a postprocessing step to encourage right-angle crossings. Such an approach, however, necessarily adds visual complexity (inflection points) and uses extra ink.

4 User Study

4.1 Method

We designed a controlled experiment to assess the readability of our KelpFusion technique compared to state-of-the-art set visualization techniques. We had two goals: (1) to evaluate how the graphs of KelpFusion compare to the single continuous path as generated by LineSets; (2) to evaluate how the mixed use of hulls and links compares to a single concave hull as generated by Bubble Sets. Therefore, we compared KelpFusion to these two techniques. We present an example of each in Figure 15. We conducted a controlled experiment with a within-subject design: 3 Visualization Techniques $\times$ 4 Tasks $\times$ 2 Difficulty Levels $\times$ 3 Repetitions. We recruited 13 participants who answered 72 questions each in a multiple-choice format.

Note that we make no comparison to Kelp Diagrams as KelpFusion achieves a superset of Kelp Diagrams. While assessing the performance of different settings for KelpFusion (spanning from Kelp Diagrams-like representations to Bubble Sets-like ones) would certainly prove interesting, we felt that introducing more conditions in the present study would lead to an unreasonable experiment time and cause participants' fatigue.

Datasets. We used real data of restaurant locations in the Boston area gathered from Bing Maps. We grouped cuisine, price qualification, and rating of restaurants to form sets. We filtered the data to vary the difficulty levels of different set arrangements. We estimated the difficulty of set arrangements as explained in [10], controlling the number of 2-set, 3-set and 4-set intersections. Statistics are presented in Table 1. We controlled the number of elements in each set to ensure that size comparison questions were not too simple or too ambiguous to answer, and questions on counting elements were not too tedious. The spatial arrangement of the sets was not controlled as we used real geographic data. Figure 15 shows example pictures used in the experiment. Colors for the sets were based on ColorBrewer [4].

Table 1. Dataset statistics.

	# sets	# elements	# 2-set	# 3-set	# 4-set
Medium 1	4	15 to 39	22	3	0
Medium 2	4	17 to 49	17	3	0
Hard 1	5	12 to 29	17	4	2
Hard 2	5	14 to 29	16	5	2

Tasks. We identified four readability tasks focusing on browsing elements contained in a set, determining which sets an element belongs to, and identifying set intersections. We opted for multiple-choice answers to encourage participants to rely on their visual perception of the sets rather than carefully counting items and spending time verifying their answers. Providing multiple answers also allowed us to reduce the completion time required for each question, enabling the within-subject comparison of all three techniques with multiple task repetitions. The task types and an example for each are given below:

SizeOverview	Are there more Thai or more French restaurants?
SizeCount	How many restaurants serve Italian food?
SetIntersection	How many Thai restaurants are rated 5?
SetMembership	What is the highlighted restaurant?

Highlighting for the SetMembership task was done by coloring the node yellow. We felt an embedding of restaurants on a map would be a familiar visualization to our non-technical study participants. Further, using a

Figure 15. Example of images used in the experiment. From left to right: Bubble Sets, LineSets, and KelpFusion. An area of high density is highlighted. Note that the images in the actual study did not feature such a highlight or any other method of interaction.

geographic embedding avoids questions of layout, which may be chosen to better suit one technique over another, or some data-based embedding which would then need to be explained to the participant. When working with a map it is tempting to include tasks based on geography, such as way-finding. However, we deliberately avoided such tasks to preserve external validity (transferability of the results to other tasks).

Participants and apparatus. We recruited 13 participants (7 males and 6 females). The age of our participants ranged between 27 and 47 years, with a mean age of about 28 years. The participant pool was balanced for age and gender and was screened to include people with general computer experience, but not programmers. All participants had normal or corrected-to-normal vision, none being color-blind. The experimenter ran 1.5 hour sessions with two participants at a time. Each participant completed the experiment on an individual PC, using a 24" widescreen monitor with a resolution of 1900 by 1200.

Procedure. Before the controlled study, participants received training for each visualization technique. They were instructed which strategies to utilize for a particular task with each visualization technique.

We counterbalanced the order of visualization techniques across users. We kept the order of tasks fixed throughout the study, advancing from easier to harder tasks. We fixed the order of the datasets from low to high difficulties. For the repetitions, we created isomorphic questions using the same data, but flipped the set arrangements both horizontally and vertically and modified labels to avoid memorization.

The experiment software displayed questions in the described order and recorded accuracy and time taken for each answer. The software first displayed the question without any visualization. When users understood the text and were ready to answer the question, they clicked a button to view the corresponding visual. Time was measured only when the visualization was visible. To keep the study at a reasonable length, we limited each question to a maximum of 40 seconds.

Finally, after the experiment, we collected user preferences and comments using a questionnaire. The study lasted approximately 60 minutes including training and post-experimental questionnaire.

4.2 Hypotheses

For accuracy and completion time, we hypothesized the following:

- (H1) KelpFusion and LineSets reduce the visual clutter caused by overlapping set regions compared to Bubble Sets. Hence, both techniques will have better completion time and accuracy performance than Bubble Sets;
- (H2) KelpFusion provides a better overview of the set size than LineSets as it creates a more effective signature image emphasizing density with the use of closed faces. Hence, KelpFusion will outperform LineSets for SizeOverview tasks;
- (H3) The closed faces generated by KelpFusion simplify the set shapes for dense regions as opposed to the zigzagging paths of the LineSets technique, thus KelpFusion will have a better performance for SetIntersection tasks;
- (H4) Compared to the sequential nature of LineSets paths, the branching geometry of KelpFusion may decrease the performance when users need to browse all elements in a set as in the case of SizeCount tasks;
- (H5) KelpFusion will outperform LineSets for SetMembership tasks when many sets intersect (that is, for difficult datasets), as the concentric circles around points used by LineSets may overlap in dense regions, thus decreasing its legibility.

For the participants' preferences, we hypothesized the following:

- (H6) KelpFusion and Bubble Sets better convey the concept of groups since LineSets imply a sequential order;
- (H7) KelpFusion is more aesthetically pleasing than LineSets (which has zigzagging lines) and less cluttered than Bubble Sets (which occlude much of the map).

4.3 Results

We used a repeated-measure analysis of variance (RM-ANOVA) to analyze accuracy and time performance results. We performed the RM-ANOVA on the logarithm of the task times to normalize the skewed distribution, as is standard practice with reaction time data. Analysis of the time performance is reported for correct answers only.

Accuracy. The accuracy results are summarized in Table 2 and Figure 16 (left). We found a significant effect of accuracy for Visualization Technique ($F(2,24) = 64.96, p < 0.0001$). We verified (H1) as, overall, participants had significantly more accuracy with LineSets and KelpFusion than with Bubble Sets.

Table 2. Accuracy results. Means are expressed in percentages, standard deviation is indicated in parentheses. Significant differences are indicated by *. Most accurate results are indicated in bold.

	Bubble Sets	LineSets	KelpFusion
All tasks*	54.5 (2.4)	**84.3** (4.5)	**86.5** (3.9)
SizeOverview	85.9 (3.7)	80.8 (5.6)	84.6 (4.8)
SizeCount*	41.0 (8.6)	**82.1** (7.7)	**84.6** (7.4)
SetIntersection*	37.2 (4.3)	**84.6** (4.0)	**82.1** (4.4)
SetMembership*	53.8 (5.0)	**89.7** (5.4)	**94.9** (2.9)

RM-ANOVA also revealed a significant effect of Task ($F(3,36) = 5.25,\ p < 0.01$) and of the interaction Visualization x Task ($F(6,72) = 8.68,\ p < 0.0001$). Pairwise comparisons revealed a significant difference in accuracy between visualizations for three of the tasks: SizeCount, SetIntersection and SetMembership. For all three tasks, KelpFusion and LineSets both performed about 50% more accurately than Bubble Sets, validating our first hypothesis (H1). No significant differences were found between KelpFusion and LineSets.

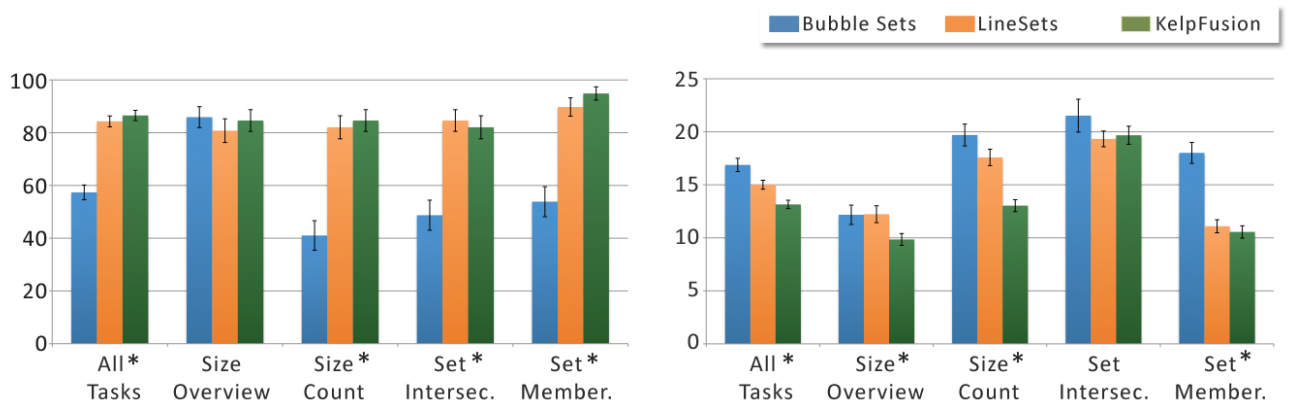


Figure 16. (left) Mean accuracy (in percentage) of each technique for tasks. (right) Mean time (in seconds) of each technique for tasks. Errors bars represent 2 standard errors. Significant differences are indicated by *.

We also found a significant effect of Difficulty ($F(1,12) = 11.26$, $p < 0.01$) and Visualization $\times$ Difficulty ($F(2,24) = 13.88$, $p < 0.0001$). Surprisingly, we found that overall, participants committed about 10% more errors with the datasets of low difficulty. Investigating this surprising finding further, we found that while KelpFusion and LineSets tend to remain constant in accuracy across tasks and difficulties, Bubble Sets had significant increases in accuracy for high difficulty datasets for SetIntersection and SetMembership tasks. We believe this may be due to the fact that in complex set arrangements, participants gave up using the set boundaries, as the Bubble Sets diagram was extremely cluttered, and relied only on the glyphs attributed to set elements, which prove to be a somewhat more accurate way to find the answer. It is important to note however, that, despite this increase, Bubble Sets remained 20% to 40% less accurate than the other techniques.

Time. For completion time, we analyzed only the correct answers using a mixed linear model capable of handling missing data cases. For KelpFusion and LineSets, we excluded about about 10% incorrect cases (312 total cases per technique). For Bubble Sets, we excluded about 17% of incorrect cases.

The completion-time results are summarized in Table 3 and Figure 16 (right). We found a significant effect of time for Visualization ($F(2,24) = 44.69$, $p < 0.0001$). Pairwise comparisons showed that KelpFusion is about 12% faster than LineSets and 27% faster than Bubble Sets. The results also revealed that LineSets is about 17% faster than Bubble Sets.

We also found a significant effect of Task ($F(3,36) = 141.62$, $p < 0.0001$) and of the interaction Visualization $\times$ Task ($F(6,72) = 10.46$, $p < 0.0001$). Pairwise comparison revealed significant differences in time between visualizations for three of the tasks: SizeOverview, SizeCount and SetMembership. For SizeOverview, we found that KelpFusion was about 17% faster than the two other techniques, as predicted in (H2). For SizeCount, we found that KelpFusion performs 35% faster than Bubble Sets. For this task, we were surprised to find that KelpFusion is also 28% faster than LineSets, as we originally thought the opposite, that is, the sequential nature of LineSets would be more effective for browsing elements of a set (H4). For SetMembership, we found that KelpFusion and LineSets are both about 40% faster than Bubble Sets, but did not observe any significant differences between KelpFusion and LineSets as we hypothesized (H5).

Table 3. Time results. Means are expressed in seconds, standard deviation is indicated in parentheses. Significant differences in time are indicated by *. Faster times are indicated in bold.

	Bubble Sets	LineSets	KelpFusion
All tasks*	17.17 (0.4)	14.19 (0.3)	**12.41** (0.3)
SizeOverview*	10.40 (0.5)	10.50 (0.6)	**8.74** (0.5)
SizeCount*	18.49 (0.8)	16.70 (0.6)	**11.95** (0.5)
SetIntersection	22.58 (1.1)	19.45 (0.8)	19.35 (0.9)
SetMembership*	17.19 (0.6)	10.10 (0.4)	**9.60** (0.4)

Contrary to our hypothesis (H3), we did not find any significant differences neither in accuracy nor completion time between LineSets and KelpFusion for the SetIntersection task.

User preference. At the end of the experiment we asked the participants to rate how confident they were answering each type of question, how cluttered and how aesthetically pleasing the visualization was, and finally whether the visualization provided a strong sense of grouping. Participants ranked each technique separately using a Likert scale from 1 (worse) to 5 (best). Figure 17 summarizes these subjective user ratings.

Participants were significantly more confident using LineSets and KelpFusion compared to Bubble Sets (both $p < 0.001$) for all tasks. There were no significant differences between KelpFusion and LineSets techniques in terms of confidence in the answers. Only for the SizeCount task, users indicated more confidence using LineSets (about 40%), however, only 3 out of 13 participants commented that the linear design helped them browsing elements (H4). On the contrary, 7 out of 13 commented on the fact that the lines were "wiggly and visually distracting when counting the elements", "intersecting in a confusing way" and "hard to tell apart". A few participants also commented that they would get lost following the lines and had to start counting again from the beginning. We believe that these comments

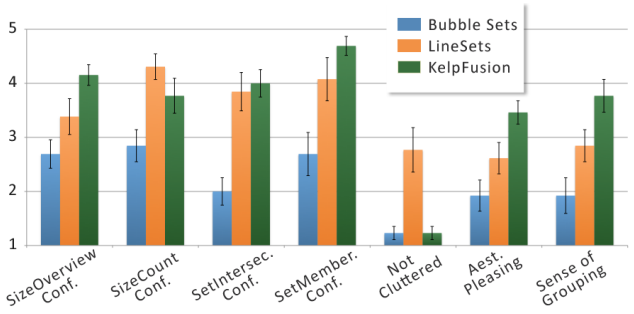


Figure 17. Subjective user ratings for each technique. Errors bars represent 2 standard errors. Significant differences are observed in all rankings.

explain why KelpFusion outperformed LineSets in terms of completion time for this task.

Users rankings and comments indicate that participants found LineSets less cluttered than the other two techniques (both $p < 0.01$). This outcome is expected due to the linear representation of LineSets, minimizing the occlusion of the background map. However, as we originally hypothesized (H6), LineSets convey less of a sense of grouping compared to KelpFusion as the linear design of LineSets implies a sequential order which is not present in the data ($p < 0.03$). To our surprise, participants rated Bubble Sets as the least effective method for conveying the notion of grouping compared to both KelpFusion ($p < 0.001$) and LineSets ($p < 0.03$). Although Bubble Sets relies on the common visual metaphor of enclosing geometries, we believe that this ranking is due to the high occlusion produced by overlapping regions in Bubble Sets, making the overall drawings difficult to read. For example, participants commented that "the overlapping color blobs made it near impossible to figure out how many were overlapping", "creating visual chaos" and a few participants commented that they "could not find a thing in there".

Finally, (H7) was partially validated as participants found KelpFusion aesthetically more pleasing than Bubble Sets ($p < 0.001$) and, while the questionnaire results were not significant between KelpFusion and LineSets, 8 out of 13 participants commented that KelpFusion was the most aesthetically pleasing representation overall.

4.4 Summary and limitations

Hypotheses summary. We confirmed that KelpFusion and LineSets outperform Bubble Sets both in accuracy and time for all tasks (H1), which seems to indicate that both techniques improve the readability of overlapping set regions. We also found that KelpFusion provides a better overview of the set size than LineSets (H2), as it facilitated faster results.

However, we originally hypothesized that the filled faces generated by KelpFusion would simplify the set shapes for dense regions as opposed to the zigzagging paths of LineSets, thus providing a better performance for SetIntersection task (H3) as well as the SetMembership task (H5). Contradictory to our hypothesis, we did not find any significant differences between the two techniques. This may point to a limitation of our study. As we opted to use real data and restrict the number of elements to a small number to limit the experiment duration, the datasets we used contained relatively few clusters of points. It may be possible that, with larger and denser clusters, KelpFusion would improve performances compared to LineSets. However, it is interesting to note that while KelpFusion was on par with LineSets for these two tasks, it did improve on the Bubble Sets technique.

We were surprised to find that KelpFusion performed faster than LineSets for counting elements of a set (H4), as we originally hypothesized the opposite. We propose an explanation later in this section.

Finally, our hypotheses on participants' preferences were partially validated. Participants found that KelpFusion conveys the concept of groups better than LineSets (H6) but, as we did not expect, KelpFusion was also ranked higher than Bubble Sets. Participants commented that KelpFusion was the more aesthetically pleasing representation overall, and the ranking indicates that it is found less cluttered than Bubble Sets (H7).

Key findings. Overall, results of this study indicate that:

(1) Using a graph holds up well compared to a single continuous path technique. We originally hypothesized that representing a set by a single continuous line would have advantages over a branching graph geometry. In particular, we hypothesized that browsing elements of a set (captured by the SizeCount task) would be facilitated by the sequential nature of a line. Therefore, we were surprised that KelpFusion performed faster than LineSets for this task. Several participants commented that single continuous lines were tortuous and difficult to follow when intersecting with others. They commented that when they got lost, they went back to the beginning of the line and resumed the counting, which may partly explain the decrease of performance using LineSets. Reflecting more on this issue, we also believe that KelpFusion may take advantage of the capacity humans have for subitizing [11], that is, for performing accurate and rapid judgments on small numbers of items grouped together. With this hypothesis, faster completion times could be explained by the fact that participants estimated the number of elements by subitizing, summing estimated numbers of elements in short branches and filled faces in KelpFusion instead of counting individual elements. Further experiments would be required to validate this hypothesis.

(2) Using a mix of hulls and links has benefits over a single concave hull technique. Results of the experiment confirmed that KelpFusion had strong benefits over using a single concave hull technique such as Bubble Sets. Results indicate that KelpFusion improved accuracy and performance time but that participants also found it to decrease clutter on the map, provide more aesthetically pleasing drawings and increase the sense of groupings. While it is hard to capture high-level impressions and memorability of a representation, we also believe that

the filled faces generated by KelpFusion provide a good association between set elements and their spatial locations, e.g. "where most elements of a given set are located". However, further experiments would be required to validate this hypothesis.

Limitations. Our study has some limitations that need to be taken into consideration. As with all controlled experiments, the results we found apply to a very limited number of datasets and a few low-level tasks. While they indicate a trend, one needs to be careful in generalizing these results. For example, Bubble Sets were originally developed for non-overlapping sets. We did not include non-overlapping datasets in this experiment as our primary goal was to understand how KelpFusion improves the readability of set intersections. Thus, Bubble Sets may remain a good choice when representing simpler set arrangements, in which sets rarely overlap.

The KelpFusion technique is composed of a mix of hulls and lines, in which hulls (filled faces) highly depend on the spatial arrangements of elements. However, as we opted to use real data, we did not control the spatial arrangements of the elements. Moreover, we had to restrict the number of displayed elements to limit the experiment to a reasonable duration. These two choices may have impacted the results we obtained. However, we believe that the comparison to other techniques was not biased. Increasing the number of elements and ensuring stronger spatial groupings of elements is likely to confirm the benefits of KelpFusion over the other techniques.

5 Conclusion and Future Work

We introduced a hybrid set visualization technique, called KelpFusion. Building on the visual representation of Kelp Diagrams, KelpFusion is composed of both linear and areal features. This bridges the gap between hull methods, such as Bubble Sets and Euler Diagrams, and linear methods, such as LineSets and Kelp Diagrams. KelpFusion uses shortest-path graphs to define the linear features of the visualization, controlling the sparsity with a single parameter. Since these shortest-path graphs naturally capture clusters and shape, this opened the possibility of filling faces in the graph to obtain areal features. The maximal size of areal features is again controlled by only a single parameter. This combination of linear and areal features makes KelpFusion the first hybrid visualization technique, covering a whole spectrum of possible set visualizations, from sparse trees to dense hulls.

We performed a controlled experiment to compare the performance, in both accuracy and time, of the new KelpFusion method with existing methods (Bubble Sets and LineSets). These experiments showed that KelpFusion is never outperformed by any of the other methods.In terms of accuracy, it performs significantly better than Bubble Sets and is on par with LineSets. In terms of completion time, it performs significantly better than both other methods. We conclude that the combination of (branching) linear features and areal features of KelpFusion is a good way to visualize sets. The preferences of the participants also indicate that KelpFusion is aesthetically more pleasing and provides a better sense of grouping. While LineSets outperformed KelpFusion in terms of being less cluttered, we do note that the parameterization of KelpFusion allows for far sparser representations as well, even up to a minimal spanning tree. In such a case, KelpFusion would be less cluttered. A user study experimenting with the various settings of KelpFusion and assessing the readability of hybrid diagrams spanning from Kelp Diagrams-like set representations to Bubble Sets-like representations is an interesting future direction.

In addition, as mentioned earlier, scalability in the number of sets and the complexity of their intersection (see Figure 9) is an issue for KelpFusion as it is for any set visualization technique that attempts to preserve the spatial location of elements. To handle a large number of sets and "untangle" complicated intersection areas, one may consider relaxing the spatial constraint on the set elements, e.g. Euler Diagrams [10]; or adopt a dedicated set representation, e.g. permutation matrices [13]. Another direction that would enable scalability to larger datasets is through interactive filtering and aggregation. To support this KelpFusion would need to be modified to achieve visual stability of set boundaries as elements and sets are added or removed from the visualization.

Acknowledgements

The authors would like to thank Kasper Dinkla for providing details and visualizations of Kelp Diagrams. W. Meulemans and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.022.707.

REFERENCES

[1] B. Alper, N. Henry 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.

[2] M. de Berg, W. Meulemans, and B. Speckmann. Delineating Imprecise Regions via Shortest-Path Graphs. In *Proc. 19th ACM Symposium on Advances in Geographic Information Systems*, pages 271–280, 2011.

[3] U. Brandes, S. Cornelsen, B. Pampel, and A. Sallaberry. Path-based supports for hypergraphs. In *International Workshop on Combinatorial Algorithms*, volume 6460 of *LNCS*, pages 20–33, 2010.

[4] C. A. Brewer. http://www.colorbrewer2.org, access date: Januari 2012.

[5] K. Buchin, M. van Kreveld, H. Meijer, B. Speckmann, and K. Verbeek. On Planar Supports for Hypergraphs. *Graph Algorithms and Applications*, 14(4):533–549, 2011.

[6] H. Byelas and A. Telea. Towards realism in drawing areas of interest on architecture diagrams. *Visual Languages and Computing*, 20(2):110–128, 2009.

[7] 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.

[8] K. Dinkla, M. van Kreveld, B. Speckmann, and M. A. Westenberg. Kelp Diagrams: Point Set Membership Visualization. *Computer Graphics Forum*, 31(3pt1):875–884, 2012.

[9] A. W. F. Edwards. *Cogwheels of the mind*. John Hopkins University Press, 2004.

[10] N. Henry Riche and T. Dwyer. Untangling Euler Diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1090–1099, 2010.

[11] E. L. Kaufman, M. W. Lord, T. W. Reese, and J. Volkmann. The Discrimination of Visual Number. *The American Journal of Psychology*, 62(4):498–525, 1949.

[12] M. Kaufmann, M. van Kreveld, and B. Speckmann. Subdivision drawings of hypergraphs. In *16th International Symposium on Graph Drawing*, volume 5417 of *LNCS*, pages 396–407, 2009.

[13] B. Kim, B. Lee, and J. Seo. Visualizing Set Concordance with Permutation Matrix and Fan Diagram *Interacting with Computers*, 19(5-6):630–643, 2007.

[14] P. Simonetto and D. Auber. Visualise Undrawable Euler Diagrams. In *Proc. 12th Conf. on Information Visualisation*, pages 594–599, 2008.

[15] P. Simonetto, D. Auber, and D. Archambault. Fully Automatic Visualisation of Overlapping Sets. *Computer Graphics Forum*, 28(3):967–974, 2009.

[16] 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.

[17] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press (Cheshire, CT), 1983.

[18] R. Wein, J. van den Berg, and D. Halperin. The visibility-Voronoi complex and its applications. *Computational Geometry*, 36(1):66–87, 2007.

[19] M. Wertheimer. *Drei Abhandlungen zur Gestalttheorie*. Palm & Enke (Erlangen), 1925.

Wouter Meulemans is a PhD student at the Department of Computer Science and Mathematics of the TU Eindhoven (the Netherlands). He received his Master's degree Cum Laude in 2010 at TU Eindhoven on the topic of delineating imprecise regions. His current research deals with cartographic visualization with an emphasis on schematic maps. His research interests include computational geometry, information visualization, and automated cartography.

Nathalie Henry Riche received her PhD in human-computer interaction and information visualization in 2008 from the University of Paris-Sud and INRIA (France) and the University of Sydney (Australia). She joined Microsoft Research (USA) as a full-time researcher in 2008. Her interests lie in the visual exploration of graphs and networks, as well as the design of novel interactions to create and interact with visualizations.

Bettina Speckmann received her PhD in computer science from the University of British Columbia (Canada) in 2001. She spent two years as a postdoc at the Institute for Theoretical Computer Science of ETH Zurich (Switzerland) and became an assistant professor at the Department of Mathematics and Computer Science at TU Eindhoven (the Netherlands) in 2003. In 2008 she became associate professor, and in 2012 full professor at TU Eindhoven. Bettina's research interests include the design and analysis of algorithms and data structures, discrete and computational geometry, applications of computational geometry to geographic information systems, graph drawing, and cartographic visualization.

Basak Alper received her PhD from the Media Arts and Technology Program at UC Santa Barbara (USA) in 2013. She holds two MS degrees in Computer Science and Multimedia Engineering. Her current research interests lie at the intersection of information and scientific visualization.

Tim Dwyer received his PhD from the University of Sydney (Australia) in 2005. He was a Research Fellow at Monash University (Australia) until 2008, then became a Visiting Researcher and Senior Software Development Engineer at Microsoft (USA) until 2012. He is now a Senior Lecturer and Larkins Fellow at Monash University (Australia). He has worked extensively in the areas of graph layout and interactive visualization.