2014 IEEE Pacific Visualization Symposium

Color Tunneling : Interactive Exploration and Selection in Volumetric Datasets

C. Hurter[*]
ENAC,
Univ. of Toulouse, France

A. R. Taylor[†]
Univ. of Calgary, Canada

S. Carpendale[‡]
Univ. of Calgary, Canada

A. Telea[§]
Univ. of Groningen, the Netherlands
Univ. Carol Davila, Romania

Abstract

Interactive data exploration and manipulation are often hindered by dataset sizes. For 3D data, this is aggravated by occlusion, important adjacencies, and entangled patterns. Such challenges make visual interaction via common filtering techniques hard. We describe a set of real-time multi-dimensional data deformation techniques that aim to help users to easily select, analyze, and eliminate spatial-and-data patterns. Our techniques allow animation between view configurations, semantic filtering and view deformation. Any data subset can be selected at any step along the animation. Data can be filtered and deformed to reduce occlusion and ease complex data selections. Our techniques are simple to learn and implement, flexible, and real-time interactive with datasets of tens of millions of data points. We demonstrate our techniques on three domain areas: 2D image segmentation and manipulation, 3D medical volume exploration, and astrophysical exploration.

Index Terms: I.3.6 [Methodology and Techniques]: Interaction techniques—

1 Introduction

Volumetric datasets are found in many fields of science, such as engineering, material sciences, medical imaging, and astrophysics. One of the most used visualization methods for such datasets is direct volume rendering (DVR), which can show all values in the dataset. In contrast, techniques such as isosurfaces or slicing focus on data subsets, which requires users to select *a priori* the structures of interest.

Although DVR does not require an *a priori* selection step, it also comes with one major challenge: occlusion. On two-dimensional display devices, one cannot see more than a single data value per pixel. Yet, there are typically tens, or even hundreds, of such data values along each view ray centered at such a pixel. Hence, discovering patterns of interest hidden inside the data volume can be challenging.

Apart from *a priori* selection of structures of interest, several techniques have been proposed for exploring 3D data volumes. Transfer functions map values along a view ray to RGBA components which are next combined to convey an aggregated insight at each screen pixel. However, to make deep-hidden patterns visible in the final 2D image, transfer functions require careful, and often non-trivial settings. Interactive focus-plus-context (F+C) techniques offer an alternative by allowing users to locally manipulate the geometry and/or appearance of the data volume in order to 'peek' inside it, while keeping the overall spatial context of the entire volume.

In this paper, we extend F+C interactive exploration of 2D and 3D datasets in several directions. We propose a set of linked views that display subsets of data attributes. Example views are 3D DVR plots, 2D scatterplots, and 2D and 3D histograms. Views are linked by brushing and free-form selection. Using the optimal view(s), one can find structures of interest, *e.g.* spatially compact zones in DVR renderings or peaks in histograms, and highlight or erase such structures in all views at once. We enhance classical histogram views with shading, sorting, and depth, to allow spotting complex patterns with greater ease. We propose a smooth animation between multiple views, to locate (and select) data patterns which are hard to isolate in static plots. We integrate a F+C deformation that adds the ability to uncover locally occluded spatial patterns in our views. We present a GPU implementation of our techniques, which we call *color tunneling*, that creates real-time interactive, animated, explorations of datasets of tens of millions of points on a modern PC. We illustrate color tunneling with examples on 2D image editing, 3D medical visualization, and astrophysics visualization.

This paper is structured as follows. Section 2 presents related work in volume F+C exploration. Section 3 outlines our design. Section 4 details the proposed interactive exploration-and-selection techniques: view linking, warp animation, lock, brush, and dig. Section 5 presents three applications. Section 6 details the implementation. Section 7 discusses our F+C technique. Finally, Sec. 7.1 concludes the paper.

2 Related work

Occlusion is an inherent problem in 3D volume visualization. Several types of techniques alleviate this problem and help users to locate and/or select structures of interest from 3D data volumes, as follows.

Magic lenses: Magic Lenses locally modify a screen area by user-selected operators to change the appearance of shapes [3]. The idea was extended for complex effect compositing and interactive lens parameter editing [2]. Tangible magic lenses extend the base concept 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 [32] or flexible [22]. Nonlinear projection deforms 3D scenes in image space, as if seen through a cylindrical or spherical lens [40].

Semantic lenses, focus + context, and deformation: The 'dust & magnet' tool declutters scatterplots by several data-attribute-driven magnets in screen space [41]. Niels *et al.* visualize ship motions on a map by blending trajectories into smoothly shaded shapes [36]. They highlight specific trajectories by a semantic lens that works on the shading values, but does no deformation, as positions are found too important to be altered. For large datasets, deformation techniques locally change the spatial data layout to give more space to important data elements. Many variations of the original fisheye view [14] exist, *e.g.* Elastic Presentation [5], Sigma Lenses [26], and Jelly Lenses [27]. The table lens locally distorts the Cartesian layout of cells in a data table to give more space to specific table rows or columns [29]. For node-link layouts, techniques include edge deformations, *e.g.* EdgeLens [39], bring-neighbors lens [34], edge plucking [38, 37], and link sliding and 'bring & go' techniques [34, 15] and their generalizations [30]. The MoleView technique deforms data based on both spatial position and data values, allowing to 'dig' in hidden data layers [18]. Histomages links two element-based plots (images and their histograms) to allow an easier selection and editing of features in the 2D or color space [6].

Occlusion challenge: In volume datasets, occlusion is typically larger than in the above examples. If position has specific semantics, it should be carefully preserved. Several F+C techniques refine the above interaction principles for this context. A viewer-aligned radial warping of elements close to the focus is used to push data points away in 2.5D [4].

[*]e-mail:christophe.hurter@enac.fr
[†]e-mail:russ@ras.ucalgary.ca
[‡]e-mail:sheelagh@cpsc.ucalgary.ca
[§]e-a.c.telea@rug.nl

BalloonProbe uses the same radial warp idea for elements close to a focus in 3D synthetic scenes [12]. For 3D medical scans, complex manipulations (cut, peel, ply, dilate, retract) are used to expose inner structures [10], with optional animation [23]. Similar techniques are provided by Gimlenses for segmented mesh datasets [28]. Illustrative F+C techniques are further generalized in [11]. Multiple foci help to selectively enhance specific structures [17]. A detailed taxonomy of F+C techniques for volume data is given in [8].

Transfer functions: Identification and selection of features of interest in volumetric data is also supported by transfer functions. Essentially, these are mappings from subsets of the attribute domain to specific color and transparency values. Kniss *et al.* extend classical 1D and 2D transfer functions to higher dimensions in order to easier find and isolate structures of interest in volumetric data [20]. Conceptually, transfer functions propose a different way for feature isolation than F+C techniques: While the latter rely chiefly on *brushing* to select such features, transfer functions require the (careful) *design* of a data-to-appearance mapping for the same task. Although the design of this mapping can be significantly assisted by automatic data analysis [9, 31, 25], or direct WSYWIG interaction [16], their specification still involves considerable user effort.

Overall, basic F+C deformation techniques are simple to use and learn, but are hard to control in terms of what gets deformed and what does not. More complex techniques achieve better control, but also require more complex interaction tools. We next present a way to reconcile the simplicity of the former techniques with the flexibility of the latter, by combining linked views, interaction, and animation.

3 Principle

Our principle, called *color tunneling*, is a set of interactive techniques that expand the possibilities for selection and exploration of images and volume datasets, while avoiding the use of complex menus and interface components. We provide support for a rich spectrum of exploratory activities via the integration of animation and brushing.

Our input data is a uniformly sampled multivariate field $F : D \to V, D \subset \mathbb{R}^n, V \subset \mathbb{R}^m$. For simplicity, our examples consider $n = 2$ (images) and $n = 3$ (volumes). As data attributes V we consider RGB color (for images) and scalar data values, data gradients, and volumetric shading for volumes (*i.e.* the color of voxels as assigned by whichever DVR method was chosen). We use next the term *data points* to refer to voxels ($n = 3$) or pixels ($n = 2$). However, color tunneling works directly for higher dimensions and/or more attributes.

Given such a field F, we consider two types of views to show all data points in F: *Scatterplot* views map 2 or 3 dimensions or attributes in $D \cup V$ to $\mathbb{R}^2$ or $\mathbb{R}^3$ respectively, and additional attributes in V to color. For instance, a DVR of F maps D to $\mathbb{R}^3$, and scalars $v \in V$ to color. 2D *histograms* map one attribute $v \in V$ to the x axis and the number of points in D having values $v = x$ to the y axis. Fig. 1 shows such a DVR and a density histogram view for a 3D CT scan. Key to exploring F is the linking of such views by interaction and animation. A typical scenario goes as follows: The user creates a view which shows three dimensions of D, *e.g.* a DVR of F. Usually, such a view occludes interesting structures found deep inside F. Next, the user creates one or more scatterplot or histogram views by selecting combinations of dimensions and attributes of interest. Finally, the user employs interactive techniques to explore the data and isolate structures of interest: *view linking, brush, warp, dig*. These techniques are detailed next.

4 Interaction techniques

Since we are exploring dense multivariate datasets which exhibit significant overlap between groups of voxels or pixels, we need tools to interactively unveil the occluded structures of interest. For this, we propose five interactive techniques as follows:

- *View linking:* Three configurable views (2 exploration views and one lock view) for data exploration (Sec. 4.1),
- *Warp:* Animates a view between two configurations (Sec. 4.2),
- *Lock:* Locks items not to be affected by brush or dig (Sec. 4.3),
- *Brush:* With the lock view, brushing allows adding or removing data in a view (Sec. 4.4),
- *Dig:* With the lock view, digging pushes data points away from the lens center to unveil occluded structures (Sec. 4.5).

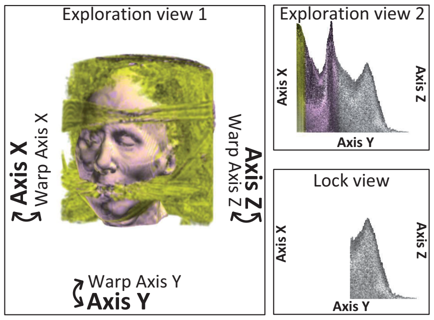

Figure 1: Two exploration views to perform data exploration and a lock view to configure the brushing and dig techniques.

4.1 View configuration and linking

We use two exploration views with possibly different configurations. Both views offer standard pan, zoom, and camera rotation. Users can interactively choose the mapping of the input data ($D \cup V$) to the view axes (Fig. 1): Double-clicking with the left mouse button on a view axis (x, y, or z) shows all data dimensions in $D \cup V$. After we select a dimension $d \in D \cup V$, we start a smooth animation between the current view configuration and the new one given by the menu choice. The animation, detailed in Sec. 4.2, helps users to preserve the 'mental map' [1] and, equally importantly, helps visually tracking patterns of interest. Both views are linked by showing the same set of data points. This enables complex data-selection operations by incremental selection or filtering, and also avoids multiple visual configuration changes. This dual-view design, originally used for trajectory analysis [19], is extended here for the more general case of DVR data exploration.

4.2 Animation between view configurations

Given any two views V_1 and V_2, we link the views by executing a linear interpolation $\mathbf{p}(t) = (1-t)\mathbf{p}_1 + t\mathbf{p}_2$, or warping, of each point $\mathbf{p}_1 \in V_1$ to its corresponding point $\mathbf{p}_2 \in V_2$. The shading $s(\mathbf{p})$ of the points is fixed: When examining a 2D image, s is the color of the image pixels; for a 3D data volume, s is the color of the voxels given by DVR. Fixed shading allows following trajectories of specific groups of data points as they move from V_1 to V_2. The key value of animation is to create a dense sequence of intermediate frames between V_1 and V_2, in which data patterns of interest become *more visible* than in *both* V_1 and V_2.

While the left mouse button configures views, the right button controls animation. Clicking this button starts the animation (from V_1 to V_2). Dragging the mouse horizontally with the button pressed controls the time t, i.e. the animation speed and direction (V_1 to V_2 or back). Releasing the button stops the animation at any moment. Next, we can use the *brush* tool to select patterns in the interpolated view $V(t)$ or the *dig* tool to reduce occlusion at desired points.

No constraints are put on the configurations V_1 and V_2 that we animate between. Both V_1 and V_2 can be 2D or 3D views, and they can map different data attributes to axes differently. When animating towards a 2D view, *e.g* going from a DVR view (V_1) towards its 2D histogram (V_2), we use V_1's z (depth) values to order pixels in the 2D view V_2 (pixels with low z values get under pixels with high z values).

Cornerstone to this work, the warp animation has proven effective in many use-cases (see also the associated video): Animating between a DVR and a scatterplot unveils brain structures (Sec. 5.1). Animating between a data cube and a 2D histogram is used to detect and select

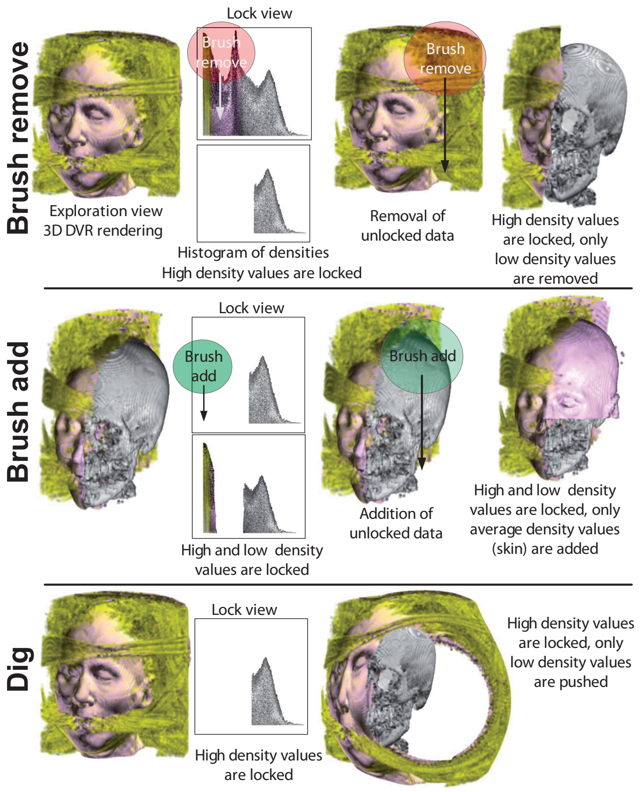
Brush remove
Lock view
Brush remove
Brush remove
Exploration view
3D DVR rendering
Histogram of densities
High density values are locked
Removal of unlocked data
High density values are locked, only low density values are removed
Brush add
Lock view
Brush add
Brush add
High and low density values are locked
Addition of unlocked data
High and low density values are locked, only average density values (skin) are added
Dig
Lock view
High density values are locked
High density values are locked, only low density values are pushed

Figure 2: Brush, dig, and lock tools. Locked items are not affected by brush and dig. We used transfer functions to make air voxels transparent and standard gradient shading. We see that the head is surrounded by a large amount of uninteresting noise (yellow).

outliers in astronomical data (Sec. 5.2). Animating between two scatterplots is used to eliminate complex regions in a 2D image (Sec. 5.3).

4.3 Lock

This view allows specifying data points not to be affected by the brush or the dig tool. Lock can be done at the start or during an animation (Sec. 4.2). We use a 'lock brush' (Fig. 1) with *add* and *remove* modes to add, respectively remove, points in a given view from the locked point-set. These modes are invoked by using the Shift, respectively Control, keys with the left mouse button, as shown by the red, respectively green brush circles in Fig 2. The lock-brush size is adjusted with the mouse wheel. Locked data points are drawn (visible), while unlocked ones are not drawn. Double-clicking in a view inverts the lock set, *i.e.* makes all unlocked points locked and conversely (see Fig. 2 for different lock view instances). Finally, the lock view axis can be configured as discussed in Sec. 4.1, and the user can perform the warp animation to choose a suitable visual configuration (Sec. 4.2).

4.4 Filtering brush

Data points can be removed if they are uninteresting and/or to reduce clutter. For this, we use a filtering brush combined with locking (Sec. 4.3). As for the lock tool, we can *remove* points in the filtering brush (with the Ctrl key), *add* back points which fall in the brush but were removed earlier (using the Shift key), and control the filtering brush size by the mouse wheel. In *remove* mode, only locked points are affected by the filtering brush. Fig. 2 shows this. Here, we first lock all points having a high density value (removal of low density values in the lock view). Using next the filtering brush in removal mode eliminates only low-density data points. This unveils the skull structure. Conversely, using the filtering brush in *add* mode, adds only unlocked data points. If we first lock both low and high-density values (using the removal brush on the average-density values in the lock view), only average-density brushed items are added. In our example, this restores the skin (Fig. 2, brush add).

4.5 Dig

To further alleviate occlusion, we propose a *dig* tool. Digging smoothly pushes data points from a focus point (mouse pointer) away in radial direction. The dig radius is controlled by the mouse wheel.

Conceptually, our dig tool is a 3D version of the MoleView principle [18]: Given a focus point $\mathbf{x} \in \mathbb{R}^2$, and a radius $r \in \mathbb{R}^+$, when the user presses the mouse button, we compute, for each point $\mathbf{p}$ whose screen projection falls in the disk $\mathscr{C}$ of radius r centered at $\mathbf{x}$, a displacement $\mathbf{p}_{disp} = r\mathbf{v}/\|\mathbf{v}\|$. Here, $\mathbf{v} = \mathbf{p} - \mathbf{x} - ((\mathbf{p} - \mathbf{x}) \cdot \mathbf{n})\mathbf{n}$ is the shortest vector from the view ray passing through $\mathbf{x}$ towards $\mathbf{p}$, and $\mathbf{n}$ is the view plane normal. While the mouse button stays pressed, we interpolate $\mathbf{p}$ towards $\mathbf{p}_{disp}$ for all $\mathbf{p} \in \mathscr{C}$, *i.e.* compute $\mathbf{p}(t) = (1-t)\mathbf{p} + t\mathbf{p}_{disp}$ for $t \in [0,1]$, using 50..100 time steps. Upon button release, we do the inverse interpolation from the displaced position $\mathbf{p}(t)$ towards the original location $\mathbf{p}$, *i.e.* let points snap back to their locations. Points close to the focus $\mathbf{x}$ will quickly drift towards the focus boundary $\partial\mathscr{C}$; points close to $\partial\mathscr{C}$ move slower towards it. This creates a clear gap around $\mathbf{x}$ and a progressively weaker deformation towards $\mathscr{C}$.

The dig tool helps exploring the dataset by unveiling structures hidden by the pushed items. Also, the dig tool can be used to forecast the effect of further brushing actions: The pushed items are the unlocked ones, and thus will be affected by subsequent brushing.

5 Applications

Below we illustrate the application of color tunneling on several 2D and 3D datasets from various application domains.

5.1 Medical imaging

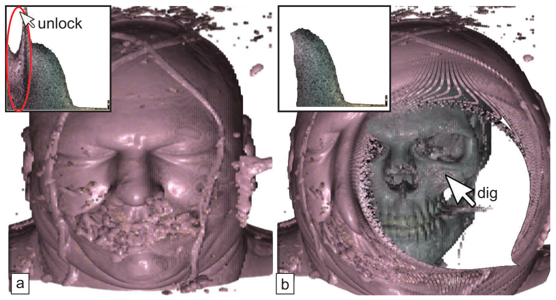

Figure 3: Opening a human head DVR to expose the skull structure.

Consider the 3D scans in Fig. 2 ($128 \times 128 \times 112$ voxels) and in the video and Fig. 3 ($128 \times 256 \times 256$ voxels). We want to "peek" inside the head to see the skull structure. For this, we first create a tissue-density histogram view and color its points by the DVR values given by gradient shading. The tall histogram peak indicates the largest voxel count in the volume, which are soft-tissue voxels. To the left of this peak, we see pink histogram points. These correspond to the skin tissue, which has the same color in the DVR view. In the middle of the peak, we see a thin dark vertical band. These are low-gradient voxels, which matches the fact that there are no density interfaces in soft tissue. In contrast, the pink histogram points are bright, which matches the DVR highlights at the skin-air and skin-soft tissue interfaces. Now that we understand the meaning of the histogram points, we select and lock all histogram points, and next select and unlock all (pink) points to the left of the peak. Finally, we apply the dig effect to the DVR view (Fig. 2 dig). This pushes unlocked voxels away from the focus, and reveals the hidden skull structure inside (gray).

Fig. 4 shows a second scenario. Here, we want to expose the top part of the brain structure in our head scan. Simple filtering cannot easily achieve this. The human head in this scan consists of a succession of layers with non-monotonic density values (low for skin, high for bone,

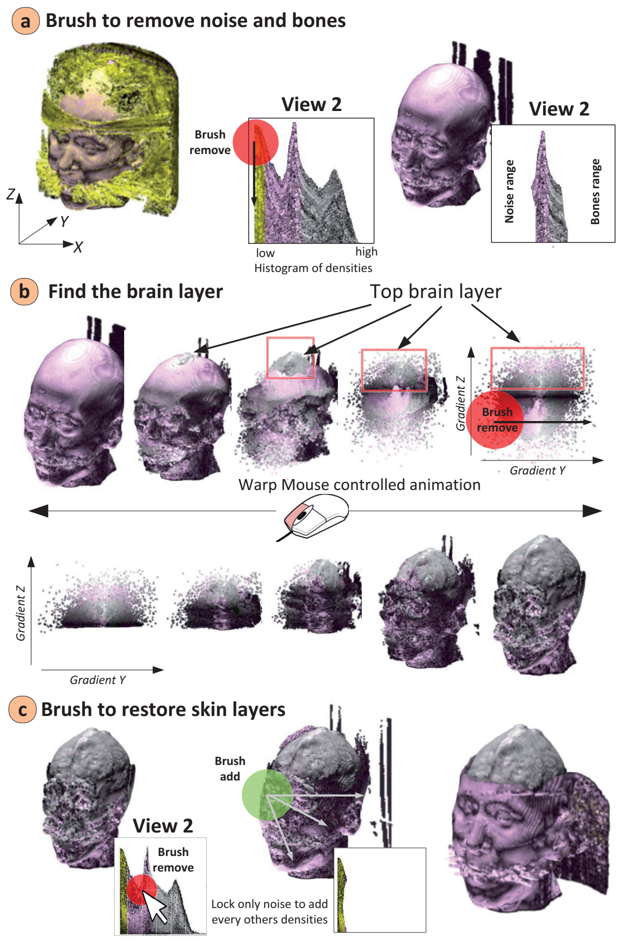

Figure 4: Exposing the top part of the brain structure in a 3D scan.

and low again for the deeply-nested brain structure). Simply filtering out the bones will not help to display the brain since the skin will not be filtered and will occlude it. Conversely, filtering on the skin density will also remove the brain which has a similar density value.

To solve our task, we use color tunneling. First, we use a density histogram view and *erase* the noise (low density) and bone (highest density) values (Fig. 4 a). Next, we create a 2D scatterplot of the z value of the density gradient *vs* the x gradient value, and use the *warp* tool to animate between the 3D DVR and this scatterplot (Fig. 4 b). Warping a few times back and forth, we see that the top-part of the brain is warped to the top-half of the 2D scatterplot. This matches the fact that, in this area, z density gradients are large. We now remove the brain lower part by erasing the scatterplot's lower half (Fig. 4 b, right image). However, this also erases some skin parts. To get these back, we use the density histogram view to *unlock* points in the skin density range (Fig. 4 c, left). Finally, we use the add brush in the DVR view to paint back the skin voxels in the damaged areas (Fig. 4 c, middle). Since only soft-density voxels are unlocked for editing, and we brush only over skin areas, only skin voxels get affected; bone or noise voxels are not painted back. Fig. 4 c (right) shows the final result.

5.2 Astrophysical data

We next consider a 3D cube of astrophysical measurements of the large-scale structure of hydrogen gas intensities in our Milky Way Galaxy ($1024 \times 1024 \times 160$ 16-bit integer voxels, 320 MB total) [33]. The x and y axes map polar sky coordinates, and z maps radiation wavelength, which translates to distance through the Galaxy along ray paths. Color maps gas intensity. Fig. 5 a shows our data cube, rendered with DVR.

In this view, astrophysicists using our tool, and who provided the feedback outlined in this section, could only see color layers that indicate regions of denser hydrogen gas from the spiral arms of the Milky Way, such as the prominent yellow slab spread over a large part of the xy subspace. These are regions of the Galaxy where the cycles of star birth and death play out. We now choose a scatterplot of intensity *vs* wavelength (Fig. 5 d). This shows two interesting phenomena. First, we see a thin compact horizontal black bar, not visible in the initial data cube. This tells that the respective intensity is present in *all* wavelengths. Secondly, we notice a white gap in the intensity-wavelength space, above the black bar at the distance of the bright spiral arm (Fig. 5 d, red marker). This tells that, for the respective wavelengths, there exist only high (purple..yelow) intensities, but no intermediate (blue..green) intensities. This situation does not occur for any other wavelengths, as there is a single such gap in the scatterplot. We now warp the scatterplot towards the original data cube: The intermediate frames (Fig. 5 b,c) show that the gap corresponds to the spatial region marked in red in Fig. 5 a, right inside the yellow wavelength band. This lack of low intensities at the location of the spiral arm shows an absence of low density hydrogen in this region. Some phenomena may have swept up the gas into high density structure, a step on the way to forming new stars.

Fig. 6 shows a second scenario. In the DVR image, we notice a few constant-intensity lines parallel with the wavelength (z) axis (Fig. 6 a). Such lines are created by radiation from objects in the far universe being absorbed by gas in our Milky Way. The properties of these lines can be used to measure the Galaxy temperature. We would like to select such lines for closer analysis. Doing this via spatial or value-range filtering is hard, since the lines are embedded in surrounding data, and also do not have a perfectly constant intensity. Also, we would like to find if similar lines exist deeper in the data cube.

We can select these lines as follows. First, we build a histogram of the intensity gradients of our data points. Gradient is a good detector for the *boundaries* of these lines, as intensity rapidly changes between the relatively constant value inside lines and varying values outside. We next sort histogram points vertically based on their intensity value, and order them in depth with high-intensity voxels first. Fig. 6 e shows the result. We see that relatively few voxels have high gradients, while the vast majority of the data is represented by a well defined distribution of gradients. Our lines of interest are located in the former voxels (histogram tail). Also, we see several color bands in the smooth part of the histogram, with a thin purple (high-intensity value) band at the top, and most points having low values (green). Such bands emerge because of our y sorting on intensity. This distribution reveals the kinematics of the Galaxy and the velocity structure of the gas, represented by gradients in intensity with wavelength.

Over the high-gradient tail, we mainly see the same green shade as on the lines in Fig. 6 a. This indicates that for these high gradients, points do not have high intensity values (purple). Our lines of interest thus occupy regions of low intensity values and high gradient.

To find our desired lines, we now warp between the histogram and DVR views. In the intermediate frames (Figs. 6 b-d), we see several horizontal lines appearing, which smoothly move from the histogram tail towards their spatial locations in the DVR view. To select *all* such lines, we thus simply select the histogram tail. For more control, we can use the transition views to select any desired line located at *specific* spatial positions. The animation unearths several additional such lines *inside* the data cube, which the DVR view (Fig. 6 a) did not show.

5.3 Image segmentation and manipulation

We next illustrate color tunneling for three use-cases for 2D images, presented in increasing order of complexity.

Dead pixel isolation: Consider a 2D color photograph, shown as a Cartesian plot (Fig. 7 left). Photos often contain isolated pixel groups whose color slightly differs from their surroundings, such as 'dead pixels' due to imperfections of digital cameras. Isolating such pixels (*e.g.* for retouching) is hard: They are visible neither in Cartesian nor in hue-saturation plots (Fig. 7 right). However, if we

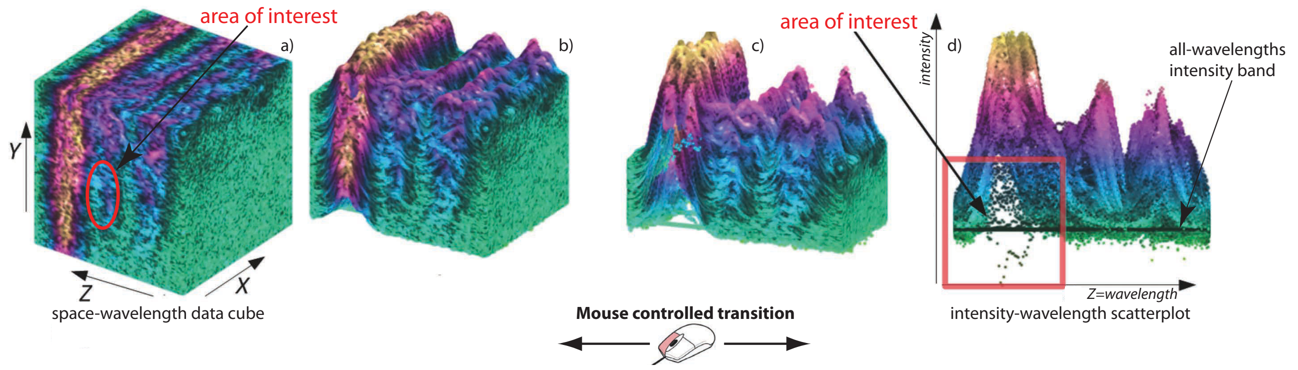
area of interest
a)
b)
c)
area of interest
d)
intensity
all-wavelengths intensity band
Y
Z
X
space-wavelength data cube
Mouse controlled transition
Z=wavelength
intensity-wavelength scatterplot

Figure 5: Finding intensity outliers with isolated ranges in an astronomical data cube.

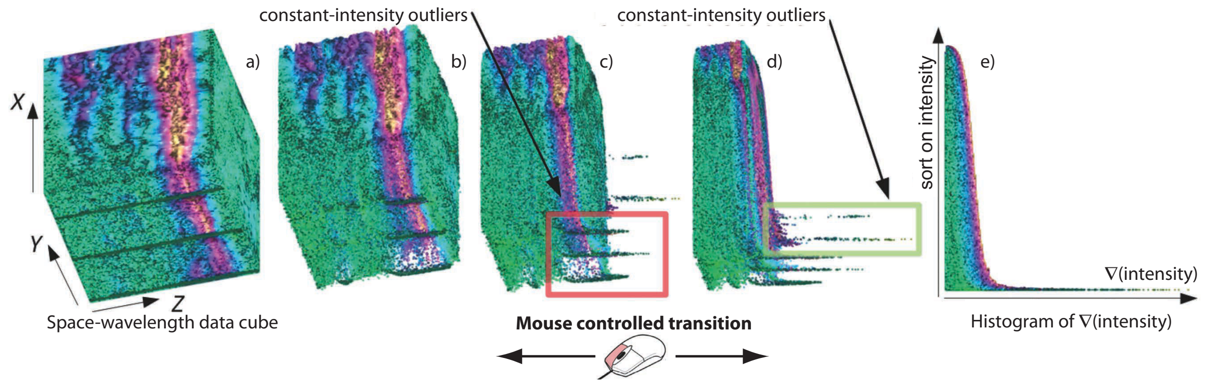
constant-intensity outliers
constant-intensity outliers
a)
b)
c)
d)
e)
X
Y
Z
sort on intensity
∇(intensity)
Space-wavelength data cube
Mouse controlled transition
Histogram of ∇(intensity)

Figure 6: Locating constant-intensity line outliers in an astronomical data cube.

warp between the two plots, such pixels clearly show up as outliers (Fig. 7 middle frame). Why does our animation highlight such outliers? The explanation is as follows. Similar-color compact spatial regions in the Cartesian plot, *e.g.* the uniform image background or the orange fish, move as *compact* blocks to their corresponding regions in the hue-saturation plot. Outlier pixels in such regions have different hue and/or saturation values, so are warped on different trajectories. In our case, these are pixels on the dark image background, whose color slightly differs from their uniform vicinity. We can stop the animation at any frame showing such pixels to select them for *e.g.* retouching.

Complex selections: Consider an input image (Fig. 8 a), where we want to select and remove the indicated wavy colorband. Selecting this area in image-space is hard, as *both* its shape and color distribution are quite complex. We use instead a mix of erase and warp, as follows. First, we use a hue-saturation scatterplot to select the band. Figs. 8 g-j show the brushing of the desired pixels in the histogram view. A dotted curve shows the brush trajectory. Directly erasing this selection also eliminates several pixels outside the desired band (Figs. 8 b-e). To correct this too large selection, we use the warp tool. Figs. 8 k-n show several frames from warping the result of our previous erase (Figs. 8 e) towards a hue-saturation plot. During the warp, our band is shifted away from its spatial position. This provides precisely the extra empty space around the undesired selections, which we can now cancel by direct brushing in the warped frame (Fig. 8 m, mouse positions). Fig. 8 n shows the warped frame after removal of the undesired selections. Fig. 8 p shows the final image, with the initial colorband precisely removed. The editing took around one minute. We also tried to isolate this color band using classical fuzzy brushing in the Graphic Converter editor [21]. To achieve the selection quality in Fig. 8 p, an expert user of this tool needed several trial-and-error passes (6 minutes). Note also that selecting these pixels using only brushing in the hue-saturation plot is *equally* hard: Fig. 8 o shows this plot after the unwanted selection correction was done using warping (Figs. 8 m,n). Comparing this image with the hue-saturation plot *before* warp editing (Fig. 8 j), we see no visible difference. Thus, the warp helped indeed to correct the selection in a way that is not possible using only the hue-saturation plot.

Image segmentation: Fig. 9 (a) shows a skin scan of a *naevus*, or mole, acquired with a Handyscope optical dermatoscope (2448 $\times$ 3264 pixels). Dermatology specialists need to segment such scans into normal skin, the mole, and the internal mole structure [7, 24], prior to applying various metrics [13] to predict the potential malignity of the mole, *i.e.*, its chance to be(come) a melanoma. A manual segmentation produced by a dermatologist, shown in image (a), takes a few minutes to complete, depending on training level and image complexity. Processing many such images, *e.g.* during routine clinical screening, is time-consuming and tedious. Automatic segmentation yields in general suboptimal results, given the high variability of tumor morphologies and color ranges. Moreover, medical users typically desire to closely control of the segmentation process. Color tunneling can help this process. Image (b) shows a hue-saturation polar plot of our scan. Here, a simple selection in the mid-saturation range captures well both the inner-structure and mole-skin boundaries. Indeed, for mole images of white-skin subjects, boundaries have a saturation located between the pale (desaturated) skin and mole (saturated) colors. The brush thickness controls the allowed boundary fuzziness – thicker brushes allow selecting fuzzier boundaries. Warping this image towards a gradient magnitude *vs* brightness plot shows how our selection splits into two clusters (images (d-f)). To map these clusters to spatial locations, the user next warps the gradient-brightness plot towards the initial Cartesian plot (images (g-i)). Playing this animation a few times, the user sees that the upper and lower clusters encode the mole-skin boundary and inner-structure boundaries respectively, which have (on average) different brightnesses. This justifies the use of a brightness plot axis. To separately select these two boundaries, the user erases the undesired cluster from the gradient-brightness plot (images (j,k) and (l,m)). Additionally, for noisy images, the user can erase high-gradient points (the

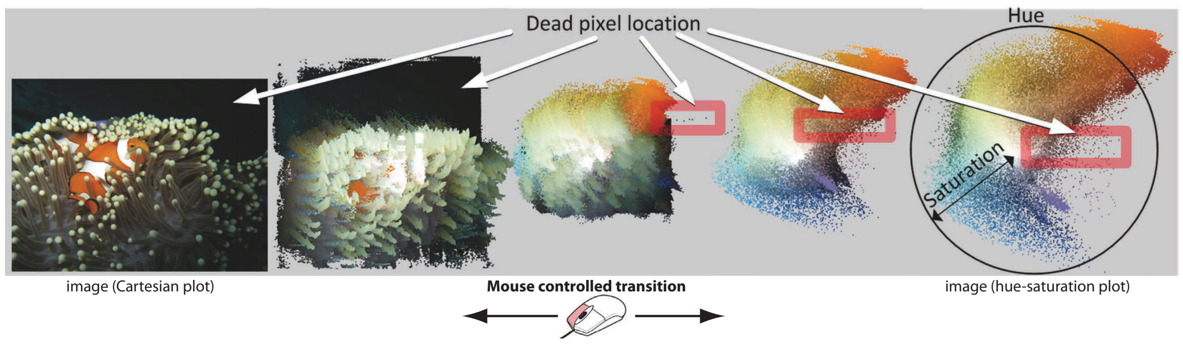

Figure 7: Dead pixel isolation. Warping between an image (left) and its hue-saturation plot (right) allows finding a few outlier pixels (marked red).

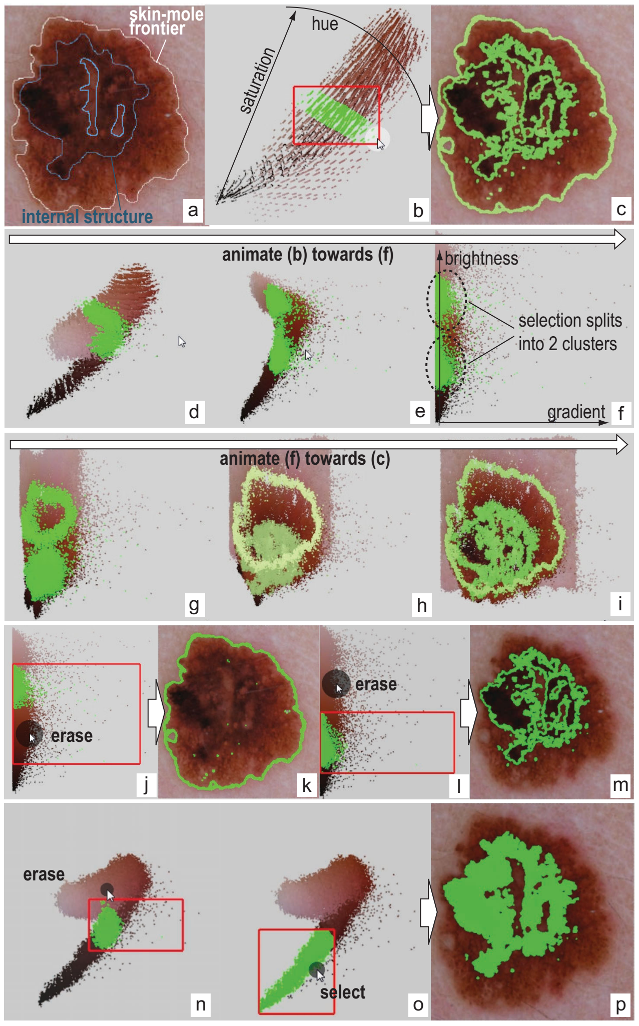

Figure 9: Skin tumor segmentation scenario.

tops of the cluster bumps) to make the selection more robust. This justifies the use of a gradient plot axis. Finally, to select the entire inner structure, the user animates a few times between the gradient-brightness and Cartesian plots, and notices that the desired structure corresponds to the 'tail' of the scatterplot in an intermediate frame (images (n,o)). To select this structure, the user first erases a few outlier pixels (image (n)), and then adds the tail shape to the selection (image (o)). Image (p) shows the successful selection of the inner structure. Note, for both the skin-mole and internal-structure boundaries, the similarity with the manual segmentations.

The above scenario was executed by an experienced dermatologist (11 years of clinical practice) after 15 minutes of pre-training using our tool, and successfully tried out on several dermatoscopic images of different types of *naevi* morphologies, with resolutions from 600^2 to 2448×3264 pixels. The user noted that, while color tunneling is not significantly faster than manual segmentation for clearly delimited *naevi*, it is faster *and* easier to use than manual segmentation on images exhibiting fuzzy complex boundaries or acquisition noise (around 2 minutes/image). In particular, the user found selecting entire regions with just a few brush strokes and animation moves to be much easier (and requiring less concentration) with color tunneling than using classical encircle-and-flood-fill operations. The user also commented that adding new scatterplot types, *e.g.* image local smoothness/contrast *vs* hue, could make color tunneling a valuable tool for selecting and analyzing specific diagnostic factors for skin tumors. Given this positive feedback, we aim to explore this direction in future work.

6 Implementation

Interactivity is key to our proposal. For datasets of 100K elements, brushing and warping can be implemented at interactive framerates in the standard OpenGL pipeline [18, 6]. Our datasets are up to two orders of magnitude larger, *e.g.* 3D volumetric scans of 512^3 voxels.

Even if modern GPUs can render 10 Mpixels/second, *interacting* with such data sizes is not trivial. We need to update in real time both selection state (for brushing) and point positions (for digging). We next detail how these operations can be efficiently done using pixel and vertex shaders. In a vertex shader, we can change the position of the drawn element, and thus achieve the digging effect. One challenging aspect for warping is writing the updated element positions. Shaders were designed to write to textures, not to arbitrary buffers. Render-to-texture is accurate but requires a complex implementation and extra writing and reading passes. Recently, OpenGL added *transform feedback*, a feature that allows vertex shaders to write to arbitrary buffers. We heavily rely on this feature, using a two-step process: First, position data is processed to account for both brushing (element selection) and warping (element displacement). Secondly, processed data is copied back into the rendering pipeline for the next brushing or warp step. This is more efficient than render-to-texture since it does not require texture I/O. Also, all data stays on the GPU, which gives an additional speed boost. Apart from the above, the implementation of color tunneling is straightforward. Only point primitives are used for rendering. Shading is freely specifiable on a per-point basis.

We implemented color tunneling using the transform feedback technique with OpenGL 4.1 and C#. For completeness, we also added classical range-based attribute selection by GUI sliders to our tool. The obtained rendering performance is one to two orders of magnitude larger than using render-to-texture [19] or direct mode rendering [18]. On a Core-i7 3.4 GHz with a GeForce GTX 580, this allows us to manipulate datasets of over 10M elements at 20 frames/second.

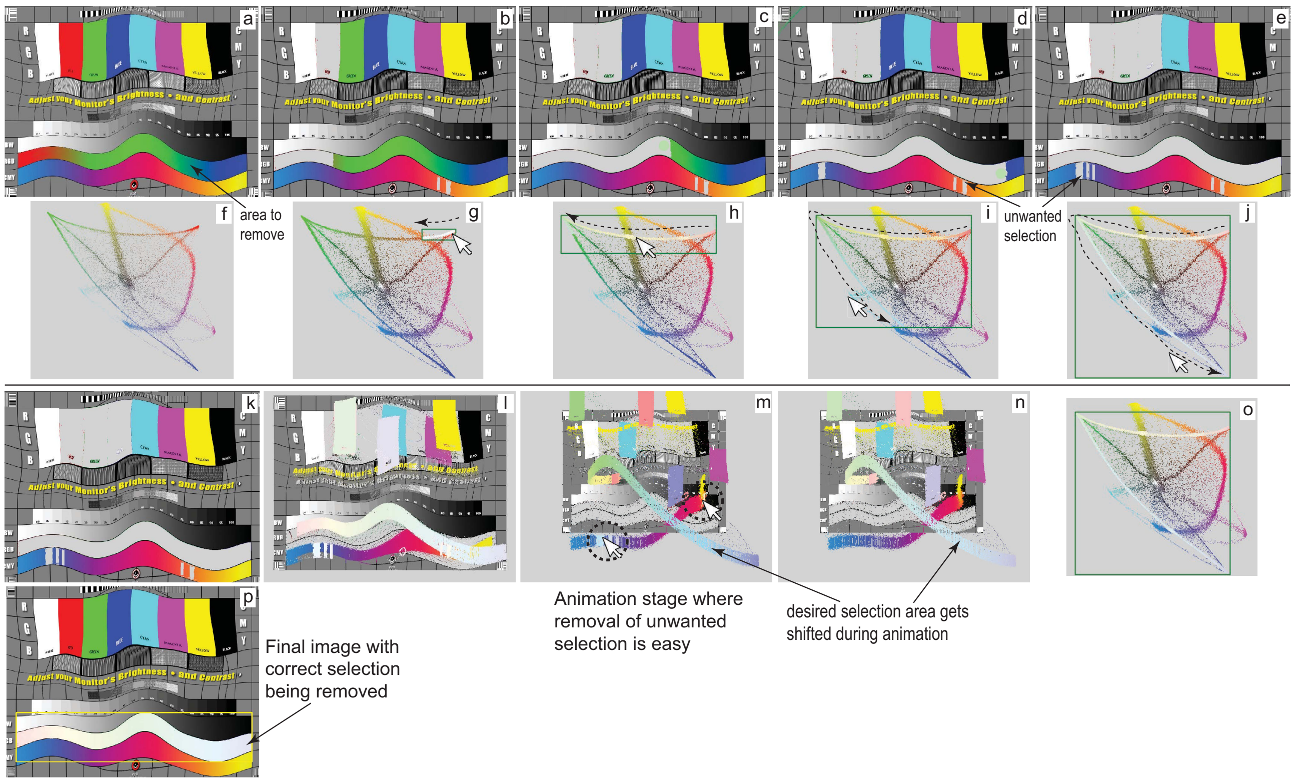
a
b
c
d
e
f
area to remove
g
h
i
unwanted selection
j
k
l
m
n
o
p
Final image with correct selection being removed
Animation stage where removal of unwanted selection is easy
desired selection area gets shifted during animation

Figure 8: Removing a complex area from a color image using a combination of brushing, animation, and linked scatterplots.

7 Discussion

We next discuss several relevant aspects of color tunneling.

Data in focus: While we borrow the data-driven F+C deformation idea from MoleView [18], a key difference exists. MoleView specifies focus points as attribute value-ranges via a range slider. Instead, we use the lock view to select our points of interest. This is more flexible, as it allows fine-grained, discontinuous, selections. In contrast to F+C deformation techniques which push away *all* points close to focus [12, 17, 39, 37], we *explicitly* specify points of interest on a fine-grained basis, using any of the lock views (Sec. 4.3). Also, in contrast to MoleView, which works purely in 2D, and to other isotropic 3D F+C techniques [12], we move points in the brush radially away from the view-ray. This creates an empty cylinder around the 2D focus point (Fig. 3), and makes our dig tool work without having to specify a focus point in 3D.

Selection space: Histomages [6] also allows selecting data in linked views. However, only pairs of *static* views (2D Cartesian plot and its histogram) are offered. We allow selection in an automatically generated *continuum* of views, created by warping between pairs of user-configured views. Animations can be stopped at any stage. Each such stage is a static intermediate view where one can explore, brush, and select data. Thus, our animation serves *also* the task of data exploration and selection, besides the goal of mental map preservation covered by earlier work. In contrast to 'semantic layers' [23], our data selection is much simpler, but equally powerful, as we use only 2D brushing instead of a complex family of 3D widgets and GUI sliders.

Scope: Several of the selection and exploration use-cases presented here are also targeted by existing techniques such as F+C deformations, range sliders, and transfer functions. However, we argue that color tunneling makes these use-cases simpler to address. As such, we see color tunneling as a *complement*, and not a replacement, of the rich set of existing multivariate data exploration techniques.

Scalability: All our views are essentially point-based plots. Our plot implementation using OpenGL's transform feedback allows achieving interactive frame rates for over 10M points. In contrast, earlier related techniques [18, 6, 17, 11, 8] propose more complex deformation and rendering implementations, which cannot achieve this scalability.

Extensions: Color tunneling can be extended in several directions. For instance, fuzzy selections can be easily added, *e.g.* by rendering points with an alpha value based on their distance to the 2D selection brush center. This would expose the selection confidence in all views. Additionally, multiple selections can be easily added.

Limitations: Point-based rendering creates sampling artifacts, see *e.g.* the small-scale skin ripples in Fig. 4. Such artifacts are small, and exist only for a few frames of the dig animation (see video). They can be removed by computing a dense sampling of the space D by backtracing the dig deformation field [17]. However, this is expensive (minutes or more), and would decrease our frame rate prohibitively. Separately, we note that better mechanisms to select the relevant exploration views are needed, especially for high-dimensional datasets, apart from the trial-and-error procedure described here. Finally, we note that formal broad user studies are needed to confirm the early usability results of color tunneling outlined by our studies presented in Secs. 5.2 and 5.3.

7.1 Conclusions

We have presented color tunneling, a set of interactive techniques for exploration and selection of structures from multidimensional datasets. Color tunneling combines simple operations: linked views, lock, dig, brush, and warp animation. Together, these operations, which are invoked by simple mouse-based brushing and clicking, without using any complex menus or other user-interface elements, support a rich spectrum of exploratory activities in volume datasets.

In contrast to previous animation techniques [18, 6], users can control and stop the animation at any stage. This yields an infinite set of in-between views where one can brush, dig, select, and explore the data. We thus use animation as an exploration tool rather than only for preserving the mental-map between two views. We illustrate this by animations of 3D cube to scatterplot (Sec. 5.1), scatterplot to scatterplot (Sec. 5.3), 3D cube to histogram (Secs. 5.1), 5.2) and 3D cube to 3D cube (Sec. 5.2). We also present a new interaction tool: the lock view. Locked items are not affected by our dig, warp and brush tools. Locking leverages brushing by allowing complex selections of brushable items, in contrast to brushing compact ranges [19, 35]. Concluding, our contributions are as follows:

- using animation as a controlled data exploration-and-selection technique,
- improved brushing with a flexible selection of brushable items,
- improved dig tool (lens deformation) with a flexible selection of pushable items,
- a simple implementation able to handle over 10M displayed data points at a frame rate of 20 images per second on a modern GPU.

Many applications and extensions of color tunneling are possible. Color tunneling is directly applicable to other high-variate datasets, *e.g.* CFD and geophysical data. The technique can be valuable for exploring dense scatterplots created by multidimensional scaling (MDS), in particular for *explaining* the meaning of point clusters in such projections. Finally, computing interpolation paths between arbitrary pairs of element-based plots so that structures and patterns are optimally highlighted is a promising future work direction.

Acknowledgements

We thank A. Diaconeasa MD PhD and D. Boda MD PhD, dermato-oncologists at the Univ. of Medicine and Pharmacy ‘Carol Davila’, Bucharest, Romania, for their joint efforts in the acquisition and analysis of dermatologic images with color tunneling. This project has been supported by the grant PN-II-RU-TE-2011-3-2049 “Image-assisted diagnosis and prognosis of cutaneous melanocitary tumors” offered by ANCS, Romania.

References

[1] D. Archambault, H. Purchase, and B. Pinaud. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE TVCG*, 17(4):539–552, 2011.

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

[3] 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.

[4] S. Carpendale, A. Fall, D. Cowperthwaite, J. Fall, and F. Fracchia. Case study: Visual access for landscape event based temporal data. In *Proc. IEEE Visualization*, pages 425–428, 1996.

[5] S. Carpendale and C. Montagnese. A framework for unifying presentation space. In *Proc. ACM UIST*, pages 61–70, 2001.

[6] F. Chevalier, P. Dragicevic, and C. Hurter. Histomages: fully synchronized views for image editing. In *Proc. ACM UIST*, pages 281–286, 2012.

[7] E. Claridge, P. Hall, M. Keefe, and J. Allen. Shape analysis for classification of malignant melanoma. *J. Biomed. Eng.*, 14(3):229–234, 2000.

[8] M. Cohen. *Focus and Context for Volume Visualization*. PhD thesis, Dept. of Computer Science, Univ. of Leeds, UK, 2007.

[9] C. Correa and K. Ma. Size-based transfer functions: A new volume exploration technique. *IEEE TVCG*, 14(6):1380–1387, 2008.

[10] C. Correa, D. Silver, and M. Chen. Feature aligned volume manipulation for illustration and visualization. *IEEE TVCG*, 12(5):1069–1076, 2006.

[11] C. Correa, D. Silver, and M. Chen. Illustrative deformation for data exploration. *IEEE TVCG*, 13(6):1320–1327, 2007.

[12] N. Elmqvist. BalloonProbe: Reducing occlusion in 3D using interactive space distortion. In *Proc. ACM VRST*, pages 134–137, 2005.

[13] R. Friedman, D. Rigel, and A. Kopf. Early detection of malignant melanoma: The role of physician examination and self-examination of the skin. *CA Cancer J Clin*, 35(3):130–151, 1985.

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

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

[16] H. Guo, N. Mao, and X. Yuan. WYSIWYG (what you see is what you get) volume visualization. *IEEE TVCG*, 17(12):2106–2114, 2011.

[17] W. H. Hsu, K. L. Ma, and C. Correa. A rendering framework for multiscale views of 3D models. *ACM TOG (SIGGRAPH Asia)*, 30(6), 2011.

[18] C. Hurter, O. Ersoy, and A. Telea. MoleView: An attribute and structure-based semantic lens for large element-based plots. *IEEE TVCG*, 17(12):2600–2609, 2011.

[19] 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.

[20] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE TVCG*, 8(3):270–285, 2002.

[21] Lemkesoft Inc. GraphicConverter editor, 2013. www.lemkesoft.de.

[22] 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.

[23] M. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *Proc. IEEE Visualization*, pages 401–408, 2003.

[24] A. Parolin, E. Herzer, and C. Jung. Semi-automated diagnosis of melanoma through the analysis of dermatological images. In *Proc. SIBGRAPI*, pages 71–78, 2010.

[25] D. Patel, M. Haidacher, J.-P. Balabanian, and M. E. Gröller. Moment curves. In *Proc. PacificVis*, pages 201–208, 2009.

[26] E. Pietriga and C. Appert. Sigma lenses: focus-context transitions combining space, time and translucence. In *Proc. ACM CHI*, pages 1343–1352, 2008.

[27] C. Pindat, E. Pietriga, O. Chapuis, and C. Puech. JellyLens: Content-aware adaptive lenses. In *Proc. ACM UIST*, pages 261–270, 2012.

[28] C. Pindat, E. Pietriga, O. Chapuis, and C. Puech. Drilling into complex 3D models with gimlenses. In *Proc. ACM VRST*, pages 135–143, 2013.

[29] 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.

[30] N. H. Riche, T. Dwyer, B. Lee, and S. Carpendale. Exploring the design space of interactive link curvature in network diagrams. In *Proc. AVI*, pages 506–513. ACM, 2012.

[31] P. Sereda, A. Vilanova, I. Serlie, and F. Gerritsen. Visualization of boundaries in volumetric data sets using lh histograms. *IEEE TVCG*, 12(2):208–218, 2006.

[32] 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.

[33] A. Taylor, S. Gibson, M. Peracaula, P. Martin, T. Landecker, C. Brunt, P. Dewdney, S. Dougherty, A. Gray, L. Higgs, C. Kerton, L. Knee, R. Kothes, C. Purton, B. Uyaniker, B. Wallace, A. Willis, and D. Durand. The Canadian galactic plane survey. *Astron J*, 125(6):3145–3164, 2003.

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

[35] M. O. Ward. Xmdvtool: integrating multiple methods for visualizing multivariate data. In *Proc. IEEE Visualization*, pages 326–333, 1994.

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

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

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

[39] 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.

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

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