MoleView: An Attribute and Structure-based Semantic Lens for Large Element-based Plots

C. Hurter, O. Ersoy, A. Telea

Abstract—We present MoleView, a novel technique for interactive exploration of multivariate relational data. Given a spatial embedding of the data, in terms of a scatter plot or graph layout, we propose a semantic lens which selects a specific spatial and attribute-related data range. The lens keeps the selected data in focus unchanged and continuously deforms the data out of the selection range in order to maintain the context around the focus. Specific deformations include distance-based repulsion of scatter plot points, deforming straight-line node-link graph drawings, and as varying the simplification degree of bundled edge graph layouts. Using a brushing-based technique, we further show the applicability of our semantic lens for scenarios requiring a complex selection of the zones of interest. Our technique is simple to implement and provides real-time performance on large datasets. We demonstrate our technique with actual data from air and road traffic control, medical imaging, and software comprehension applications.

Index Terms—Semantic lenses, magic lenses, graph bundling, attribute filtering

In recent years, the amount of data which information visualization techniques are confronted with has increased massively, whereas display sizes have remained largely identical. Infovis techniques address this challenge in two main ways. First, datasets are simplified by clustering or subsampling, so they deliver manageable data amounts with respect to available screen size. Secondly, mapping techniques maximize the amount of information displayed per screen space area, or information density.

However effective, techniques of the second type create additional challenges to explorative user interaction. Consider the case of dense node-link layouts or multivariate datasets displayed as scatterplots or parallel coordinates. Such techniques create significant amounts of *overlap* between the drawn elements (points or edges). This simplifies the resulting visualization by reducing the number of perceived elements. However, overlaps make it harder to explore the dataset: In typical 2D visualizations, it is hard or even impossible to see what is hidden 'under' the front-most elements, even when using transparency. Hidden elements cannot be easily selected and/or brushed over without additional interaction effort. We have the situation of a compact visualization (desirable from the viewpoint of scalability and, optionally, clutter reduction) which is suboptimal for interactive exploration. Finally, there are use-cases when a given dataset may be best understood by using several layouts, one for each aspect being examined. Displaying a one layout in separate linked views of all layouts can be suboptimal as it increases the effort required from the user to correlate between the different views.

In this paper, we present MoleView, a framework for interactive exploration of large *element-based plots*, which are sets of discrete data elements, each with several data and/or position (layout) attributes, which are visualized using a single, rather than multiple, views. Examples thereof are node-link layouts, (multidimensional) scatter plots, and images. Our contributions are as follows. First, we extend the well-know semantic lens with a range-based attribute filter to select a 'data layer' at a user-defined point, *i.e.* a set of data elements falling within the lens' position and attribute filter values. Instead of hiding the elements in the lens which fail passing the attribute filter, we use a dynamic re-layouting technique to smoothly push these away from the lens, or pull them back, hence the name of our technique. Second, we extend our data-driven deformation idea to explore bundled graphs. Given a bundled and unbundled version of the same graph, we use the MoleView to control the bundling strength *and* which edges get bundled at a certain location. In this way, users can explore bundled graphs (*e.g.* dig into a bundle to extract edges of interest based on attribute value) or, conversely, interactively simplify a given layout by bundling uninteresting edges. Finally, we extend the semantic lens concept for the task of exploring a dataset by the smooth animated interpolation between two completely different layouts of the same data, using as example the exploration of two-dimensional scalar images. Our technique has just a few parameters which are simple to control by end users, can be efficiently implemented to provide real-time interaction on large datasets, and can be easily incorporated in existing Infovis data exploration applications.

The structure of this paper is as follows. Section 1 presents related work in the area of semantic lenses for attribute-based exploration. Section 2 describes the principle of the MoleView technique and its three different modes on utilization (elements, bundles, and dual-layout), and illustrates our technique in practice on a range of real-world datasets. Section 3 discusses the presented technique. Finally, Section 4 concludes the paper with future work directions.

1 Related work

Related work in Infovis falls within several areas, as follows.

Magic lenses: The Magic Lens introduced the idea of locally modifying a screen region based on a user-selected operator [2]. Originally used for modifying the graphics appearance and/or editing the properties of shapes at a focal point, the Magic Lens was subsequently extended to allow more complex operations such as complex effect compositing and interactive lens parameter editing [1]. Tangible magic lenses extended the base concept to allow users to 'slice' through, or zoom in, layered 2D or 3D datasets by interactively moving a 3D tracked physical planar object (the lens) which is either rigid [24] or flexible [15]. Nonlinear projection was added to magic lenses to deform 3D scenes as if seen through a cylindrical or spherical lens, working fully in image space, *i.e.* without access to the actual 3D scene [34].

Semantic lenses, focus and context, and deformation techniques: The dust and magnet technique allows users to de-clutter large scattered plots by placing several data-attribute-driven 'magnets' in the display space and moving data points close to them based on the points' attributes [35]. This metaphor is somewhat similar to the preset controller [29] which is, however, used for the inverse operation of synthesizing data values based on the distance of a cursor to several data-attributed presets. The bundled graph visualization presented

- *C. Hurter is with DGAC-DSNA, Toulouse, France, E.mail: christophe-hurter@aviation-civile.gouv.fr.*
- *O. Ersoy and A. Telea are with the University of Groningen, the Netherlands, E.mail: o.ersoy@rug.nl, a.c.telea@rug.nl.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

in [10] for comparing software hierarchies proposes a circular and a line-based lens which allow users to interactively select a bundle of interest by drawing and/or brushing over the displayed graph. However, no deformation is used here: focus+context is reached by color-based highlighting the selected edges. In a different context, Niels *et al.* visualize vessel movements (trajectories) on a geographical map using a blending technique which groups close trajectories into smooth shaded shapes [30]. Overdraw is eliminated as the dataset is shown as a continuous shaded map. A simple form of semantic lens is used to emphasize specific trajectories, *e.g.* slow moving ships, by tuning the shading and blending parameters. However, spatial deformation is not used to declutter trajectories, since position data is deemed too important to be altered.

Deformation techniques are used for visualizing large datasets by locally changing the underlying spatial layout of the data elements in order to dedicate more space to important data elements than to less important elements. Many variations have been proposed from the original fisheye view [7]. For data tables, the table lens locally distorts the Cartesian cell layout to give more space to specific table rows or columns [20]. For node-link layouts, techniques include local edge deformations, or re-layouts, such as the EdgeLens and its variations [33], and selective edge hiding based on attributes at the position of a user-specified focus point. The local edge lens and bring-neighbors lens of [28] are variations of EdgeLens which remove edges between nodes within a focus zone (lens) and pull nodes connected to nodes-in-focus within the lens, respectively. Edge plucking allows the user to explicitly drag groups of edges away to clarify cluttered zones and/or specify nodes or edges to be left unmoved [32, 31]. However effective, edge plucking requires a certain amount of manual effort. Link sliding and 'bring & go' techniques [16] assist the exploration of node-link diagrams by constraining the user-controlled focus point along a given path in a snap-to-edge manner and moving nodes connected to a node of interest close to that point. Fisheye techniques have also been proposed for trees [28, 8].

Edge bundling techniques trade off clutter for overdraw in the visualization of large graphs by geometrically grouping spatially close edges into so-called bundles. Bundling techniques include hierarchical edge bundles (HEBs) [9], geometric-based edge bundles [5], flow maps [19], force-directed edge bundles [11], and Voronoi-based edge bundles [13]. The visualization is simplified by creating additional empty space. However, overdraw, or edge congestion, makes interactive selection of specific edges difficult [32]. Since many edges overlap, local interaction techniques such as edge plucking are less applicable here below bundle level. The 'digging lens' presented in [26] partially addresses this problem by thinning overlapping bundles at the focus location to allow one to see and/or select bundles obscured due to the inherent overdraw.

Within the large body of work and variations of lens techniques, we frame our contribution as follows:

1. *position and data*: we generalize semantic lenses to work on combined position and data attributes rather than on position or data only, as present in most existing lens applications;

2. *lens shape:* we generalize the lens from a fixed or parameterized shape (as present in existing work) to arbitrary 2D shapes which are interactively specified by the user via direct painting;

3. *animation:* we use smooth animation to continuously deform elements within the lens, for any 2D lens shape;

4. *dual layout:* we generalize the deformation to interpolate between two different spatial layouts of a given dataset, apart from repelling elements based on distance to a focal point;

5. *element types:* we propose a single lens principle and implementation for points, pixels, graph edges, or edge bundles, or any other element that has position and data values.

2 MoleView principle

The principle of MoleView is as follows (see also Fig. 1). As input, we consider a dataset $D = \{s_i\}$ consisting of a set of data elements s_i which all have 2D layout positions $L = \{p_i = (x_i, y_i) \in \mathbf{R}^2\}$. Examples thereof are scatter plots, where s_i are data points; images, where s_i are pixels with color information; and node-link graph drawings, where s_i are nodes, edge control points, entire edges, or entire edge bundles. Any other dataset can be considered as long as it provides 2D position information. Within the given layout, the positions of different elements can overlap, *e.g.* in the case of (bundled) graph drawings (in which case clutter and overdraw are an issue), or not *e.g.* in the case of images. Each s_i can have an application-specific attribute vector $v_i = \{v_{ij}\}$. For simplicity, we next consider only numerical attributes $v_{ij} \in \mathbf{R}$. However, the MoleView principle applies equally well for other attribute value types.

When exploring a 2D rendering of D, users first define a so-called *focus* zone $Z \subset \mathbf{R}^2$. Our central goal is to support tasks which involve exploration of the spatial structure and data attribute distribution of elements $s_i \in D$ within the focus zone (*i.e.* $p_i \in Z$). The provided support is offered in terms of a semantic lens applied on the zone of interest. Our lens combines a flexible, easy to use, animation-based mechanism for specifying the focus zone, containment of data elements in the focus zone, and attribute values to explore, and also the type of spatial deformation applied to the data elements in and/or outside the lens.

We guide the design of our lens by the following:

- exploration should use a *single* view rather than linked views;
- the zone and attribute range of interest should be easily specifiable by simple mouse-driven operations;
- the lens should address the overdraw problem in dense visualizations by allowing users to 'see' behind the front-most elements;
- the lens should provide a focus-and-context metaphor on the dataset D. Changes to the layout L of D should be *smooth* so that users maintain their mental map when using the lens;

The general mechanism proposed is as follows. First, we select the set of data elements $D^Z \subset D$ which are spatially within Z. Spatial containment is determined by the desired effect and type of data elements, *e.g.* points or curves. Secondly, we filter D^Z to a subset of data elements D^{sel} which are within the attribute range of interest A. Like for spatial containment, attribute selection can involve different types of filters for different tasks. We call the set $D^{filt} = D^Z \setminus D^{sel}$ the set of filtered elements, *i.e.* elements that fall in the lens spatially but not data-wise. The most important step is the third one: We apply a smooth, time-animated, spatial deformation $\Delta : \mathbf{R}^2 \times \mathbf{R}^+ \rightarrow \mathbf{R}^2$ from the original layout $L^{filt} = \{p_i \in \mathbf{R}^2 | s_i \in D^{filt}\}$ of the elements in D^{filt} to yield a new layout $L^{filt}_{new} = \Delta(L, t)$. The time parameter $t > 0$ controls the animation of the deformation, *i.e.* morphs in both directions between L^{filt} and L^{filt}_{new} as the lens is activated, respectively deactivated. Suitable choices of the deformation function Δ allow us to perform decluttering, selective fisheye-like exploration on specific data elements, bundled graph exploration, and also correlation of data elements between different layouts.

We next detail three different instances of the MoleView lens principle outlined above: element-based exploration (Sec. 2.1), bundle exploration (Sec. 2.2), and dual-layout exploration (Sec. 2.3).

2.1 Element-based exploration

In this mode, we consider the exploration of a dataset D whose elements s_i have the minimal amount of information: position p_i and an attribute value v_i. We first define the zone of interest Z as a distance field $D_Z(P) : \mathbf{R}^2 \rightarrow \mathbf{R}^2$. The distance field D_Z is defined using a so-called *control set* $P \subset \mathbf{R}^2$, as follows. First, we compute the distance transform $DT_P : \mathbf{R}^2 \rightarrow \mathbf{R}_+$ [3]

$$DT_P(x \in \mathbf{R}^2) = \min_{y \in p} \|x - y\| \quad (1)$$

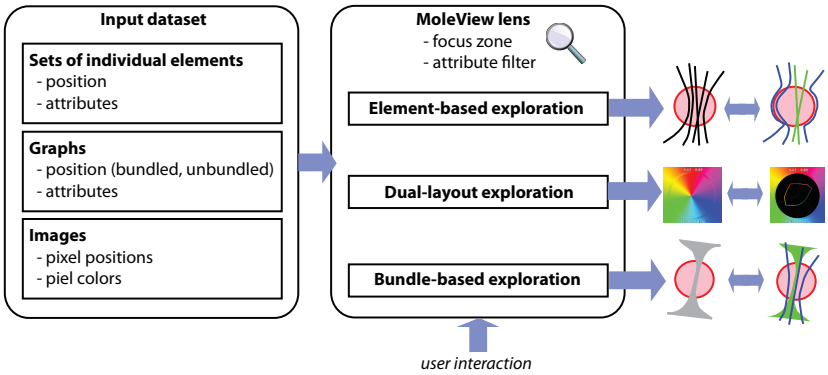

Fig. 1. MoleView interactive exploration pipeline

Given DT_P, Z is simply the level set of DT_P at a user-specified distance $\delta > 0$. Hence, we select all data elements *spatially* falling within Z as

$$D^Z = \{s_i \in D | DT_P(p_i) \leq \delta\} \tag{2}$$

If P is a compact set, then Z is also be compact. However, this is not a constraint – the set P can be any collection of points, lines, or surfaces in 2D. Computing D^Z is simple, no matter how complex the the data element shapes are: We render a shape and apply the point-in-region test (Eqn. 2) when visiting each rendered pixel, an operation efficiently supported by graphics hardware.

Given D^Z, we next select the elements $D^{sel} \subset D^Z$ which are within the zone of interest *and* also have attribute values of interest. In this paper, we use attribute-range selection

$$D^{sel} = \{s_i \in D^Z | v_i \in [v_{min}, v_{max}]\} \tag{3}$$

Other attribute tests can be substituted easily without affecting the implementation or ease of use of our method. The spatial and attribute tests (Eqns. 2 and 3) can be done in a single rendering pass.

The element-based exploration works now as follows. The user specifies the control set P by direct interaction, *i.e.* brushing in the visualization using the mouse. In the simplest case, one selects one or more screen points which will form P, similar to [33]. In this case, DT_P is a superposition of point radial distance functions. The size of the zone of interest δ is via the mouse wheel with a modifier key (Control). More complex interactive specifications of P, yielding more complex distance transforms DT_P, are described in Sec. 2.4. Apart from P, the user also specifies an attribute filter to select elements based on their data values. For the filter in Eqn. 3, we specify the range $[v_{min}, v_{max}]$ by moving the mouse wheel.

The MoleView comes now into action: We keep the points D^{sel} which fall spatially and data-wise in the lens at their original locations p_i and define the layout L^{filt} for the filtered points D^{filt} so as to push them away from the exploration focus (see Fig. 2). For this, we move the points $p_i \in D^{filt}$ in the gradient field $-\nabla DT_P$ with a speed $|\mathbf{v}|$ which decreases as points get close to the lens border and further from the control set P. In detail, the motion field $\mathbf{v} : \mathbf{R}^2 \to \mathbf{R}^2$ is defined by

$$\mathbf{v}(x) = -\nabla DT_P(x)\ \lambda\left(\frac{DT_P(x)}{\delta}\right) \tag{4}$$

The function $\lambda : [0,1] \to [0,1], \lambda(0) = 1, \lambda(1) = 0$ lowers the speed, *i.e.* decelerates points, as they get close to the lens border. In practice, exponential decaying profiles give smooth animation results. The advection implicitly yields a deformation $\Delta(t)$ which gradually pushes points away from P and slows them down at the lens border. Different speed profiles as function of the distance to P can be easily substituted.

The advection in Eqn. 4 is applied when the lens is activated by mouse clicking and is done as long as the mouse button is kept pressed. During this period, we continuously update the position of the points in D^{filt} and redraw them, thereby creating a smooth animation. As the user changes the control set by moving the mouse, points keep moving as they enter into, or exit from, the zone of interest. When the lens is deactivated by mouse button release, we change $\mathbf{v}$ to an attraction field $\mathbf{V}$, defined at the current location of the displaced points p_i^{disp} as

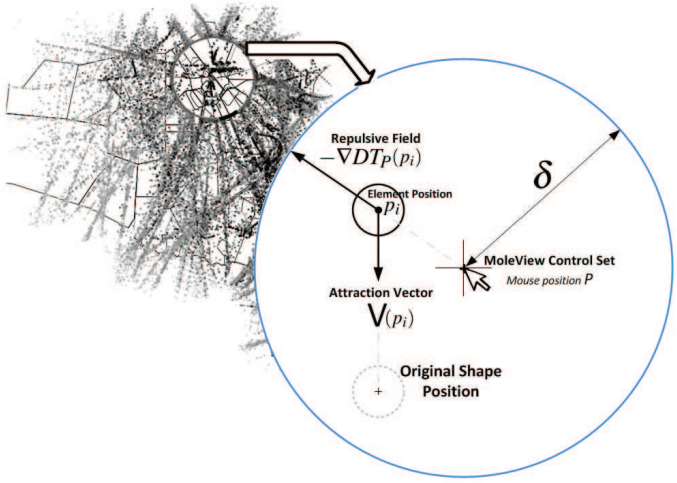

Fig. 2. MoleView element-based exploration mode

$$\mathbf{V}(p_i^{disp}) = p_i - p_i^{disp} \tag{5}$$

where p_i are the point positions before displacement. The effect is that the displaced points smoothly go back to their original positions with decelerating speed, thus reversing the lens effect.

For additional cues, we change the rendering of the displaced elements p_i using $DT_P(p_i)$ by linearly interpolating their transparency between a low value α_{min} at $DT_P = 0$ and a maximal value $\alpha_{max} = 1$ at $DT_P = \delta$, *i.e.* on the border of Z.

Point dataset example: We first consider a 2D point plot of a multivariate dataset using multidimensional scaling (MDS) [18]. The points are text documents placed on the 2D plane so as to reflect the similarity of topics they contain. Document similarity is computed using a cosine-based distance between term vectors extracted from the documents' text [22]. Document topics, found by the classification algorithm underlying the MDS layout, are saved as point data attributes. Due to overdraw, it is hard to see which are the point topics within a given spatial region. This insight is important for MDS plot users, *e.g.* to detect data points which are close to a topic classification border, and for MDS algorithm designers, to assess the algorithm ability to separate different topics.

Figure 3 shows the element-based lens applied to this dataset. The selected attribute range-of-interest matches the purple-colored topic.

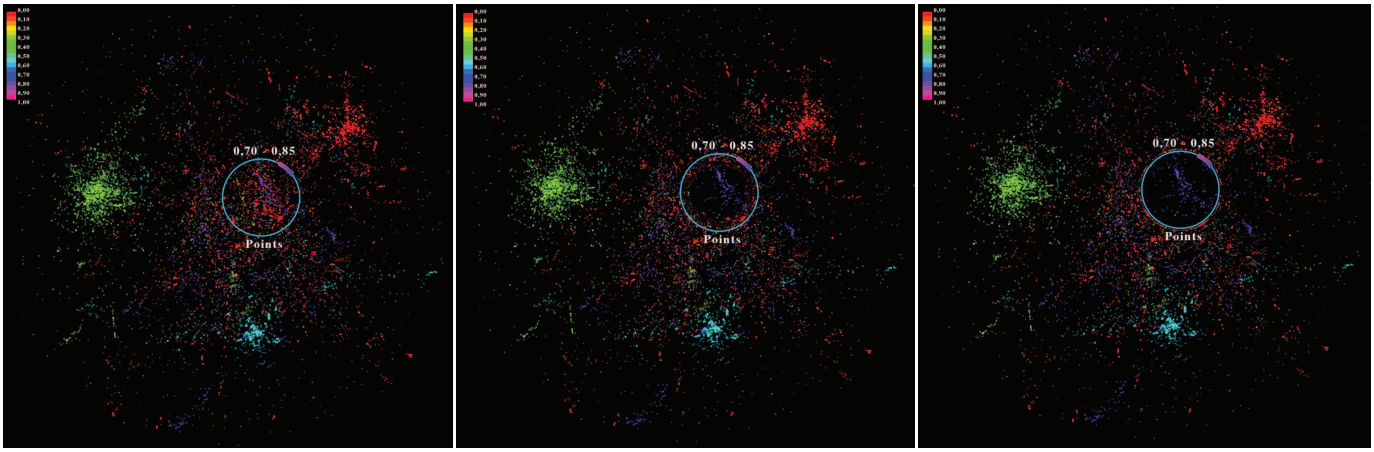

Fig. 3. Element-based exploration of an MDS plot for text documents. Colors are document topics. Points outside the range of interest are gradually pushed to the lens border

When the lens is activated, points outside this topic are smoothly pushed towards the lens periphery, while points within the topic stay unchanged. By changing the attribute range with the mouse wheel, we can browse through the topics overlapping at a given location. Points are pushed or attracted with respect to the lens center as they exit, respectively enter, the range of interest, yielding a sequence of smooth transitions, which helps understanding the image.

Trail dataset example: Our second example dataset is a set of trajectories (trails) whose end points indicate airport locations in France. Trails are flight routes between airports, recorded by air traffic authorities (17275 flight routes) [12]. Each trail is a sequence of points with geographical and altitude data at the respective location. Altitude is visualized by color mapping. Note that this dataset is not, strictly speaking, a graph since trails do not always share start and end points.

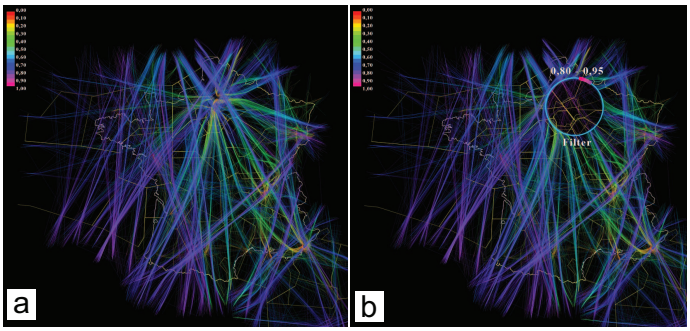

Fig. 4. Flight trails dataset (a) and element-based MoleView lens (b)

Rendering the entire trail set with altitude-colored edges yields an image of very limited usefulness, given the high data occlusion and clutter (Fig. 4 a). An important task here is to find flights with a certain altitude, or altitude variation, over a given spatial region, *e.g.* high-altitude flights, or take-off and/or touch-down flight segments [12]. We could use the technique of Niels *et al.* [30] to reduce clutter, but this would not address the specific task of emphasizing specific flight segments. Also, the method in [30] uses blending to eliminate overdraw, which makes it hard to see individual flight routes.

Figure 4 b shows the element-based MoleView on the flight dataset. We select a circular zone of interest by moving the lens to the desired location. Next, we tune the radius and altitude range for the zone of interest using the mouse wheel. The selected altitude range $[v_{min}, v_{max}]$ is shown by the colored bar on the lens's periphery, which moves around the center as the mouse wheel is turned. As we change these two parameters, flight routes are dynamically pushed to the lens periphery or brought back to their original position. The edge control points are moved smoothly the gradient field of DT_P, which yields a smooth visualization, allowing to follow how edges are filtered in or out from the lens. Overall, edges continuously move in or out of the lens as parameters are changed (see the video material). Edges which are selected in the lens stay unmoved, which makes them easy to spot. The obtained effect reminds of a mole pushing earth (data elements) around as it digs at several locations, hence the name for our technique.

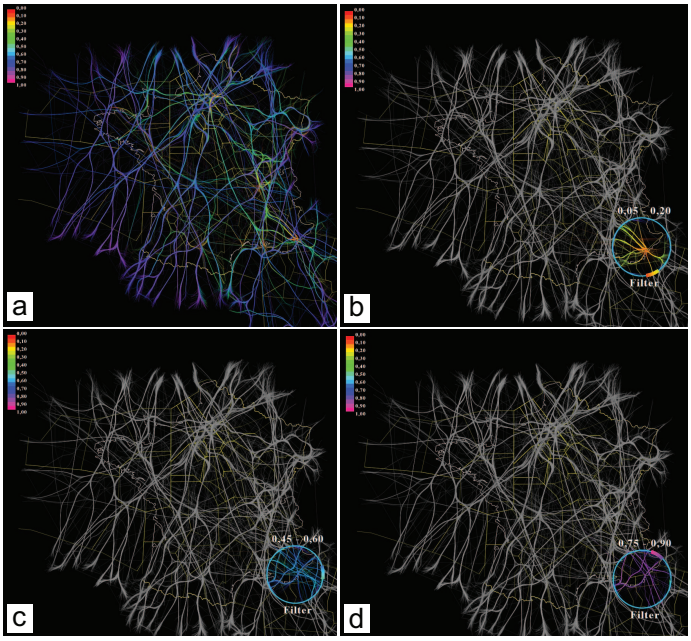

Fig. 5. Bundled flight trails (a). Attribute-based MoleView lens for three altitude levels (b-d)

Bundled graph example: We next show element-based exploration for bundled graphs. Data elements s_i are individual control points of the bundled edges. Figure 5 a shows the graph in Fig. 4 bundled by the method presented in [6]. Any other bundling methods can be used equally well, *e.g.* [11, 5, 13]. Compared to the unbundled view (Fig. 5 a), bundling reduces clutter and allows us to spot groups of close flight routes. However, w now cannot see the *altitudes* of these flights, *e.g.* if flight connection patterns captured by the bundles are similar or different for different altitudes, given the inherent overdraw. With the MoleView, we select a zone of interest around an

agglomeration and push control points for edges in that area matching our altitude filter outside of the bundle. The effect is similar to locally bundling edges within the desired attribute range. Figure 5 b-d show this for three altitude ranges (low, medium, and high) at the same location. We additionally emphasize the selection effect by rendering selected elements D^{sel} with their colors as set by the original visualization and desaturate the elements in D^{filt}. We now see that the bundling patterns of these flights are different, *i.e.* plane routes group differently on altitude. The exploration above is useful in answering questions such as whether a certain group of flights (bundle) contains flights of a specific altitude range. If the graph would encode a software system structure, like the one in Fig. 10 (discussed further), the question addressed would be whether a given system-to-system connection contains dependencies of a given type.

Image data example: Figure 6 show the element-based lens applied to image data. The elements of our dataset D are pixels in an image. A pixel with image (x,y) coordinates is attributed by its grayscale or color value. Images (a-c) show the lens applied to an ultrahigh-resolution angiography image of the human eye [14]. The attribute filter was selected to retain the bright pixels corresponding to important blood vessels and push the darker pixels away from the focus of interest. The three images show how filtered pixels are pushed away, revealing the blood vessels in context. Images (d-f) show the lens applied to a color-coded image of the traffic in Lisbon at night [4]. Green hues show relatively slow moving vehicles. This time, the attribute filter was set to work on hues, retaining the green range. The three frames reveal the slow motion traffic close to the focus of interest. However, the spatial map context is preserved, as filtered pixels are gradually pushed away from the focus (or brought back in, when releasing the mouse). In contrast, traditional value-based filtering would not preserve the context but abruptly eliminate elements out of the attribute range of interest from the visualization.

2.2 Bundle-based exploration

Our second scenario, called *bundle-based exploration*, considers the more specific case of a dataset D representing a bundled graph. Data elements s_i are now either individual edge control points, entire edges, or entire bundles. Such datasets can be obtained using one of the many available bundling methods [9, 5, 11, 13]. As explained in Sec. 1, bundled layouts provide simplified visualizations of large graphs but also increase overdraw. This makes it difficult to understand which edges exactly are part of a given bundle, unless the bundling is data-driven, which is not the case in all examples we are aware of. For instance, hierarchical edge bundles (HEBs) used in software visualization have proved of limited success beyond assessing the overall modularity of a system [9, 10]. Such edges are annotated with data attributes *e.g.* type of dependency (call, uses, inherits, includes, reads, writes, owns), or number of times and moment when a function gets called. Real-world software comprehension tasks such as reverse-engineering, architecture quality assessment, and performance assessment need to understand how such attributes are distributed over the edges in a bundle.

Given a control set and zone of interest defined by the user (Sec. 2.1), we consider a bundled layout L^b and an unbundled layout L^u of the explored graph . We now apply our semantic lens pipeline (Sec. 2) by setting the original and deformed layouts L and L^{filt} to the bundled and unbundled layouts L^b and L^u respectively. The deformation Δ smoothly interpolates between the two layouts rather than moving points away from the zone of interest as for the element-based exploration (see Fig. 7):

$$\Delta(t,p_i) = \lambda(t)L^b + (1-\lambda(t))L^u \qquad (6)$$

Just as for the distance-field-based deformation (Eqn. 4), different speed profiles λ can be used to control the animation. The attraction term $\mathbf{V}$ (Eqn. 5) is identical to the element-based exploration. When the lens is deactivated, displaced elements snap back smoothly from the positions in one layout to the positions in the second layout.

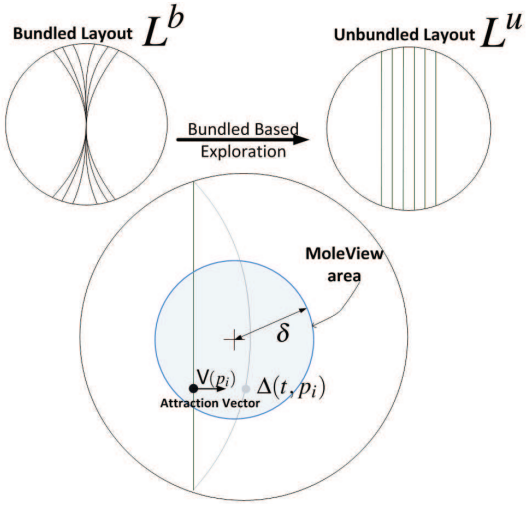

Fig. 7. MoleView bundle-based exploration mode

Point-level exploration: Figure 8 left shows the bundle-based lens for the flights graph. Compared to Fig. 5, filtered elements are now moved towards their unbundled locations rather than being pushed towards the lens periphery, yielding a smooth local transition between the bundled and unbundled layouts for the selected edge portions. Since the lens uses both position and attribute values, this is different than simply unbundling the *entire* bundle in the zone of interest. The reverse scenario where selected elements are moved towards their unbundled layout is obtained by applying the deformation (Eqn. 6) on the set D^{sel} rather than on D^{filt} (see Fig. 8 right). In this case, the lens supports the task of locally showing selected elements in their original spatial context, and filtered elements using the simplified bundled view.

By swapping the layouts L^b and L^u in Eqn. 6 and applying the lens on an unbundled graph, we obtain two complementary effects, *i.e.* we can locally bundle selected elements while leaving all filtered elements at their original locations, or locally bundle filtered elements leaving all selected ones at the original locations. These scenarios are useful when the user wants to keep the original *context* (unbundled graph) and wants to apply the structural simplification (bundling) on the *focus* zone. Figures 8 illustrate the above scenarios.

Edge-level exploration: We can also apply our lens on entire edges. Elements s_i of our dataset D are now whole edges rather than edge control points. The method stays the same, but we now apply the deformation (Eqn. 6) to *all* control points of edges in the lens rather than to points in the lens. The lens has now bundles (or unbundles) an entire set of selected edges (Fig. 9). Here, flights through the Paris area are smoothly bundled, while other flights are drawn at their original locations. This is useful when one wants to explore a set of trals in detail, *i.e.* see them in their entirety in their original positions, rather than applying unbundling to a spatially confined region.

Bundle-level exploration: At the coarsest level, we can apply our lens on entire *bundles*. For this, we need explicit bundle identities as groups of edges. Given a bundled layout, we compute such edge groups, or clusters, using the bottom-up hierarchical agglomerative clustering scheme based on Euclidean distance between edge control points in [26]. This gives a partition $C = \{c_i\}$ of the edge set in the input graph into clusters which contain edges which we visually perceive as a bundle. Other edge clustering schemes can be used, if desired.

Given such a partition C of the edge set, we can now directly use our lens on entire bundles by considering a whole bundle as a data element s_i in any of the exploration modes described above. The advantage is now that users can brush a single branch of a bundle and directly

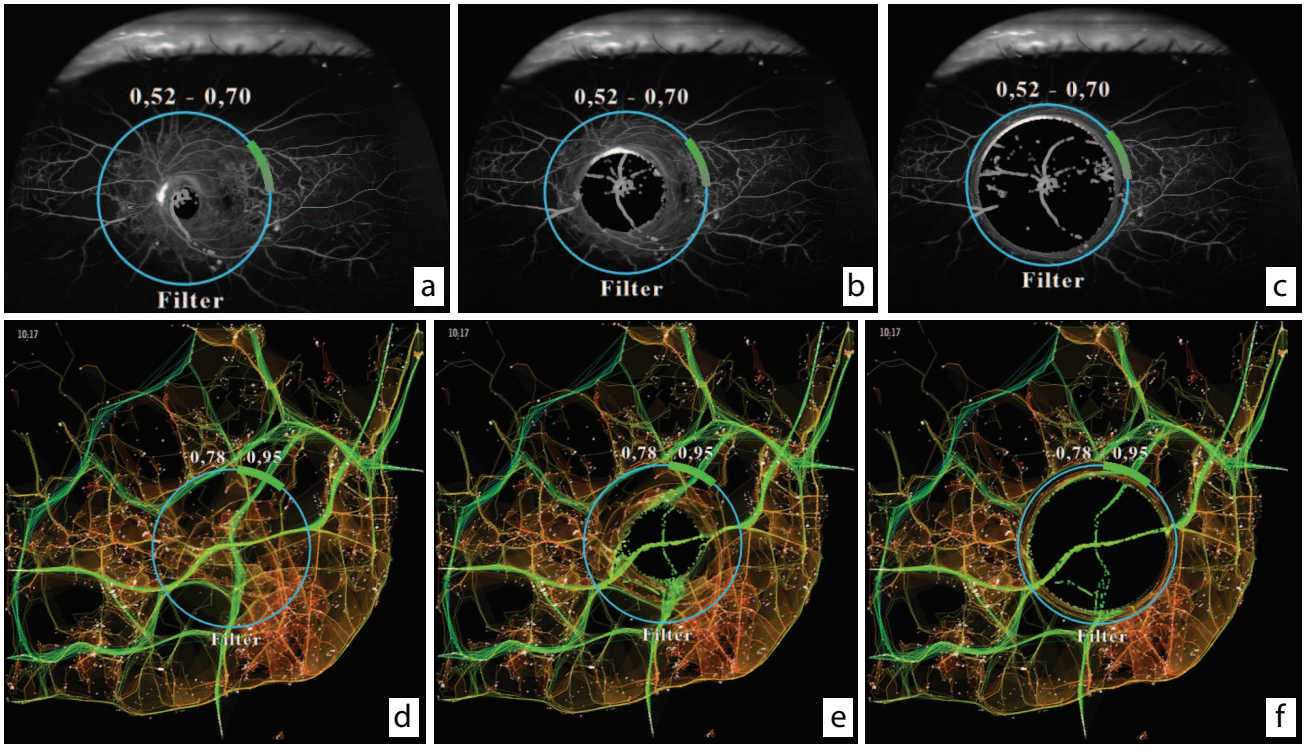

Fig. 6. Element-based MoleView applied to grayscale angiography image (a-c) and color-mapped traffic speed image (d-f)

explore the entire bundle. Figure 10 shows this for a radial layout depicting the structure of a software system (nodes are software entities, while edges are dependencies). Bundles are explicitly identified using edge clustering and assigned different colors (a). Alternatively, this can be done by clustering edges relating specific coarse-scale subsystems, if a software containment hierarchy is available. Local unbundling reveals the structure of a specific zone of interest (b). This can be useful *e.g.* if edges are colored on another attribute than the one used for bundling, *e.g.* edge type, as it allows one to explore the different types within a bundle, without modifying the overall bundled layout. If we consider entire edges as elements, the lens can be used to unbundle one or more entire bundles under the lens (b). Finally, we can combine the local and whole-edge unbundling effects to achieve a two-stage unbundling effect (c). When animated, this gives additional cues as to the identities of the bundles brushed by the lens, but keeps clutter minimal within the lens area. This is useful *e.g.* when we do not use colors to show bundle identities and users are interested in seeing all edges within a certain bundle passing through a spatial region.

2.3 Dual-layout exploration

Our third scenario, called *dual-layout exploration*, considers a dataset D which is explored via two completely different spatial layouts. An example thereof are images, seen either as pixels arranged according to a Cartesian layout or histogram layout. The two layouts serve different purposes: the Cartesian one allows finding specific shapes; the histogram shows data value distributions. Typical visualizations interested in above aspects use two views linked via brushing and/or selection. However, as outlined earlier, a two-view mode is suboptimal as it requires users to explicitly correlate two images. This applies even more so if correlation is needed only at certain zones of interest.

We can use our semantic lens (Sec. 2) to address the correlated exploration of datasets which use different layouts for different views on the data. To illustrate this, we consider two layouts of an image: the inherent Cartesian layout L^C of pixels in the image, and a polar coordinate plot L^P with hue mapped to the angular axis and saturation mapped to the radius. Value (luminance) plays the role of the attribute values v_i of our data elements which are affected by the attribute filter.

To apply the semantic lens, we define a time-dependent deformation $\Delta(t)$ which links the positions of corresponding data elements (pixels) p_i^C and p_i^P in the two layouts L^C and L^P respectively

$$\Delta(t, p_i) = \lambda(t)p_i^C + (1-\lambda(t))p_i^P \tag{7}$$

Compared to the element-based exploration mode, our goal is now different: We wish to correlate the spatial distribution of data elements in two layouts rather than filter away elements having a certain attribute range. For this, we apply our semantic lens on all elements falling within the zone of interest, *i.e.* $D^{sel} = D^Z$.

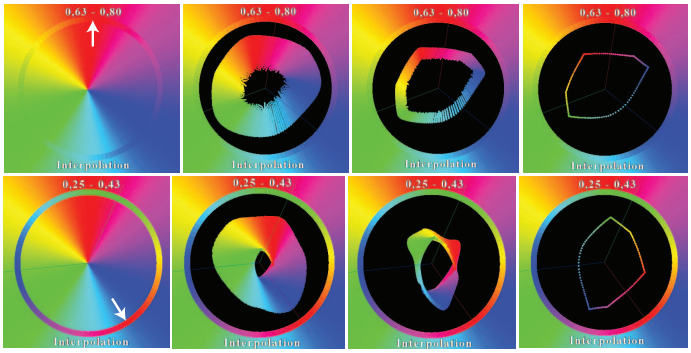

Fig. 11. Dual-layout lens applied to a simple image. Top and bottom rows show different rotations of the HSV space, with origin on the angle axis indicated by the arrow

Figure 11 illustrates the dual-layout on a simple image containing the full color spectrum. The upper row uses a HSV polar layout L_P in which the zero hue value, red, is at the top (as shown by the arrow). When applying the dual-layout lens, pixels are smoothly advected in the deformation field $\Delta(t)$ (Eqn. 7) from their location in the Cartesian layout L_C, *i.e.* original image to their location in the HSV polar coordinate layout L_P. This allows the user to locally query an image and see the hue and saturation distribution over that zone of interest. If we draw the points in L_P using alpha blending, we effectively obtain

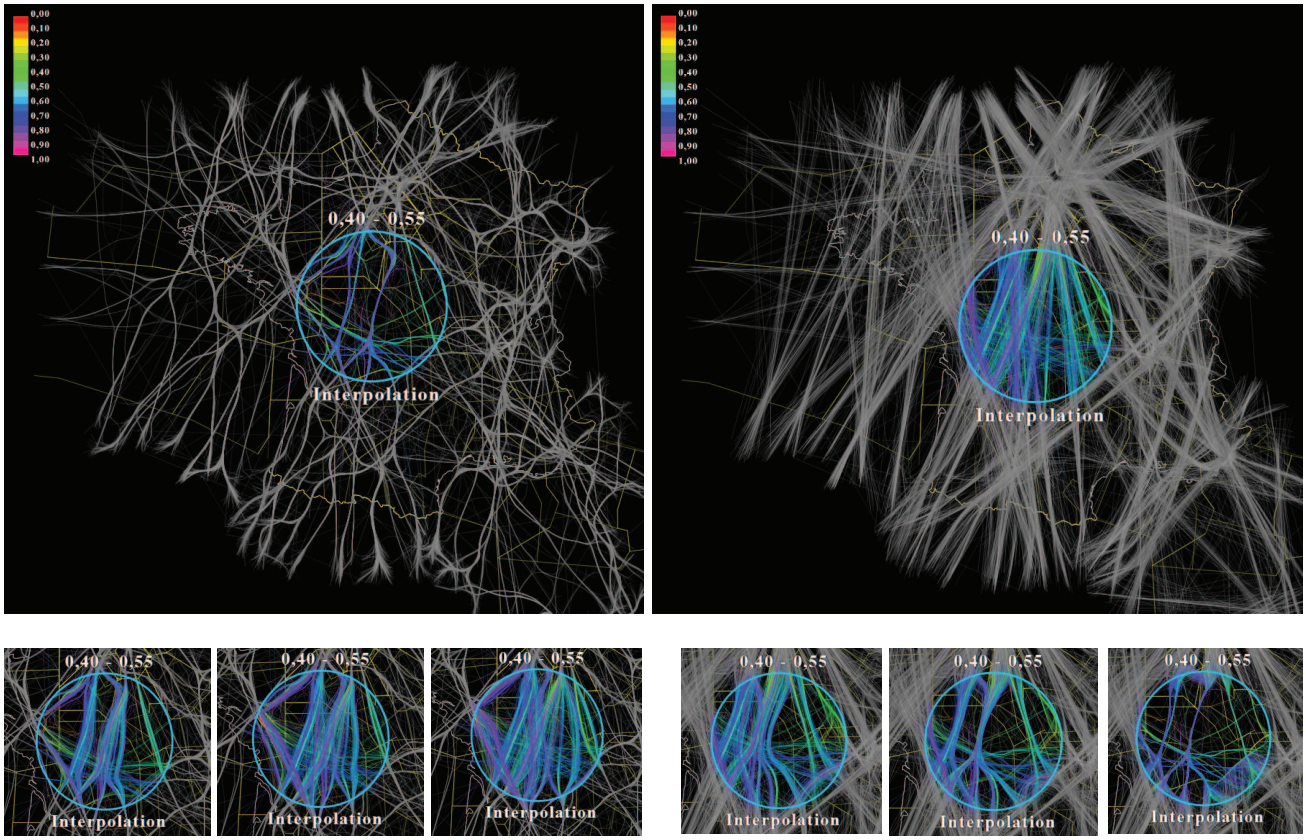

Fig. 8. Bundle-based exploration (Sec. 2.2). Local unbundling (left). Local bundling (right)

a histogram of the hues and saturations of the pixels in the zone of interest.

Figure 12 shows the dual-layout lens applied to two color-mapped scalar fields. The first field (a-d) shows the frequency of lightning occurrences on the surface of the Earth with a heat colormap (cold colors=low frequency, hot colors=high frequency) [17]. Using the dual-layout lens, we see that zones in the geographical areas (b) and (c) have a similar distribution of lightning occurrences: the pixel pattern in the HSV space within the lens is nearly identical. This is not evident from the original image, since the pixels in the two indicated regions have relatively complex color patterns. In contrast, the zone under the lens in figure (d) shows a different pixel color distribution than the zones (b) and (c) – the green-blue 'tail' of the shape we see in the lens in (b) and (c) is now missing. This indicates that this geographical zone has no lightning frequencies corresponding to these value ranges. Again, the original image (a) does not show this – the pixel color patterns in the three regions are looking relatively similar.

The second scalar field (e-g) in Fig. 12 shows a 3D skeleton, or medial axis, of a cow model. The skeleton is computed using the voxel-based method in [21]. Skeleton voxels are colored with their so-called importance with a blue-to-red rainbow colormap. Less important skeleton points (blue) correspond to small-scale object features, *e.g.* the horns or hoof tips. Most important points (yellow and red) correspond to large-scale object features, like the rump. Skeleton importance can be used to simplify the object by pruning away less important points. The skeletonization method in [21] *conjectures*, but does not rigorously prove or disprove, that the importance of skeleton points varies smoothly over small, connected, areas of the skeleton.

We use our dual-layout lens to investigate this hypothesis. Image (f) shows the lens applied to the head region. We see here a continuous blue-to-green curve, which shows that voxels in this region have, indeed, importances which compactly cover the low-to-medium range. Applying the lens to the back rump region (g) shows, as expected, a broader color spectrum, since points in this area have importances spanning from very low (blue) to highest in the model (red). However, this curve is not continuous, but broken in the yellow range. This indicates that there are no voxels here with medium-high importance values, which raises questions on the validity of the conjecture in [21]. Applying the lens to the front rump region (h) shows a similar curve as in region (g). Again, we see small interruptions of the curve, which strengthen our supposition that the conjecture may not be valid. Moreover, we see a red portion in the curve, showing that there are high-importance voxels in this area. Manual direct inspection of the model from different viewpoints such as the one shown in (e), however, does not show such voxels, which potentially may lead analysts to the conclusion that the model's highest-importance region is only located in the back rump region. Given that we worked with this 3D skeletonization method and this specific model for about a year in a different project, this was an unexpected result, which we only discovered using the MoleView lens. Close examination revealed the answer: the front rump region does, indeed, contain high-importance voxels, but these are hidden from virtually any viewpoint, as they are located precisely at the intersection of several 3D skeletal manifolds which meet in that region, so they are hardly visible from the outside. Hence, standard examination of the 3D color-coded voxel set did not reveal these outliers, but application of the MoleView lens did.

2.4 Specification of the zone of interest

The exploration modes described in the previous sections use a simple selection of the zone of interest as one, or several, radial regions determined by user-specified points or foci. Alternatively, whole edges or entire edge bundles that intersect such regions can be selected. However, in more complex scenarios, users are interested to specify zones of interest on a finer-grained, more flexible, level. For example, in the flight visualization, one can be interested to unbundle, or emphasize attribute-based edges, which are part of a given geographical area.

We achieve this by allowing the user to 'paint', or brush, the control set P directly on the screen using the mouse (see Fig. 13), by recording

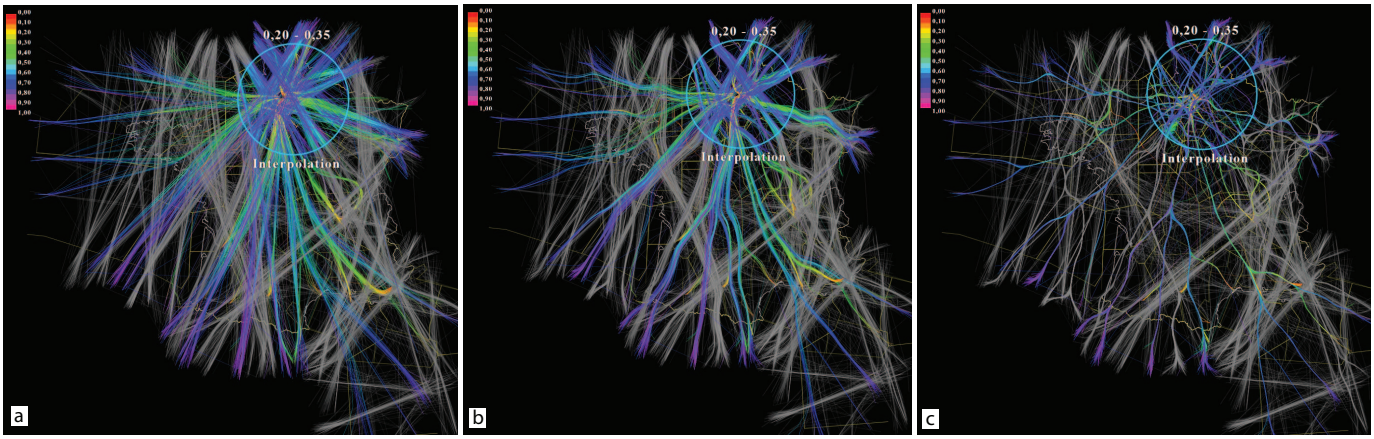

Fig. 9. Smooth bundling of entire flight paths within a zone of interest. The original unbundled dataset (a) is gradually bundled within the zone of interest (b), finally yielding the bundled dataset (c)

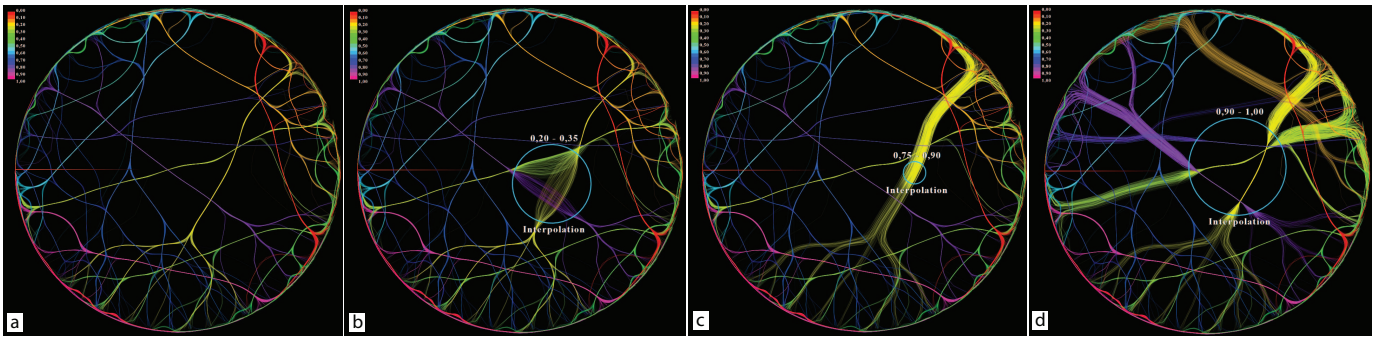

Fig. 10. MoleView applied at bundle level on a software dependency graph using a radial layout. Original bundled graph, with bundles colored by bundle id (a). Local unbundling effect (b). Whole-edge unbundling (c). Combined local and whole-edge unbundling (d)

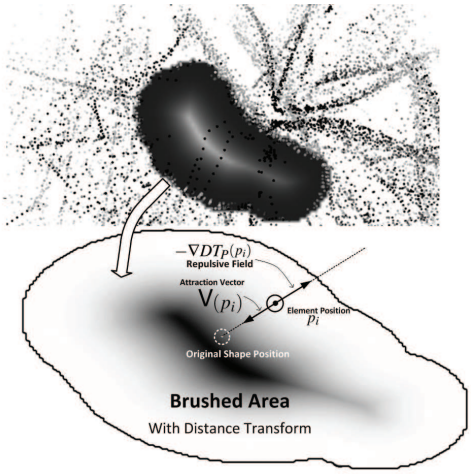

Fig. 13. Interactively painting zones of interest (see Sec. 2.4)

the mouse path on the screen, and using this path as control set P. The remainder of our entire method stays identical, as we can directly compute distance transforms of such pixel paths in exactly the same way we do it for individual points (Sec. 2.5).

Figure 14 illustrates the specification of zones of interest for the flight dataset. Here, the user is interested in seeing low-altitude flights that pass over geographical zones located close to some main airports in France. Air traffic controllers are particularly interested in such flight patterns for planning purposes as flight routes can get readily crowded in such zones. In Fig. 14 b, the user has painted the areas of interest directly on the visualization. Using the element-based exploration lens (Sec. 2.1) smoothly pushes away the mid-to-high altitude uninteresting flights (green), revealing the low altitude critical flights (purple). The distance transform profiles for the brushed zones are shown in grayscale (black=high distance, white=low distance to the control set). Using the same mechanism, arbitrarily complex zones of interest can be easily painted, see *e.g.* Fig. 14 c for a freehand example.

2.5 Implementation

The MoleView lens can be efficiently and easily implemented atop of any existing visualization metaphor consisting of several discrete, data-attributed, elements with 2D spatial positions, as follows.

First, we compute the distance transform DT_P of the control set (Eqn. 1, Sec. 2.1) using the augmented fast marching method (AFMM) [27]. The shape on which the AFMM is computed is identical to the control set P, which is interactively drawn by the user, as explained previously. Besides the distance transform, the AFMM also delivers the feature transform of its input shape $FT_P : \mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined as

$$FT_P(x \in \mathbf{R}^2) = argmin_{y \in P} \|x - y\| \quad (8)$$

Since $|FT_P| = -\nabla DT_P$ [27], we can obtain in this way the gradient field we need for deformation with no numerically sensitive operations such as differentiation, regardless of the complexity of the input image. The AFMM efficiently computes the distance and feature transform of an image of 800^2 pixels in roughly 0.25 seconds on a typical 2.5 GHz modern PC. The complexity of the AFMM is $O(N \log N)$ for an image of N pixels. If desired, a significantly faster CUDA-based implementation of distance and feature transforms can be used [25], which provides $O(N)$ complexity and treats images of 800^2 pixels on 0.02 seconds per image on a Nvidia GT 330M. Performance is important when specifying user-drawn zones of interest (Sec. 2.4), since

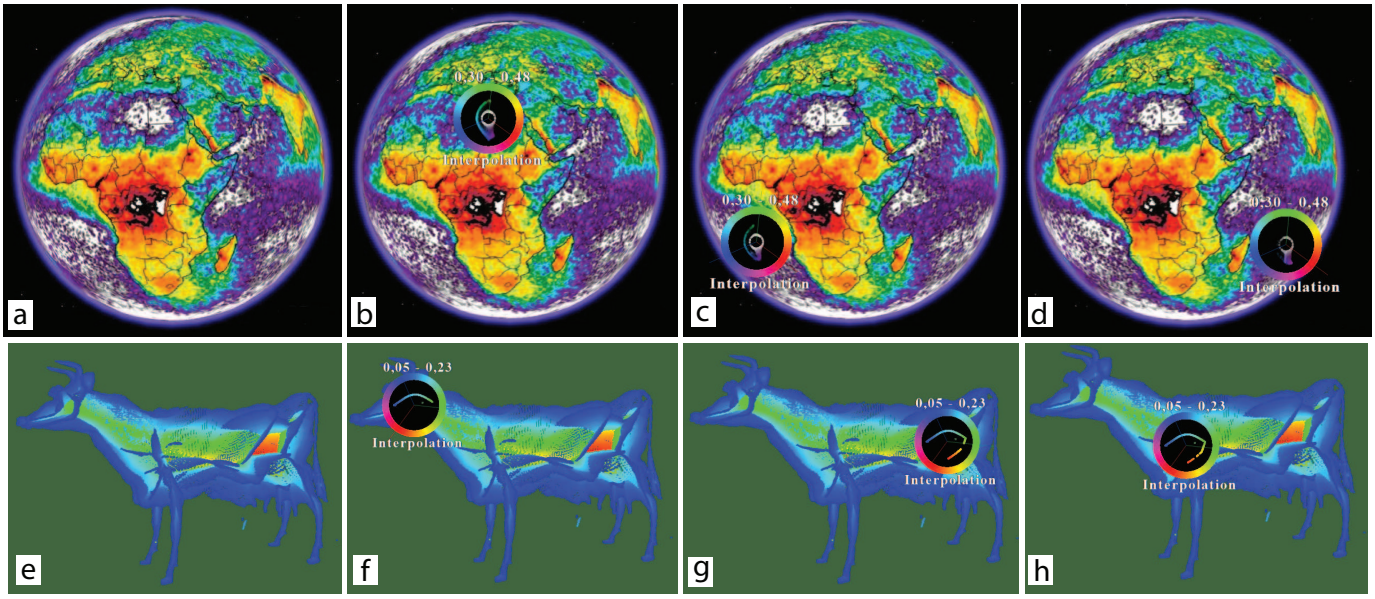

Fig. 12. Dual-layout lens applied to two color-coded scalar field images. Top row: lightning frequency on the surface of the Earth (heat colormap). Bottom row: 3D skeleton color-coded by importance (rainbow colormap)

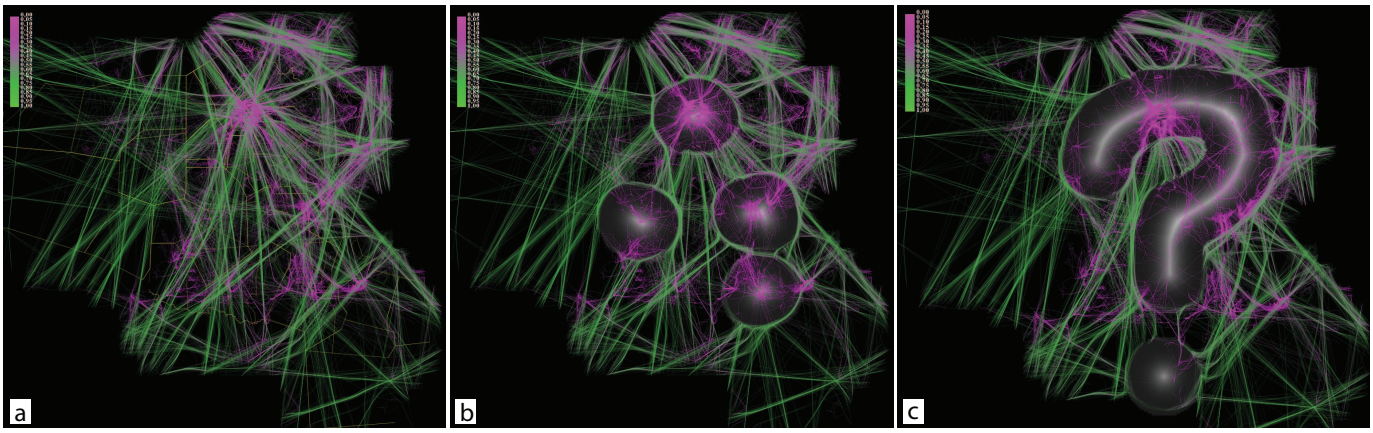

Fig. 14. Interactive specification of a zone of interest (see Sec. 2.4). Flight visualization without lens (a). Focus on low-altitude flights in areas around the main airports (b). Free-form painting of the zone of interest (c). Distance transform profiles are shown in gray

such zones may have arbitrarily complex shape, as compared to the simple set of points shown in Secs. 2.1 and 2.2. Using the above, our entire method can be implemented to achieve real-time frame rates on a typical modern PC for datasets having hundreds of thousands of data elements. For large datasets, implementing the displacements (Eqns. 4, 6 and 7) on the GPU using CUDA is straightforward, as these are independent, simple, point operations.

3 Discussion

Animation is a key element to the effectiveness of MoleView: by continuously (and smoothly) changing the position of the points affected by the lens, users can brush through a dataset and obtain a continuous, smooth, change of the visualization. The continuous effect is also present when the lens is toggled between activated and deactivated states: points smoothly move as affected by the lens at activation, or move back to their original position as the lens is deactivated. This type of motion allows the creation of a focus-and-context effect. As opposed to other techniques, this is realized by position changes in time, rather than just spatial distortions. Hence, even when the user does not move the lens, the visualization changes smoothly. The same holds for situations when the lens is moved.

The MoleView and the bubble variant of EdgeLens [33] produce similar results in particular cases. Specifically, this happens if the control set is a set of discrete, relatively widely spaced, points, and we do not apply the attribute filter. However, there are several differences, as follows. First, MoleView is not specifically limited to decluttering edges in node-link diagrams, but can be applied essentially to any set of discrete elements which have data and 2D position. Examples shown here demonstrate this for bundled and unbundled graphs, scatterplots, and images. For this, the usage of a general advection field, rather than controlling edge shapes using Bézier curves as in EdgeLens, is essential. In particular, the field used to morph an image to its pixel color histogram, is computed by using the two layouts of the image and HSV histogram respectively (Sec. 2.3). Another important ingredient of MoleView is the ability to select the attribute range to act upon. This allows one to explore based on data *and* spatial position rather than spatial position only as in EdgeLens. As such, MoleView and EdgeLens address overlapping, but not identical, use-cases.

Our control set (Sec. 2) is a general subset $P \subset \mathbf{R}^2$, specified *e.g.* by direct painting in the visualization. The lens shape, and its repulsion vector field computed using the feature transform FT_P, yield very different deformation patterns than displacing a set of control points under the influence of a few discrete foci as in EdgeLens. Specifically, FT_P yields a locally smooth field wherever the control set P does not

have strong curvature discontinuities, as known from medial axis theory [23]. Practically, if the user draws P a a set of lines, this field will always be smooth if the lines do not intersect. At intersection points, there is only a null set of discontinuities corresponding to the feature points of the branching points of the skeleton S_Z of the zone of interest Z [27, 23]. For example, if the user draws P as$n$ lines which intersect *exactly* in the same point, we will have n such discontinuities. This poses no robustness or quality problems in practice when advecting elements in FT_P, since these are moved *away* from S_Z.

An attractive aspect of the MoleView set of techniques is that they can be added with minimal intrusion to existing visualizations in a postprocessing phase, *e.g.* without having access to the actual engines which compute multidimensional scaling layouts or bundled edge layouts. In particular, for image data the dual-layout exploration presented in Sec. 2.3 can be used directly on 2D image data generated by other applications, without access to the actual underlying data points or, for the application in Fig. 12-f-h, the 3D voxel data.

Strictly speaking, the bundle-based exploration lens (Sec. 2.2) can be seen as a particular case of the more general dual-layout exploration lens, where the two layouts L^u and L^b co-exist in the same conceptual space. The difference is that the dual-layout lens propose a more aggressive semantic change – it changes the meaning of the space within the lens from a Cartesian (RGB) plot to a polar (HSV) plot. In contrast, the meaning of both the bundled and unbundled layouts is less different. As such, we choose to allow bundled and unbundled data elements to co-exist in the lens area, whereas in the image use-case the lens are shows only one of the RGB or HSV layouts.

4 Conclusion

In this paper we have presented MoleView, a set of interactive lens techniques for the exploration of large datasets rendered as sets of 2D objects. The principle of the MoleView is based on a combination of attribute-based filtering with local displacements of the data points in a force field determined by the zone of interest and dataset layout values. Three exploration modes are presented. The element-based mode repels filtered data points in a distance field, thus unearthing specific data values which may be obscured due to overdraw. The bundle-based mode locally deforms a bundled layout into an unbundled one or conversely, thus helping users to dig into the structure of tight bundles for edges having specific data values. Finally, the dual-layout mode smoothly interpolates point positions between two different layouts which highlight different data aspects allowing the user to correlate between the two data views.

We next plan to extend the exploratory scenarios supported by the MoleView to additional use-cases. For example, the attribute filter can be made to operate on a histogram of the data values in the lens rather than the values themselves, allowing users to select data outliers from a large mass. Secondly, the dual-layout exploration lens principle can be applied to other layouts than Cartesian RGB plots and HSV polar plots, *e.g.* to smoothly interpolate between completely different graph layouts for graph exploration or between different 2D plots which show pairs of dimensions in a multivariate dataset in a single view.

References

[1] E. Bier, M. Stone, and K. Pier. Enhanced illustration using MagicLens filters. *IEEE CG & A*, 17(6):62–70, 1997.

[2] E. Bier, M. Stone, K. Pier, W. Buxton, and T. DeRose. Toolglass and magic lenses: The see-through interface. In *Proc. ACM SIGGRAPH*, pages 137–145, 1993.

[3] L. Costa and R. Cesar. *Shape analysis and classification: Theory and practice*. CRC Press, 2000.

[4] P. Cruz. Boundaries in information visualization - towards information aesthetics, 2010. MSc Thesis, U. Coimbra, Portugal.

[5] W. Cui, H. Zhou, H. Qu, P. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE TVCG*, 14(6):1277–1284, 2008.

[6] O. Ersoy, C. Hurter, F. Paulovich, G. Cantareira, and A. Telea. Skeleton-based edge bundling for graph visualization. In *Proc. InfoVis*, 2011.

[7] G. Furnas. Generalized fisheye views. In *Proc. ACM CHI*, pages 16–23, 1986.

[8] E. Gansner, Y. Koren, and S. North. Topological fisheye views for visualizing large graphs. In *Proc. IEEE InfoVis*, pages 175–182, 2004.

[9] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE TVCG*, 12(5):741–748, 2006.

[10] D. Holten and J. J. van Wijk. Visual comparison of hierarchically organized data. *Comp. Graph. Forum*, 21(4):759–766, 2008.

[11] D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *Comp. Graph. Forum*, 28(3):670–677, 2009.

[12] C. Hurter, B. Tissoires, and S. Conversy. FromDaDy: Spreading data across views to support iterative exploration of aircraft trajectories. *IEEE TVCG*, 15(6):1017–1024, 2009.

[13] A. Lambert, R. Bourqui, and D. Auber. Winding roads: Routing edges into bundles. *Comp. Graph. Forum*, 29(3):432–439, 2010.

[14] H. Lawrence and Y. Kulkarni. Anterior segment and fundus photography, 2011. `emedicine.medscape.com/article/1228681-overview`.

[15] J. Looser, R. Grasset, and M. Billinghurst. A 3D flexible and tangible magic lens in augmented reality. In *Proc. ISMAR*, pages 254–262. IEEE, 2007.

[16] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, and J.-D. Fekete. Topology-aware navigation in large networks. In *Proc. ACM CHI*, pages 320–329, 2009.

[17] NASA Team. Earth lightning map, 2011. `thunder.msfc.nasa.gov/data`.

[18] F. Paulovich, L. Nonato, R. Minghim, and H. Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE TVCG*, 14(3):564–575, 2008.

[19] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. In *Proc. InfoVis*, pages 219–224, 2005.

[20] R. Rao and S. Card. The table lens: merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proc. ACM CHI*, pages 348–356, 1994.

[21] D. Reniers, J. J. van Wijk, and A. Telea. Computing multiscale skeletons of genus 0 objects using a global importance measure. *IEEE TVCG*, 14(2):355–368, 2008.

[22] G. Salton. Developments in automatic text retrieval. *Science*, 253:974–980, 1991.

[23] K. Siddiqi and S. Pizer. *Medial Representations: Mathematics, Algorithms and Applications*. Springer, 1999.

[24] M. Spindler and R. Dachselt. Exploring information spaces by using tangible magic lenses in a tabletop environment. In *Proc. ACM CHI (EA)*, pages 243–248, 2010.

[25] A. Telea. CUDA skeletonization and distance transform toolkit, 2011. `www.cs.rug.nl/~alext/CUDASKEL`.

[26] A. Telea and O. Ersoy. Image-based edge bundles: Simplified visualization of large graphs. *Comp. Graph. Forum*, 29(3):543–551, 2010.

[27] A. Telea and J. J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *Proc. VisSym*, pages 251–259, 2002.

[28] C. Tominski, J. Abello, F. van Ham, and H. Schumann. Fisheye tree views and lenses for graph visualization. In *Proc. Information Visualisation*, pages 202–210, 2006.

[29] J. J. van Wijk and C. W. A. M. van Overveld. Preset based interaction with high dimensional parameter spaces. In F. Post, G. Nielsen, and G. Bonneau, editors, *Data visualization - State of the art*, pages 391–406. Kluwer, 2003.

[30] N. Willems, H. van de Wetering, and J. J. van Wijk. Visualization of vessel movements. *Comp. Graph. Forum*, 28(3):959–966, 2009.

[31] N. Wong and S. Carpendale. Supporting interactive graph exploration with edge plucking. In *Proc. IEEE Visualization (interactive posters)*, 2005.

[32] N. Wong and S. Carpendale. Supporting interactive graph exploration using edge plucking. In *Proc. SPIE*, pages 235–246, 2007.

[33] N. Wong, S. Carpendale, and S. Greenberg. EdgeLens: An interactive method for managing edge congestion in graphs. In *Proc. IEEE InfoVis*, pages 167–175, 2003.

[34] Y. Yang, J. Chen, and M. Beheshti. Nonlinear perspective projections and magic lenses: 3D view deformation. *IEEE CG & A*, 25(1):567–582, 2005.

[35] J. Yi, R. Melton, J. Stasko, and J. Jacko. Dust & magnet: Multivariate information visualization using a magnet metaphor. *J. of Information Visualization*, 4(4):542–551, 2006.