Volume Visualization and Visual Queries for Large High-Dimensional Datasets

G. Reina and T. Ertl

{reina, ertl}@vis.uni-stuttgart.de
VIS Group, University of Stuttgart

Abstract

We propose a flexible approach for the visualization of large, high-dimensional datasets. The raw, high-dimensional data is mapped into an abstract 3D distance space using the FastMap algorithm, which helps, together with other linear preprocessing steps, to make changes to the resulting 3D representation within a few seconds. Thus exploration of such datasets is a less tedious task compared to other techniques. We use volumes with four components to enable the user to brush an attribute selection onto the volume for inspection. We exploit multiple transfer functions for displaying these attributes and also to filter one attribute with values of another. An advantage of this volume sampling approach is that the rendering performance is independent of the dataset size. The drawback of limited resolution can be overcome by providing a linked detail view for a freely selectable portion of space. Examples of the inspection and filtering possibilities using a silvicultural dataset illustrate the strengths of our approach.

Categories and Subject Descriptors (according to ACM CCS): I.3.3: Interactive Rendering, Large Data, High-Dimensional Data, Volume Visualization

1. Introduction

Interactive visualization of large data is a current problem with the tendency of getting worse as the availability of data increases as well as the possibilities of generating large datasets in decreasing periods of time with little to no user interaction required. Examples for this can be found in biochemistry, network traffic analysis, or simulation, just to name a few examples. Even though at least 5 dimensions are straightforwardly visualizable (position, color, size being the most intuitive mapping), the data that has to be visualized tends to have a dimensionality much higher than that. We propose to use volume visualization for scattered data, an approach borrowed from scientific visualization and capable of interactively visualizing millions of data items without performance degradation. Our approach also tries to alleviate the *curse of dimensionality* [Bel61] with an interactive tool which allows for mapping a selectable subset of the available dimensions into 3D. We make use of modern graphics hardware to render the dimensions beyond the three positional ones upon user interaction.

We choose a 3D rendering method for the data because it provides us with more space to layout the data as well as benefiting from an additional axis when mapping high-dimensional data. We also limited ourselves, on the other hand, to mapping three dimensions to allow for user-selectable dimensions to be brushed onto the remaining display dimensions as will be described in section 4. A major problem that arises when using 3D rendering techniques is occlusion of potentially important data. The volume rendering approach, however, enables us to alleviate this problem by using a semi-transparent representation of the scattered data depending on the density of points in a given portion of space. Another factor to consider is performance degradation when choosing a 3D representation over 2D, but volume rendering comes with the benefit of being reasonably interactive on current hardware when choosing moderate volume resolutions (up to 256^3) on the one hand, and being, on the other hand, performance-wise invariant to the number of data items that have to be visualized once the volume we want to render is generated. A handicap of volume rendering as a concept is the quantization and binning of the data that has to be displayed, since basically only discrete voxels are available in the volume we are rendering. To overcome this limitation, we make a 3D volume brush and a linked detail view available, which can display the actual data points contained inside the 3D brush, thereby providing the user with a volume-rendered context of focused points visible in a linked 3D scatterplot. The main strength of this approach is the possibility of interactively visualizing massive datasets without performance degradation. The user has control over interactive highlighting and intuitive on-the-fly range-based

filtering that does not require to know the exact extrema of the data, since it works relatively to the range of available values.

2. Previous Work

The visualization of high-dimensional data has seen different classical approaches, like parallel coordinates [Ins85], star glyphs [SFGF72] and icon techniques, like stick figures [PG88] or chernoff faces [Che73], which however are strongly limited in respect to large datasets, or scatterplot matrices [BC87]. More recent approaches include circle segments [AKK96], and pixel-based visualization of multi-dimensional data as in recursive patterns [KAK95]. However, the limits of screen space always force us to either depict large datasets or many dimensions unless resorting to user interaction for shifting the focus.

A lot of research has been conducted in the area of volume rendering for scientific visualization, using 2D textures on PC hardware [BJNN97] or 3D Textures on high-end workstations [CCF94]. Recent developments improve the visual quality while not increasing the actual volume resolution, but pre-integrate inter-slice data [EKE01]. Other approaches make the definition of transfer functions more intuitive [KKH01] while taking into consideration more information (gradients and derivatives), or even enable different transfer functions and rendering techniques for defined regions of a single volume [HBH03].

First steps have been made in the direction of combining scientific visualization with information visualization, for example to visualize large, high dimensional simulation data [DGH03], [KSH03] with focus+context techniques in linked views. This approach was inspiration for our work, but we want to give the user access to the dataset as a whole for getting an overall impression and then applying filters, without the need to decide which features he wants to see beforehand. Volume visualization of relational data was already proposed [Bec97], but limited to displaying a single attribute other than the density and relatively simple filtering. Another work [HM03] details a volume rendering method for flow simulation which takes into account several dimensions in addition to the positional ones. A first approach for large chemical data visualized as volumes has also been developed [OIEE01].

3. Volume Generation

Our method for generating the volume that is going to be visualized makes use of the FastMap algorithm [FL95]. The process can be considered as a pipeline with several possibilities of feedback through user control, as shown in figure 1. A subset of the dimensions contained in the source data is internally mapped from the discovered range onto $[0,1]$ each for equalization. String attributes are treated as identifiers and have their distance modeled as inequality, i.e. a distance of 0 for equal strings and 1 otherwise. Then a pair of *pivot points* is heuristically selected based on the biggest high-

D distance found by randomly seeding a point and choosing the farthest point several times iteratively. These pivots form an axis onto which the remaining points are projected for finding the coordinate of the first dimension. The remaining two dimensions are calculated accordingly with a modified distance measurement taking into account the error produced on the previous axis. The result of these calculations are synthetic 3D coordinates that show the similarity of different data points as proximity. This method is not as accurate as a classical MDS [BG97], for example, but has the advantage of linear complexity. FastMap requires $O(N)$ compared to an improved hybrid MDS ($O(N\sqrt{N})$), as proposed in [MRC02]. When working with a data set of 2.3 million points and refining the pivot search 5 times we can cut processing time down to about $1/80$. So the user can, with acceptable processing times, tweak the dimension subset and the distance calculation method (see below) for optimal results. The computed 3D data, which now represents the
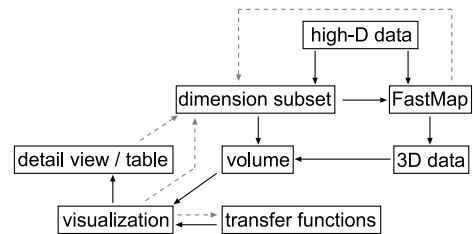

Figure 1: *Visualization pipeline showing the data flow as arrows, and user adjustments, like filtering, as dashed arrows*

whole dataset subject only to the mutual high-dimensional difference between data points is quantized (binned) into a 256^3 volume. We can encode up to 4 attributes for each voxel using a single 3D volume for data storage in the graphics card. We reserve the first component (R) for the density (i.e. number of points per voxel), since this information is most crucial and allows us to assess the distribution of the data points in distance space (the density can be scaled logarithmically or linearly). For the remaining three volume dimensions (G, B, A) the user can choose any dimension of particular interest and map different statistical per-voxel measures into them, like mean, variance, minimum or maximum. This decision is crucial for the ensuing interpretation of the visualization, as for the most part one voxel has to represent several distinct data items which cannot be visually distinguished in the volume rendered representation. Since FastMap is used for the spatial distribution of the points in 3D, one could assume that points inside one voxel are similar enough to have their attributes averaged, but this will not apply for every single attribute we can choose. Especially categorical values pose a problem, since calculating a mean from different categories conveys wrong information. The mean can only be applied to categories in combination with a variance lens as to discover the areas with uniform data. What usually would be searched for when in-

vestigating categories themselves is the category with the highest number of representants in a voxel, but this value is either very memory-intensive or slow to calculate. The time-intensive variant would be iterating the volume and processing all points for each voxel to find the category with most representants (with a cost of $|V| * N$, $|V|$ being the number of voxels). The memory-intensive variant would be processing all points once and counting occurences per voxel and category (requiring $|C| * |V|$ of space, $|C|$ being the number of categories).

The user must realize that he can only retrieve the exact data from the detail view in any case and choose the calculated value per voxel on the basis of what kind of features he wants to detect in the dataset.
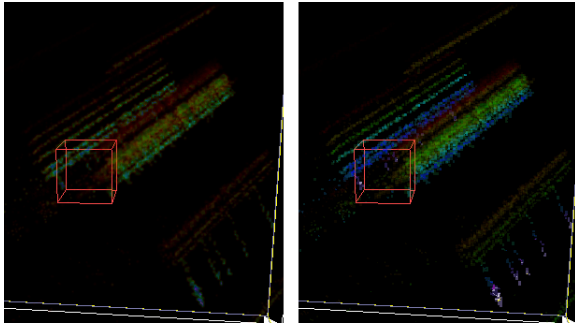
Figure 2: *Comparison of linear interpolation (left) and nearest neighbor (right) when displaying categorical values. On the left one can clearly see the artifacts of a different color introduced in areas of otherwise uniform color.*
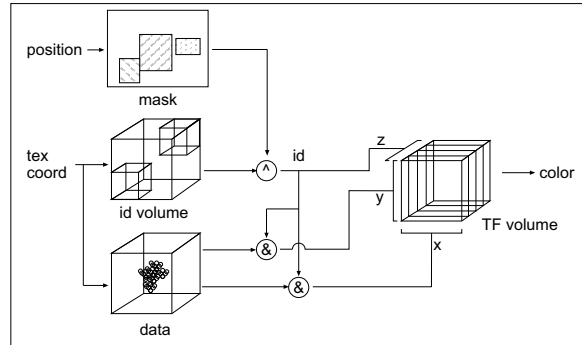
4. Rendering of the Attributed Volume

We want to be able let the user choose which attribute (point density or one of the 3 freely definable dimensions) is going to be visualized for determined areas. There are two different tools for the user to define these areas:

- Volume primitives like boxes or spheres for a rough segmentation of the dataset. These primitives are software-rasterized into a 3D-volume as IDs, or better component indices of the RGBA data volume, as to define object-space regions for each. This manipulation is afflicted with some delay by the clearing, rasterization and uploading of a 256^3 texture. However this happens only after releasing the mouse button; as long as the button is held, a 64^3 volume is used.
- The user can also utilize image-space lenses which can be freely moved in real time as a particular kind of brush to override object-space segmentation for a determined area and display the associated attribute. These lenses are rendered into a 2D mask texture which is used to hold the component index for the different regions.

Associated with each visualizable attribute is a separate transfer function, so the user can map the value range of the density and the extra attributes to different color and alpha

gradients.

When coming to the 3D texture lookup for the final ren-

Figure 3: *Fragment program diagram. The attribute IDs are combined (ˆ), prioritizing the image space mask, to select (&) one of the four volume components for each slab end. The ID also serves as Z coordinate for the transfer function lookup.*

dering, we must consider the problem categorical attributes pose, as mentioned in section 3. If sampling with nearest neighbor, we can clearly see the artifacts produced by the intersection of the volume with the view-aligned slices when zooming in. Still, we can only safely interpolate when inspecting attributes with a continuous range (see figure 2). The pixel shader we used is capable of pre-integrated volume rendering (algorithm as in [EKE01]). Since pre-integration substitutes a linear interpolation between pairs of the rendered slices, it causes the same problems as linearly interpolated sampling from the texture, at least for categorical values. If paired with interpolated lookup, it can, however, improve the visual quality when rendering the density of data items. Because of this, the user can toggle interpolation and pre-integration on and off.

In the shader (also shown schematically in figure 3), we first look up the attribute from the image-space mask and the attribute from the tag volume, discarding the latter if an attribute lens is in place. This ID is used as the z-value for a 3D transfer function lookup, the different pre-integrated 2D lookup textures being stacked along the z/r-axis of a 3D volume. We then get the values for both ends of the current slab from the volume. These values complete the transfer function lookup by providing x and y coordinates. If applied to
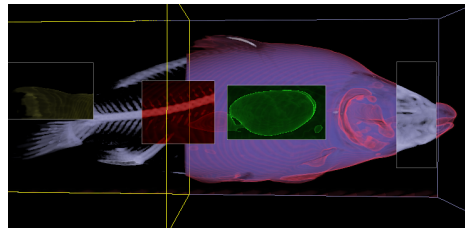
Figure 4: *Effect of 5 different transfer functions. The carp is segmented into two halves using a segmentation primitive, 4 lenses apply 4 different transfer functions in image space.*

scientific visualization, the approach yields a result as in figure 4. The tag volume segments the carp in two halves for the first and second transfer functions, the second transfer function is also associated with the lens near the carp's mouth. For this example we used a luminance volume instead of a RGBA volume since the dataset only has a density component.

5. Features

We want to enable the user to assess the global distribution of the data items with respect to their similarity and highlight determined attributes and/or filter them, so he can drill down into the data in the regions he deems most interesting and access the exact data located there. To facilitate this, our prototype provides a GUI consisting of 5 components: the context view consisting of the whole dataset rendered as a volume, a table where the user can choose the dimensions to map, a toolbar for switching interaction modes and a linked detail view plus a second table displaying data points extracted from the context view. The user can move a brush box around in the context view to select the points that will be rendered in the linked detail view and shown alongside their attributes in the second table. The user can click inside the point cloud to select points, which are then highlighted in the table as well and vice versa (see figure 5).

The context view can be used for visual data mining because we integrated the additional dimensions to allow the user to inspect them by brushing in object space (with volumes) or image space (with a special kind of lens). The table showing the different attributes (and their ranges) present in a dataset allows the user to distribute the 3 available lenses to three of these attributes. Additionally, for each of these lenses there is a corresponding lens volume which can be positioned as to segment the data in object space – as long as there is no overlapping lens, since it has a higher priority. This allows for detection of regions of interest for different attributes at once by moving the lenses to regions which the user wants to inspect for (ir)regularities or extreme range values. Explorative mining can also be performed making use of the detail view to probe into the data and inspecting the neighborhood which will yield similar data because of FastMap. Since we also associated a transfer function with every lens, the source data range can be mapped to different gradients per lens/attribute, but we also gain the capability of filtering the visualization directly by using the transfer function. The user can use the alpha output of the transfer function to filter parts of the range of a single attribute without needing to know the exact limits and he also has real-time feedback on the effect of such an exclusion.

6. Results

The volume visualization of a dataset can be used to visually detect clusters of similar data and outliers which can be considered for further analysis. In Figure 6 we can see the `cov-`
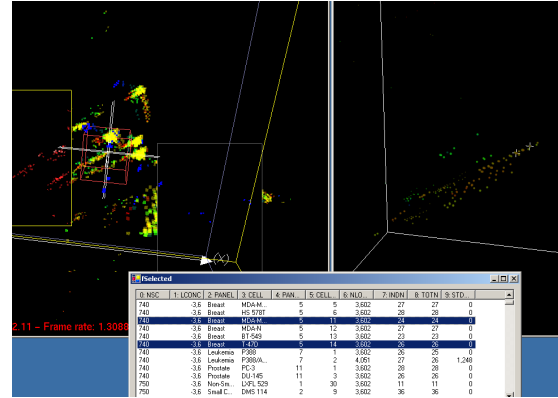
Figure 5: *Picking linked between windows: The selection can be made in the detail view or in the corresponding table, the highlight is spanned across the windows.*

`type` 54D dataset from the UCI Machine Learning Repository with several visually separable clusters. The dataset consists of 581,012 entries of 7 tree types located in four wilderness areas in the Roosevelt National Forest of northern Colorado. Each entry has several attributes, like a soil type classification, wilderness location, ground elevation, slope, shade, distances to hydrology etc. The four different wildernesses which were investigated form several clusters each, the shapes of which can be seen in figure 7 on the left, where the effect of using 4 different lenses for filtering the binary wilderness flags is collaged. If we use fractional distances (instead of euclidean) for calculating the FastMap, we can benefit from the better measurement quality [AHK01] and get more homogenous clusters for the wildernesses (figure 7 right). This also shows that quality of our approach depends significantly on the quality of the dimensionality reduction. From the accompanying description we have the information that the wilderness with the highest mean elevation is Neota. To define a range-based filter, we set the transparency output of the transfer function accordingly. Since every lens comes with its own transfer function, we can just filter out the lower 85 percent of the data points inside the lens by setting alpha to 0 (see figure 6 right). Even more interesting data mining
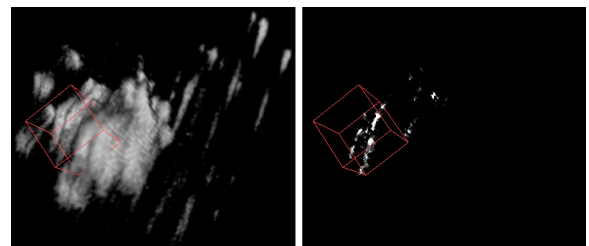
Figure 6: *Before (left) and after (right) filtering the trees having an elevation lower than 85% of the elevation range.*

possibilities arise with a slightly modified fragment program which allows for modulation of a transfer function assigned
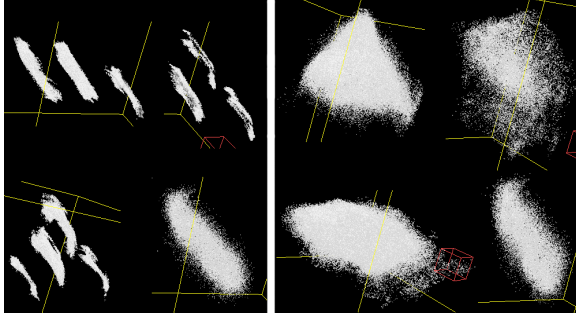
Figure 7: *Collage of 4 wilderness clusters, FastMap with euclidean distances to the left (higher fragmentation), fractional distances ($f = 0.3$) to the right (better defined clusters)*

by using a segmentation volume brush with a filtering lens. If we use the first volume and set the transfer function in such a way that each of the different trees gets a distinct color (then averaged per voxel), we get a result as in figure 8. We can then assign a transfer function filtering out low elevations to visually discover the most common trees for high elevations (see figure 9. Another example would be to only display the tree population for a certain wilderness, in this case of Rawah (see figure 11). We must keep in mind, however, that these filtering tools work only on a voxel basis, so the results depend on whether we sampled the mean or minimum etc. of all data points in a determined voxel. To get the exact data, we still have to use the detail view (figure 10) to see exactly which kinds of trees live at the higher elevations, in this case it would be exclusively Krummholz. However this kind of data mining process only makes sense as long as the user can apply filtering and highlighting to the volume view since otherwise the user would have to inspect every possible subvolume of the data to find regions of interest.
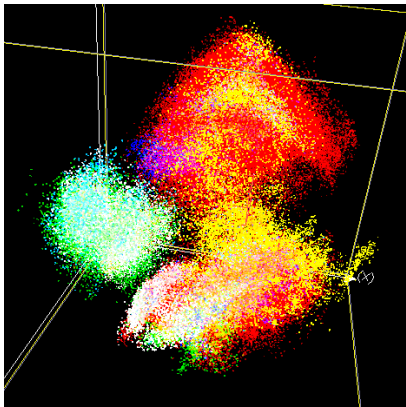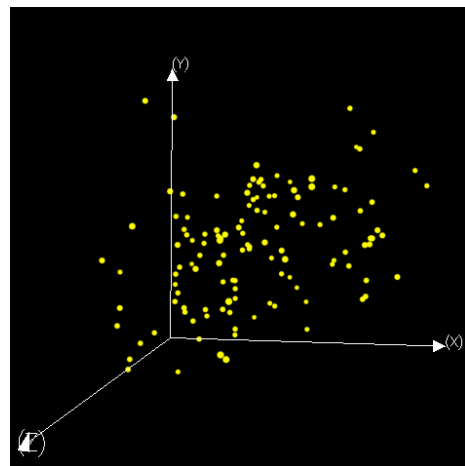

Figure 8: *Trees colored by type with a transfer function divided in 7 parts (see color plate).*

7. Performance

As with all volume rendering approaches, the performance of our rendering approach is limited by the fill rate, i.e. mem-
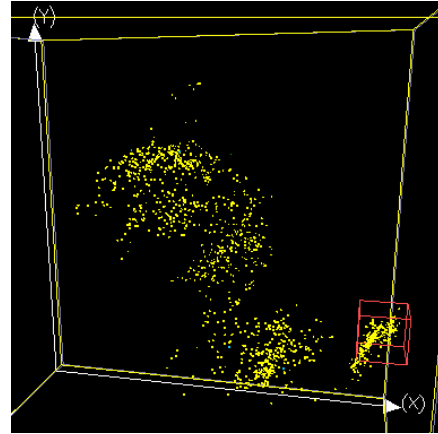

Figure 9: *Trees colored by type, filtered by a second transfer function making all trees beneath 85% elevation transparent*

Figure 10: *A detail view of the actual trees in regions of high elevation, coloring taken over from the transfer function in figure 9.*

ory bandwidth of the graphics card used. Furthermore, we need to use two rendering passes because of the 5 texture lookups we need (slab front, slab rear, lens mask, id volume and transfer function, the last one dependent on all previous ones and a number of calculations). Depending on the viewed size, on a Radeon 9700 Pro, this yields an average of 7.5fps in a 600^2 window (averaged over all viewing angles, since 3D texture organization in ATI hardware causes slowdowns when inspecting the volume from the 'rear'). The execution of FastMap on 580K points in 54D takes about 132 seconds on an intel P4-2.8Ghz Machine, while 2.3M points in 9D take 248 seconds. The updating of the additional dimensions including texture upload takes about 3 and 5 seconds (without particularly optimized algorithms), so it is safe to say that the user can experiment interactively with the prototypical implementation. The performance of the scatterplot is not critical, since only a small portion of the available data has to be rendered at any one time.
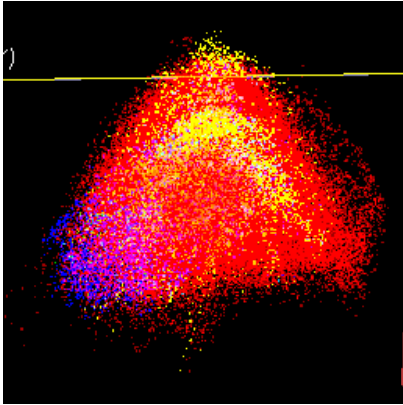
Figure 11: *Showing the trees of wilderness Rawah. Hue is produced by one transfer function on the tree type attribute and modulated by the transfer function filtering out the wilderness.*

8. Conclusions and Future Work

We have presented a novel technique for the visualization of large, high-dimensional datasets by employing dimensionality reduction and volume rendering. Additional dimensions can be brushed onto the volume by user interaction for highlighting or filtering. Our approach works well for large datasets, which we proved by showing the performance and results using a dataset with 580K entries. Future Work could include the integration of more advanced Detail Views, i.e. high-dimensional visualization approaches like those mentioned in section 2, instead of the simpler 3D scatterplot. A user study could be conducted to investigate the acceptance of our approach and further improve our solution. Finally, advanced techniques from scientific volume visualization could be adapted for handling certain filtering aspects, for example the clipping method presented in [WEE02] for volume filtering in 3D similarity space.

Acknowledgements

References

[AHK01] AGGARWAL C. C., HINNEBURG A., KEIM D. A.: On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science 1973* (2001), 420. 4

[AKK96] ANKERST M., KEIM D. A., KRIEGEL H.-P.: Circle segments: A technique for visually exploring large multidimensional data sets. In *Proceedings Visualization '96* (1996), IEEE Computer Society. 2

[BC87] BECKER R., CLEVELAND W. S.: Brushing scatterplots. *Technometrics 29*, 2 (1987), 127–142. 2

[Bec97] BECKER B. G.: Volume rendering for relational data. In *Proceedings Information Visualization '97* (1997), pp. 87–90. 2

[Bel61] BELLMAN R. E.: *Adaptive Control Processes*. Princeton Univ. Press, 1961. 1

[BG97] BORG I., GROENEN P.: *Modern Multidimensional Scaling*. Springer Verlag, New York, 1997. 2

[BJNN97] BRADY M. L., JUNG K., NGUYEN H., NGUYEN T.: Two-phase perspective ray casting for interactive volume navigation. In *Proceedings Visualization '97* (1997), IEEE Computer Society, pp. 183–190. 2

[CCF94] CABRAL B., CAM N., FORAN J.: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the ACM Symposium on Volume Visualization* (1994). 2

[Che73] CHERNOFF H.: The use of faces to represent points in k-dimensional space graphically. *Journal of American Statistical Association*, 68 (1973), 361–368. 2

[DGH03] DOLEISCH H., GASSER M., HAUSER H.: Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of VisSym '03* (2003), pp. 239–248. 2

[EKE01] ENGEL K., KRAUS M., ERTL T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Eurographics / SIGGRAPH Workshop on Graphics Hardware '01* (2001), Annual Conference Series, Addison-Wesley Publishing Company, Inc., pp. 9–16. 2, 3

[FL95] FALOUTSOS C., LIN K.-I.: FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data* (1995), pp. 163–174. 2

[HBH03] HADWIGER M., BERGER C., HAUSER H.: High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proceedings of Visualization 2003* (2003), pp. 301–308. 2

[HM03] HAUSER H., MLEJNEK M.: Interactive volume visualization of complex flow semantics. In *Proceedings Workshop Vision, Modelling, and Visualization 2003 (VMV'03)* (2003). 2

[Ins85] INSELBERG A.: The plane with parallel coordinates. *The Visual Computer*, 1 (1985), 69–91. 2

[KAK95] KEIM D. A., ANKERST M., KRIEGEL H.-P.: Recursive pattern: A technique for visualizing very large amounts of data. In *Proceedings VIS '95* (1995), IEEE Computer Society, p. 279. 2

[KKH01] KNISS J., KINDLMANN G., HANSEN C.: Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of Visualization 2001* (2001), pp. 255–262. 2

[KSH03] KOSARA R., SAHLING G. N., HAUSER H.: Interactive poster: Linking scientific and information visualization with interactive 3D scatterplots. In *Poster Compendium of VIS '03* (2003), pp. 96–97. 2

[MRC02] MORRISON A., ROSS G., CHALMERS M.: A hybrid layout algorithm for sub-quadratic multidimensional scaling. In *Proceedings of Information Visualization '02* (2002), pp. 152–158. 2

[OIEE01] OELLIEN F., IHLENFELDT W. D., ENGEL K., ERTL T.: Multi-variate interactive visualization of data from digital laboratory notebooks. In *ECDL: Workshop Generalized Documents* (2001). 2

[PG88] PICKETT R., GRINSTEIN G.: Iconographics display for visualizing multidimensional data. In *IEEE SMC '98* (1988), pp. 514–519. 2

[SFGF72] SIEGEL J., FARRELL E., GOLDWYN R., FRIEDMAN H.: The surgical implication of physiologic patterns in myocardial infarction shock. *Surgery 72* (1972), 126–141. 2

[WEE02] WEISKOPF D., ENGEL K., ERTL T.: Volume clipping via per-fragment operations in texture-based volume visualization. In *Proceedings of IEEE Visualization '02* (2002), pp. 93–100. 6
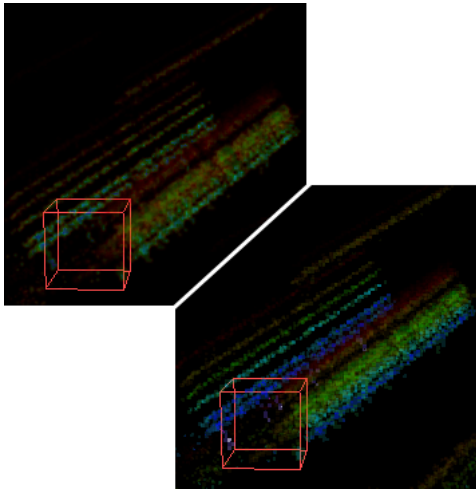
Figure 12: *Comparison of linear interpolation (left) and nearest neighbor (right) when displaying categorical values. On the left one can clearly see the artifacts of a different color introduced in areas of otherwise uniform color.*
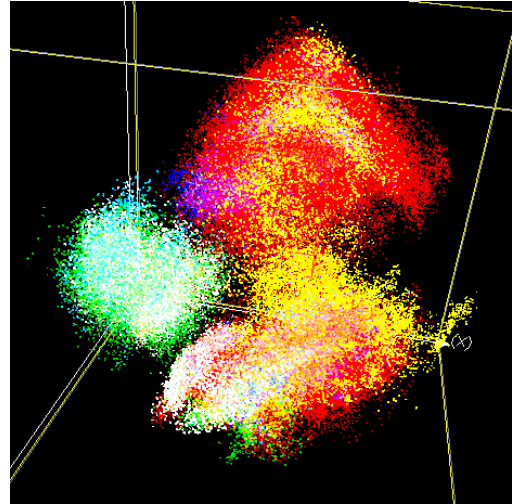
Figure 13: *Trees colored by type with a transfer function divided in 7 parts.*