Linking Scientific and Information Visualization with Interactive 3D Scatterplots

Robert Kosara Gerald N. Sahling Helwig Hauser

VRVis Research Center
Vienna, Austria
http://www.VRVis.at/vis/
Kosara@VRVis.at, niki.sahling@paradigma.net, Hauser@VRVis.at

ABSTRACT

3D scatterplots are an extension of the ubiquitous 2D scatterplots that is conceptually simple, but so far proved hard to use in practice. But by combining them with a state-of-the-art volume rendering engine, multiple views, and interaction between these views, 3D scatterplots become usable and, in fact, useful. 3D scatterplots can not only show abstract data dimensions, but also the physical layout of points, and thus provide a link between feature space and the actual object. Brushing reveals connections between parts and features that otherwise are hard to find. This link also works not only from feature space to the spatial display, but also vice versa, which gives the user more freedom in exploring the data.

Keywords
Information Visualization, Scientific Visualization, Scatterplots

1 Introduction

Scatterplots are a very ubiquitous method for visualization, and are used in many applications. A scatterplot consists of one point on a plane for each data point. The position of the point depends on the two dimensions that are displayed in the plot, which define the two axes for the plane. A scatterplot can not only show abstract data dimensions very effectively, but also provide a crude image of an object if fed with the right data (i.e., point coordinates).

Scientific visualization (SciVis) is concerned with the display of data that relates to real-world objects, e.g., medical data sets, or flows of gases. Information visualization (InfoVis), on the other hand, deals with data that is abstract and much harder to depict. Typical InfoVis data are bank transactions, telecom connection data, etc.

However, there are many applications where data is used that cannot be classified so easily, e.g., flow data with many data dimensions. In such cases, it is beneficial to combine methods from both fields, so that the data can be handled more easily.

In this paper, 3D scatterplots are presented as a new way to link scientific and information visualization. This is done by using a software volume renderer for display, and combining it with InfoVis interaction methods such as linking an brushing. The challenges in working with 3D data in this context are also discussed.

2 Related Work

2D scatterplots are a very old and well-known visualization method for unstructured data. They depict each data point in a data set as a single point (or small object) on a plane whose coordinates correspond to two of the data dimensions. If the data set has more dimensions, only two of them are used, resulting in a projection of the data points onto the plane. Additional dimensions can be shown by drawing glyphs, for example, or using color or size as additional visual attributes.

An extension of scatterplots are prosection views [6], which allow the user to select a range of values on one or more axes that is not shown. Only the points that lie in this range are projected. Another extension is the scatterplot matrix [3], which consists of many scatterplots that show all combinations of two dimensions from the data space. They are put into a matrix so that all plots with the same Y axis are in one row, and all plots with the same X axis are in the same column.

Two-dimensional projections (like they are done in scatterplots of high-dimensional data and prosection views) can also be extended to three-dimensional ones, e.g., in nVision's *worlds within worlds* approach [5].

3D Scatterplots have already been proposed, even using volume rendering [2]. But the resolution of the data

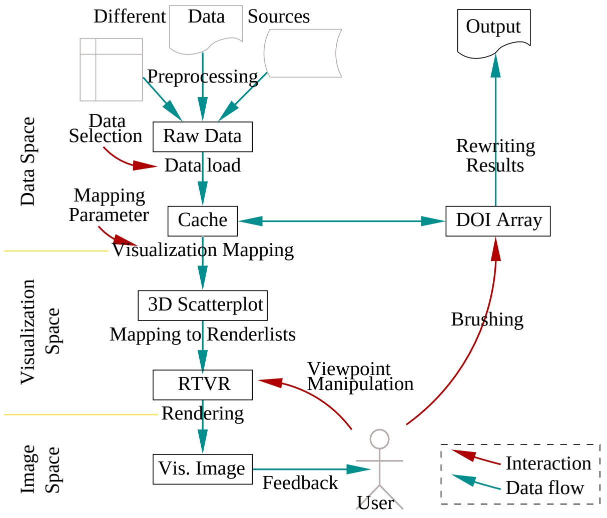


Figure 1: The Voxelplot mapping process. Data is read into the system, mapped into visualization space, and rendered by RTVR. The user can interact with different parts of the system.

there (20x50x50) is very coarse, and because the data bins are displayed in a very fuzzy way, structures in the data are very hard to see – the impression of a "data cloud" is created that may look interesting but is not necessarily a good depiction of the data. And because there is only one view on the data, interaction and the combination of different information is also quite limited.

Using multiple, linked views is one of the key ideas to combining scientific and information visualization views. One very good example for this is WEAVE [7], which allows the user to see different views like scatterplots, histograms, and a 3D rendering of an object and to brush in the 2D displays. Other work has done similar things in flow visualization [4], where the user can find out which parts of an object are described by which feature. Also in this case, linking back from the spatial display to feature space is not possible. InfoVis has also been used successfully for supporting the user with transfer function specification [10].

Voxelplot uses RTVR [11], which is a very fast Java library for interactive direct volume rendering. It organizes its data in such a way that as few points as possible have to be drawn (which is similar to a splatting approach). The image is composed using the shear-warp algorithm. RTVR can define objects in the data, whose features can be different, and which can be drawn in different ways. The latter is called two-level volume rendering (2lVR [8]), and is especially useful for focus+context visualization in 3D data sets.

3 Voxelplot

Voxelplot is an implementation of 3D scatterplots based on RTVR. Each data point is mapped to one voxel in three-dimensional visualization space depending on its value on the selected axes (hence the name *voxelplot*).

Working with data in Voxelplot consists of several steps (Figure 1) that are outlined below and described in more detail in the following sections.

1. **Data load.** The user selects the data source and the data dimensions to be loaded (to preserve memory, dimensions that are not needed are not loaded). The data is then transferred into a memory cache.

2. **Visualization Mapping.** From the source data space, only the dimensions to be displayed for each view are selected and mapped into visualization space (i.e., visual feature space). The user can influence the way in which this is done by choosing the dimensions and selecting the mapping as well as adjusting its parameters.

3. **Rendering.** The result of the mapping process is put into renderlists, which is the internal data format of RTVR. RTVR then renders the image and presents it to the user.

4. **Interaction.** The user can interact with RTVR directly (changing the viewpoint, zooming, etc.) or with Voxelplot (brushing, changing the mapping, etc.). The interactions of the different subsystems work together seamlessly.

3.1 Data Load

As data source, we mostly use database tables. Each row in the table is a data point, each column is a data dimension. In addition to the data dimensions, results from clustering algorithms, degree of interest values, etc., are also treated as first-order data and can be used as axes.

The data is put into a cache that can be swapped onto disk for dealing with large data using a mechanism provided by RTVR.

3.2 Visualization Mapping

After data load, the original data is mapped into visualization space (this is done separately for each view). The dimensions to be displayed have to be selected and assigned to visual features: one of three position axes, color, or opacity.

The mapping step is important because the data also has to fit into the feature space that is usually much smaller than the original one. This leads to the problem that important parts of the data might not get enough space to be usable. Therefore, different mapping functions can be used to spread the data over the available space.

For each dimension, the user has to select the mapping function (Figure 3), which consists of three parts:

Window The user can select a range of the data to be mapped, instead of the whole data domain. This makes it possible to eliminate outliers or to concentrate on certain parts of the data.

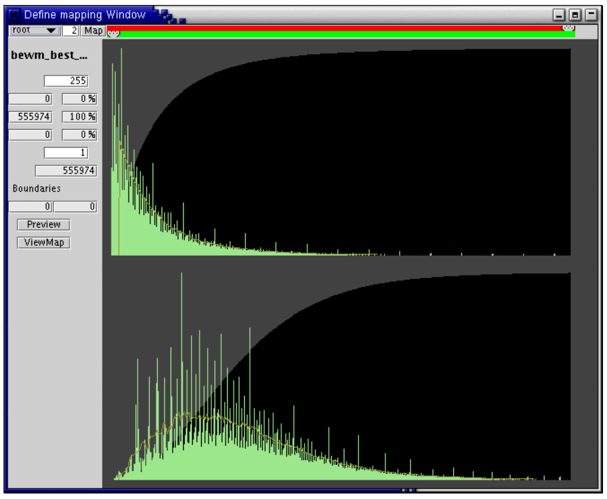


Figure 2: A histogram showing the effect of the mapping: the upper histogram is the distribution of the original data, the lower shows the effect of the mapping with a square root function. Feedback is immediate.

Gap All data values outside the defined window are mapped to values in visualization space that signify "above" or "below". These must be separated from the points that display mapped values, which is done by a gap.

Function type The most important part is of course the function type. It gives the user control over how the values are presented, and which parts receive more or less space. In Voxelplot, the user can select between linear, logarithmic, exponential, n-th power and n-th root mappings. Logarithmic and root functions give more room to the smaller part of the mapped range, while exponential and power functions compress the range and give more room to the higher values.

The selection of the function and the definition of the window is aided by a histogram preview (Figure 2) that shows the effect of the current mapping to the histogram of the dimension. This is done for each dimension separately.

In addition to the points that represent the actual data points, three axes are included in the 3D data. They have different colors (red, green, blue) so that they can be differentiated. Labelling of the axes is provided on the lower right of the user interface (Figure 4). Labels were not included in the volumetric display for several reasons. First, the size of the displayed volume does not allow enough space for labelling, which would have taken away valuable space and would have been very small. Second, the labels would have been unreadable (because they would be turned and tilted away from the user) most of the time, anyway. View-aligned labels would be a way out of this problem, but were not implemented in this prototype.

A special problem is posed by categorical dimensions,

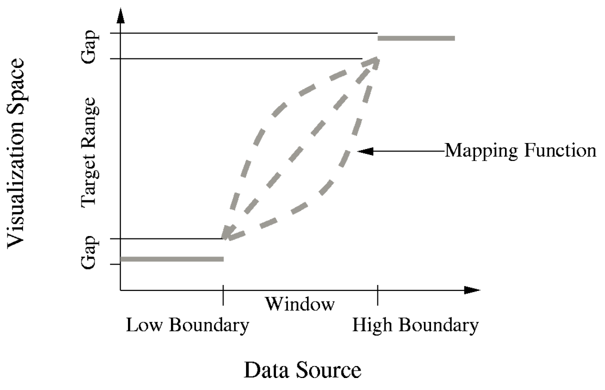


Figure 3: The mapping function. It consists of the data window that specifies the values to be mapped; the gap that separates unmapped values from mapped ones in visualization space; and the actual type of the function inside the window.

i.e., dimensions that only have a small number of different values. Typical examples for such dimensions are product categories, age groups, etc. Such dimensions must be shown properly so that the impression of categories is retained (and not smoothed or interpolated, which would distort this impression). But when such data creates very thin slices of data in the display, the structure of that data is hard to see. Therefore, categorical dimensions are best mapped to color, which is also the best use of color (because of the few colors humans can actually differentiate).

3.3 Rendering

The result of the mapping process is the input for RTVR, which is the volume renderer used. RTVR renders the data with a combination of splatting and shear-warp factorization.

One drawback of RTVR is the lack of depth cues other than transparency (which is the main depth cue in volume rendering). This is less problematic in the visualization of medical data, where objects can be seen whose three-dimensional shape is quite easily understandable. But in the case of InfoVis data with little structure, it is impossible to get an impression of depth from a static image. When the image can be rotated, however, the depth impression is very good (see Section 3.4). A color mapping that is redundant with one of the axes also helps in depth perception.

3.4 Interaction

In Voxelplot, there are two different kinds of interaction. The first are viewing interactions, which are mostly done directly with RTVR. The second are changes in the mapped dimensions, the mapping, brushing, etc., which are done with a separate control panel.

Voxelplot displays four 3D scatterplots at the same time (Figure 4), with the option to enlarge one of them to

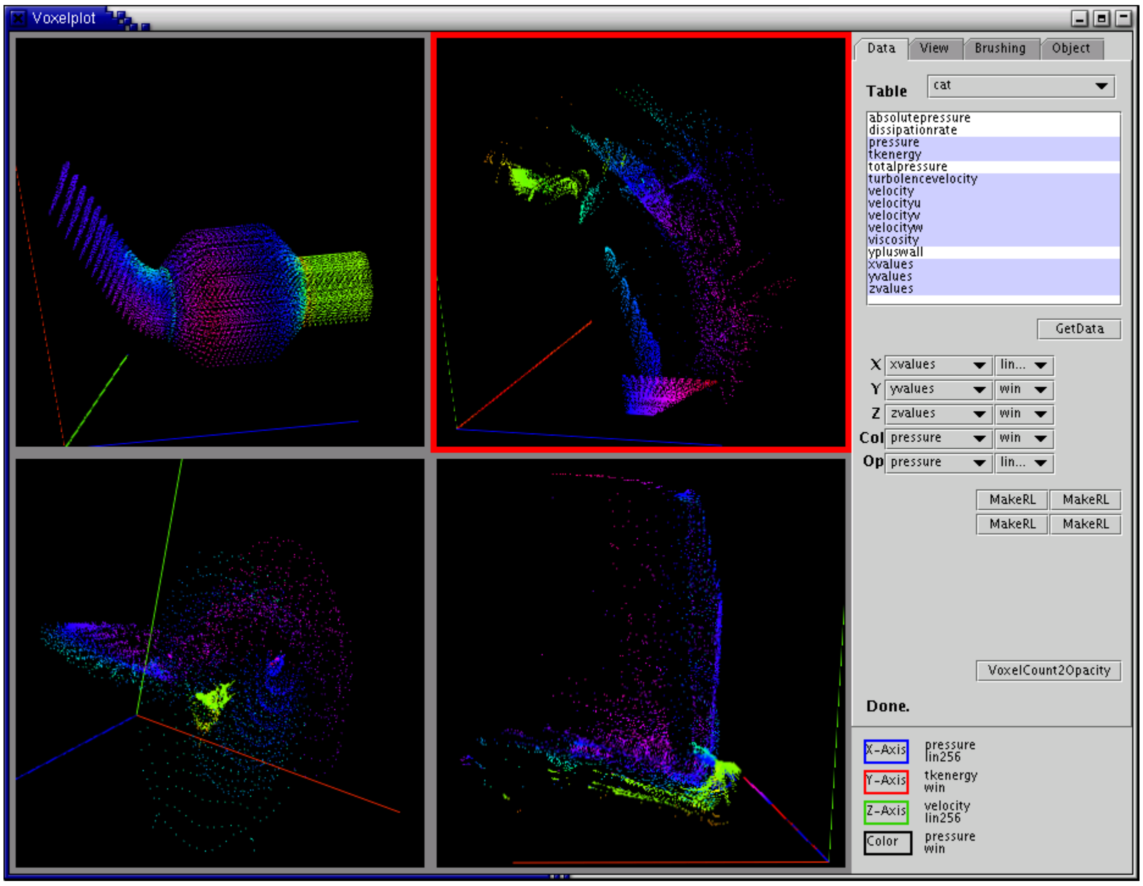


Figure 4: The whole Voxelplot application user interface. The red border marks the current view, its coordinate labels are shown in the lower right.

one large scatterplot. One plot is the current one, which the user interactions done with the interaction panel (on the right) apply to.

RTVR can handle objects in the volume data, which can be individually switched on and off, their parameters changed, rendered with different techniques (for two-level volume rendering), etc. Voxelplot provides mechanisms for combining several scatterplots in one RTVR view (where each plot is one object), changing their transparency individually, and deleting them. This way, direct comparisons can be made between different plots.

View Interactions

Constrained Viewing. RTVR provides basic interactions that are needed for any kind of interactive volume rendering. The user can rotate the object in three directions and also zoom into it. This is done with the mouse directly over the display, and feedback is immediate. Being able to rotate the object quickly is important for the user to get a 3D mental map of the object, because the impression of 3D is much stronger with motion as a depth cue[1].

[1]To get an impression of this, please also see the accompanying movie, available at http://www.vrvis.at/vis/research/voxelplot/vp.avi .

Voxelplot provides some additional viewing interactions that are simple but useful for the work with InfoVis data. The movements in the view can be constrained, so that only one component of the movement actually has an effect. This makes it easier to precisely turn the object so that a certain view is achieved.

The viewpoint can also be reset to be orthogonal to any of the axes in the view. This makes it possible to quickly return to the initial view, and also to get a first impression by looking at the three 2D projections. Because navigation in three dimensions is inherently difficult, these two enhancements are a great help to the user.

In addition, an animated rotation over 90 degrees orthogonal to any of the axes is also possible. This is useful when brushing a complex shape, where the user can “shoot” different parts with a beam brush (Section 3.4), one at a time.

View Linking. All views are linked, which means that brushing of data takes effect in all views at the same time. From the user’s point of view, this is useful, because brushing is usually used to find out which structures in one set of dimensions translate to which points in another. But this is also the more logical modus operandi because of the way brushing is implemented: the brush defines a DOI (degree of interest) function on the data, which necessarily affects all views.

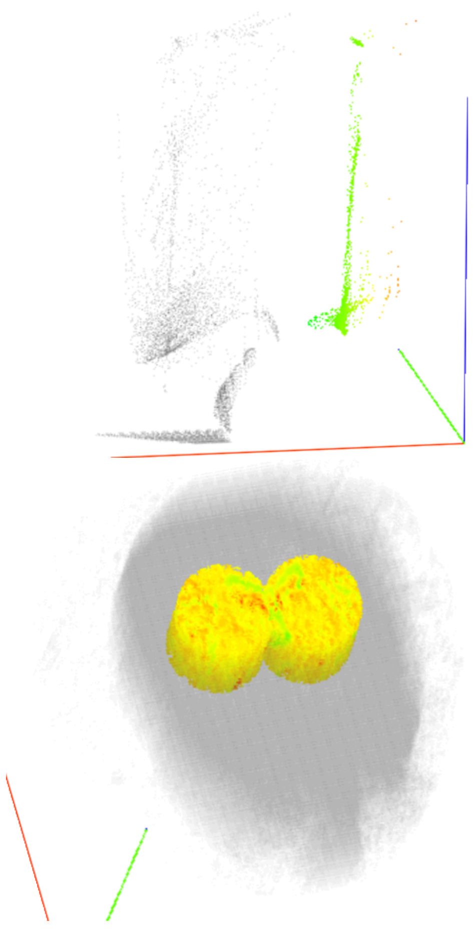


Figure 5: Brush Types. Top: Selecting high pressure in a catalytic converter dataset with a range slider in one dimension. Bottom: The result of two beam brushes through a dataset that represents a part of a CT scan of a head.

Transparency. The transparency of all voxels in a plot can be changed at the same time. This is useful when looking at data where the points are very dense (so that the effect of volume rendering can be really used), and when looking at plots with a categorical axis (see Section 5).

Brushing

In any of the scatterplots, the user can mark points, which are then labelled as interesting. This operation is called brushing in InfoVis, and is a very powerful tool. Brushed points are not only marked in their view, but the respective data points are marked in the original data, and the brushing information is passed to all other views (this is called linking). This way, the user can see the connections between the different dimensions, and which points in the different plots are the same data point in another view.

Different from systems like WEAVE, brushing can be done in any view, thus making interaction more flexible. By being able to brush the physical structure of the object, different hypothesis can be tested than when only the features can be brushed.

Brushing in 3D is more difficult than in 2D, because the user cannot just draw a rectangle around the interesting points. We therefore implemented two brushing techniques that are described in the following (see also Figure 5).

Range Brush. With a range slider[1], the user can specify a data range with respect to one data attribute by selecting a minimum and a maximum value with a slider widget. Multiple range sliders, defined on different axes, form a multi-dimensional brush, which creates a hyperbox.

Beam Brush. A more direct selection mechanism is the beam brush, which brushes all points that are inside a cylinder that lies perpendicular to the viewing plane, and whose radius the user can select. A typical selection with the beam brush works like this: the user turns the view, so that most points to be brushed lie behind each other, brushes a beam through the volume, turns the object to see which points were brushed (Figure 5b), and then uses further brushing operations with composite brushes (see below) to refine the selection. This is a very fast and intuitive operation that can be used to brush even very complex shapes easily.

Simple brushes alone are not sufficient, especially when brushing in three dimensions. We therefore also added composite brushes and other techniques that are described below.

Composite Brushes. Combining brushes is an important operation to be able to brush more complex shapes, and brush exactly the points one wants to select. The user can combine the effects of all brush types with each other, and can choose from three combination operators: OR, AND, and SUB (the latter subtract the new selection from the existing one). This set is simple to use and the user is able to select any subset of items. For example a beam brush and the interactive rotation combined with conjunctive combination of the brush forms a fast and intuitive process of selection for complicated shapes.

Focus+Context Visualization. Brushed data is displayed as usual (with the full color, opacity, etc.), while non-brushed data is semitransparent and gray. Non-brushed data is displayed and not just left out, so that it can serve as context for the brushed data – in InfoVis, this is called focus+context visualization (F+C).

Smooth Brushing. Conventionally, the boundary of a brush is a sharp edge, but this is not always appropriate. Often, data points cannot easily be classified to be of either full or no interest at all, but lie somewhere in between. Smooth Brushing [4] takes this into account by assigning an interest value that can not only have the value 0 or 1, but also any value between these extremes. Voxelplot can specify such fuzzy interest functions and the combination operators also work with them.

The result of brushing is the degree of interest (DOI) function, which assigns a value between 0 (not brushed) to 1 (brushed with 100% importance) to every data point. This function can be exported to the original data source and used in other applications, and is of course communicated to the other views. The definition of the DOI must be done on the original data points rather than on voxels to be able to also brush the respective voxels in views that show different dimensions.

4 Example: Catalytic Converter

The results of a simulation of the gas flow in a catalytic converter were visualized using Voxelplot. The data set consists of 9600 data points and 15 dimensions, among them the 3D coordinates of each data point, a velocity vector, pressure, turbulent kinetic energy (tkenergy), etc.

Generally, there are three questions the user wants to answer in scientific visualization: *Where are data of a certain characteristic?*, *What other features do these data have?*, and *What characteristics are present in a certain part of the object?* The first and last question lead from information to scientific visualization, and vice versa, while the second question can be answered with InfoVis alone.

Selecting low pressure areas in parameter space (Figure 6a) show where in the object these areas are (Figure 6b). From there, the analysis can be refined, e.g. by brushing one of the touched structures that are obviously present in the parameter space. When this is done, it turns out that they correspond to different parts of the catalytic converter (Figure 6c/d and e/f).

Because the structures are complex, they are brushed with combinations of beam brushes that add and subtract parts until the selection corresponds exactly to the structure under investigation.

We need to be able to brush from feature space to the spatial view as well as the other way to be sure that our analysis is correct. Only brushing a structure in feature space and seeing a part of the converter being brushed in the spatial view leaves the possibility that there are points in this part that are not part of the brushed structure in feature space (and that just were not visible between the brushed points). So to verify that the feature space structure indeed exactly corresponds to the part of the converter, it is necessary to brush in the spatial view and look for brushed points outside the structure that was originally brushed.

This analysis brought the structure of the multi-block simulation to light, where different parts of the catalytic converter are treated differently, and also the grids differ. The gaps between the features of adjacent parts of the grid suggest that a higher resolution could be useful, and more care should be taken at the interface between the parts to make the transitions smoother.

The results are difficult to characterize, because the discovered structures are complex. But this demonstrates how powerful the method is – even highly complex structures that are only discriminable in 3D can be found and separated.

5 Assessing Voxelplot

Even though there is a clear connection between volume and information visualization, there are of course differences. These differences must be taken into account to provide a useful visualization, and also make this program interesting.

In volume visualization, one usually looks at data from real objects, and these objects are visible in the data. The perception of 3D space works quite well when clearly defined objects are present, and not just single points that appear to be more or less randomly placed. In most areas, the user also knows what to expect from the visualization: medical data sets – even though there are variations between people, and diseases can change the shape and appearance of organs – are well understood, and the same is true for quality assurance of machine parts, aluminium foam, etc.

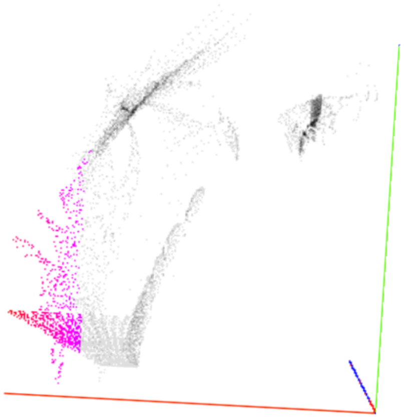
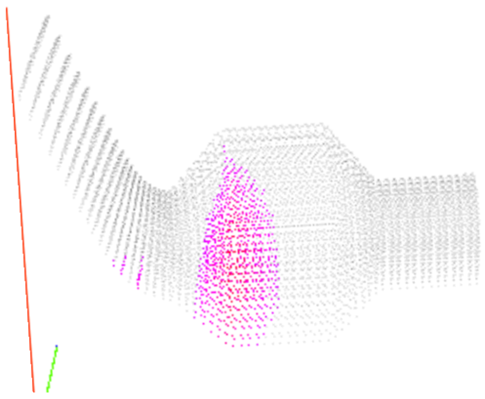
InfoVis data, on the other hand, can have any shape, and may not have any recognizable shape at all. Scattered single points are also not perceived as objects, and therefore much harder to see in the data. And even if there is a shape, the user does not know what to expect, since there is no a priori mental image of the data.

For this reason, interaction and depth cues are even more important for InfoVis data than for most volume data. But also the connection between the physical object and the more abstract dimensions is a big help, because users generally have an idea of the features they can expect in different parts of the object (like a catalytic converter).

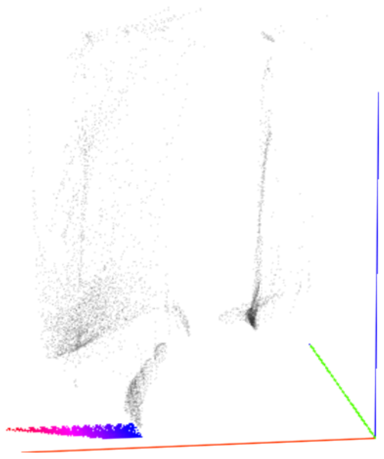
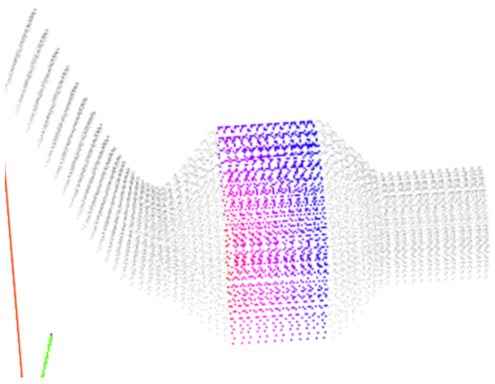
As a depth cue, color can be used which changes from blue (which appears to be in the back) to red (appears closer to the viewer). The effect is small but noticeable, and even provides a cue to the depth order (i.e., which voxel is closer than which other voxel) even when the object is rotated.

But there is also another aspect to the problem of possibly very scattered points: Single points might be anywhere, maybe in gaps between clusters of points or in areas where there is nothing else. Such points can be of great interest, but are very hard to see, especially on a dark background. Voxelplot therefore has a function that puts a halo [9] around each voxel, thus enlarging it and making it much harder to miss. This function also creates a stronger impression of objects being present in the data (because holes are closed). To avoid mistakes, this halo is white and thus easy to distinguish from the data points themselves.

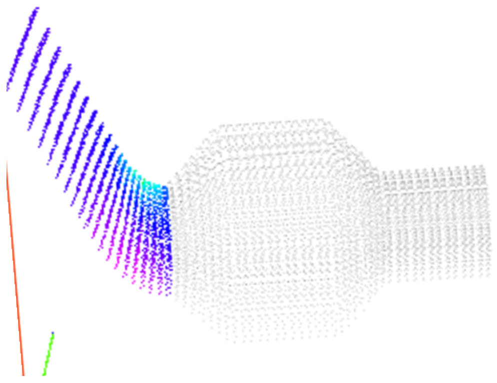
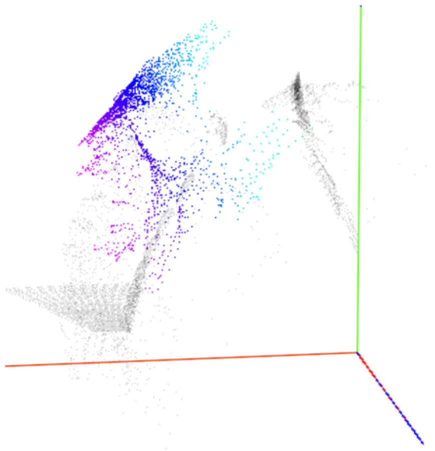
Another difference between the data presented in this paper and more common volume data is its dimensionality. Medical data sets are usually three-dimensional, as are CT data from many other applications. While rendering of such data is now quite well understood, work with higher dimensional data is quite another issue. Volume rendering is also optimized for different interactions than might be needed for higher dimensional data, like a fast switch between dimensions (which can involve heavy pre-processing) or several, concurrent views.

a) Selecting the low pressure areas in parameter space .... b) ... shows where the pressure is low in the physical object.

c) The lower part of the “spoon” ... d) ... represents the converter monolith.

e) The inlet of the converter ... f) ... maps to the structure that is above and in front of the “spoon”.

Figure 6: Examples of segmenting the catalytic converter data set in parameter space. The axes in parameter space are: pressure (red axis and color), velocity (green), tkenergy (blue)

Categorical data is not easy to deal with in scatterplots – 2D or 3D. In 3D, when there is only one axis with categorical data on it in the display, it effectively creates separate planes with 2D scatterplots, which can be useful independently of the rest. Comparing between these planes is possible in two ways. One can look at them from an oblique angle, and get a quick picture of the differences. And one can change the global transparency of the plot, and look at several planes at the same time, like a stack of layered slides (looking at them orthogonally to the categorical axis). By moving a range brush along that axis, one can switch back and forth between the layers (which act as context), and look at different planes to see differences and similarities.

In volume rendering, the source data are voxels, where they come from is secondary. In the case of Voxelplot, many values might fall into one voxel. The question is then, what features the voxel should represent: the mean of all the data points, the maximum, minimum, how many data points fall into this voxel, etc.

As a conclusion from all these points, it is clear that there are a few challenges when combining scientific and information visualization. But both sides can gain significantly from the knowledge and experience that has been collected in the other field, and from the techniques and concepts that have been developed.

6 Conclusions, Future Work

We have shown that information and scientific visualization can be integrated seamlessly and very flexibly through the use of a common method: interactive 3D scatterplots.

The combination of methods and ideas from these two different fields also makes efficient work with high-dimensional data possible and useful to engineers. 3D scatterplots can also deal with data sets that are usually considered large in Information Visualization (over one million data points).

Undoubtedly, this work is only a first step, and a lot of work remains to be done. Perhaps the most important now is to provide more depth cues to the user, like perspective projection, fog, and stereo viewing.

Acknowledgements

This work was done in the scope of the basic research on visualization (`http://www.VRVis.at/vis/`) at the VRVis Research Center in Vienna, Austria (`http://www.VRVis.at/`), which is funded by the Austrian research program Kplus. The example dataset is courtesy of AVL List GmbH, Graz, Austria.

References

[1] Christopher Ahlberg and Ben Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Human Factors in Computing Systems. Conference Proceedings CHI'94*, pages 313–317. ACM, 1994.

[2] Barry G. Becker. Volume rendering for relational data. In *IEEE Symposium on Information Visualization (InfoVis '97)*, pages 87–91. IEEE, 1997.

[3] William S. Cleveland. *The Elements of Graphing Data*. Wadsworth Inc, 1985.

[4] Helmut Doleisch and Helwig Hauser. Smooth brushing for focus+context visualization of simulation data in 3D. In *10th International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG 2002)*, pages 147–155, 2002.

[5] Steven Feiner and Clifford Beshers. Worlds within worlds: metaphors for exploring n-dimensional virtual worlds. In ACM, editor, *Third Annual Symposium on User Interface Software and Technology (UIST)*, pages 76–83. ACM Press, October 1990.

[6] George W. Furnas and Andreas Buja. Prosection views: dimensional inference through sections and projections. with a discussion by John F. Elder IV, Shingo Oue and Daniel B. Carr and a rejoinder by the authors. *Journal of Computational and Graphical Statistics*, 3(4):323–385, December 1994.

[7] D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung. WEAVE: A system for visually linking 3-D and statistical visualizations, applied to cardiac simulation and measurement data. In *Proceedings Visualization 2000*, pages 489–492. IEEE, October 2000.

[8] Helwig Hauser, Lukas Mroz, Gian-Italo Bischi, and Eduard Gröller. Two-level volume rendering. In *IEEE Transactions on Visualization and Computer Graphics*, volume 7(3), pages 242–252. IEEE Computer Society, 2001.

[9] Victoria Interrante and Chester Grosch. Strategies for effectively visualizing 3d flow with volume lic. In *Proceedings of Visualization 1997*, pages 421–424. IEEE Computer Society Press, 1997.

[10] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings IEEE Visualization 2001 (Vis'01)*, pages 255–262. IEEE Computer Society Press, 2001.

[11] Lukas Mroz and Helwig Hauser. RTVR - a flexible java library for interactive volume rendering. In *IEEE Visualization '01 (VIS '01)*, pages 279–286. IEEE, 2001.