PV251 Visualization

Autumn 2024

Study material

Lecture 9: Basic interaction concepts

Interaction in the context of information visualization is a mechanism by which we can influence what and how the user is seeing.

Interaction techniques

There are several classes of interaction techniques:

- **Navigation** - the user changes the position of the camera and scales the view (what is mapped to the screen). An example is rotation or zooming.
- **Selection** - identifying a specific object, set of objects or area of interest, to which we then apply certain operations, such as highlighting, deleting, or modification.
- **Filtering** - reduces the size of the data we map onto the screen - by removing records, dimensions, or both.
- **Reconfiguration** - the user can change the way data is mapped to graphical entities or attributes. An example is data reorganization or data distribution. In this way, we provide the user with different views of the displayed data set.
- **Encoding** - the user changes graphic attributes, such as point size or line color. The goal is to reveal various properties of the data.
- **Merge** - the user can use tools to merge different views or objects to display related items.
- **Abstracting / concretization** - level-of-detail modification.
- **Hybrid techniques** - combining some of the above techniques. For example, increasing the screen space that is used to display the detail of certain data, while reducing the space devoted to "uninteresting" data that is displayed to preserve context.

Until now, several techniques and tools have been developed for interacting with data and generally visualizing the information. Although some of them appear to be completely unrelated to others, they share basic principles and have a common goal. In this lecture, we will try to outline the basis of interaction techniques. We start with identifying the classes of interactive operations, which we describe as **operators**, and then define the so-called **operand** (the space to which the operator is applied).

We will now describe in more detail several interaction operations that are commonly used in the visualization of data and information. The list will not be exhaustive, of course, but should cover typical interaction techniques. A more detailed and comprehensive description of other techniques can be found in Keim's classification (http://nm.merz-akademie.de/~jasmin.sipahi/drittes/images/Keim2002.pdf) or the taxonomy of Ed H. Chi (http: // www -users.cs.umn.edu/~echi/papers/infovis00/Chi-TaxonomyVisualization.pdf).

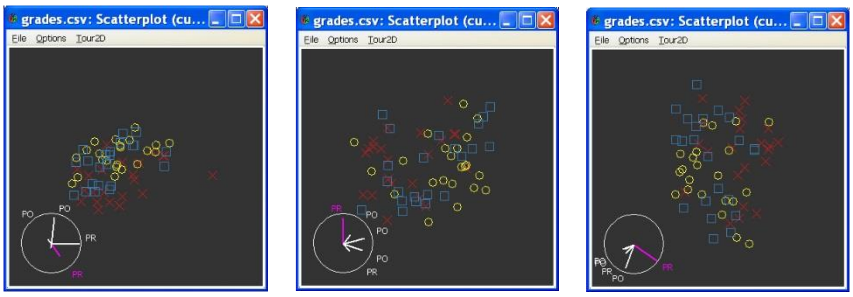
It should be mentioned that individual interaction operators can be part of several proposed classes of interaction and that almost all operators can be automatic in a given visualization, even in its non-interactive part. An example is zooming, which is available in almost all visualizations. We can look at zooming as generating a new visualization, especially if we have to display different data.

Navigation operators

Navigation (sometimes referred to as exploration) is used to find a subset of the input data to be explored, the orientation of the view of that data, and the level-of-detail (LOD). The subset being queried may be determined using a particular visual pattern, or it may be a piece of data that is subject to further detailed examination. In 3D space, navigation is typically determined by the position of the camera, the direction of the view, the shape and size of the viewing frustum, and the degree of LOD.

In visualizations that support multiple resolutions at once, the LOD corresponds to descending in the data hierarchy. Navigation operators can work with absolute or relative coordinates in a given space. Navigation can be automatic, or user controlled.

An example of automatic exploration is flying along a path over multidimensional data, which covers most or even all possible orientations of the data space when projected into 2D space (see figure). The user can influence the step size between views.


grades.csv: Scatterplot (cu...
File Options Tour2D
PO
PR
grades.csv: Scatterplot (cu...
File Options Tour2D
PR
PO
grades.csv: Scatterplot (cu...
File Options Tour2D
PR
PO

Selection operators

When selected, the user isolates a subset of the components for display, which are further subject to other operations, such as highlighting, deleting, masking, or moving to the center of the area of interest. So far, various selection variants have been developed and we usually choose the appropriate variant so that we have to decide what result we expect. For example, should the new selection replace the existing one or should it rather supplement / enrich it?

The **granularity of the selection** is also one of the main topics. By clicking on a given entity on the screen, we can select only the smallest addressable component (e.g., vertex on the edge) or we can select a wider region around the selected location (e.g., the whole object, area on the screen, surface, …).

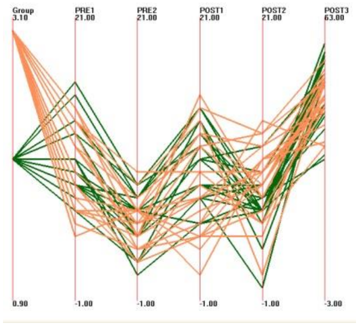
The choice can be determined in many ways. The user can click on individual entities, "draw" through the selection of entities (e.g., by holding the mouse button while moving over objects of interest) or isolate entities in some other way (for example, by selecting a rectangle, lasso, …).

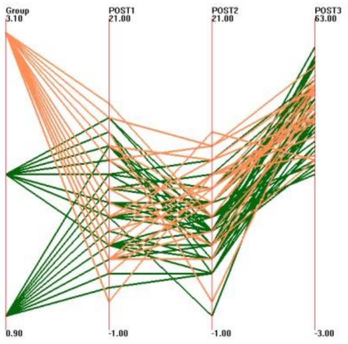
Selections can also be created indirectly when the system selects elements that match a set of user-specified restrictions. An example is selecting nodes in a graph that have a certain maximum distance from the selected node.

Filtration operators

Filtering, as the name suggests, reduces the number of data to be displayed by setting various restrictions that specify which data will be retained and which will be deleted. An example of such a filter is the so-called dynamic query specification, which was described by Shneiderman et al. (http://www.cs.umd.edu/~ben/papers/Shneiderman1994Dynamic.pdf).

To determine the extent of interest in the data, sliders are specified, during the manipulation of which the visualization is immediately updated to reflect changes caused by the user. This method of querying by setting ranges is only one way of applying filtering. Another method selects the items you want to keep or hide. An example is the function of hiding columns in Excel.





The figure shows the use of filtering to simplify the view of data and their interpretation. Rows and columns are filtered using the XmdvTool.

The difference between filtering and selection followed by deleting or masking is small, but essential. Filtering is most often done **indirectly** - for example, the filter specification is not performed on the data visualization itself, but in a separate dialog box or interface. Filtering is often done before the data is displayed to avoid displaying too much data on the screen.

In contrast, the selection is most often made **directly**, when we mark the displayed objects, for example, by clicking the mouse into the scene. Further operations performed on the selected set may lead to a result that is the same as using filtering.

Reconfiguration operators

Reconfiguration of data in a given visualization is often used to reveal properties or to deal with the complexity or scale of the data. By reorganizing the data, such as filtering out certain dimensions and rearranging the remaining ones, we can provide the user with various new views of the data. An example is a tabular visualization tool that sorts rows and columns to highlight trends and correlations between data.

Another example of reconfiguration may be to change the dimensions used to control the x and y coordinates of the drawn marks.

Popular methods of data reconfiguration are the previously mentioned PCA (principal component analysis) or MDS (multidimensional scaling), which try to preserve the relationships between data in all dimensions when projected into a lower dimension (often 2D).

Encoding operators

Any dataset can be used to generate countless different visualizations. Data properties that are not visible in one visualization method may be apparent when using another type of visualization. For example, for a particular dataset, the use of a scatterplot may lead to point overlap, while, for example, when using parallel coordinates, the points may have a unique representation.

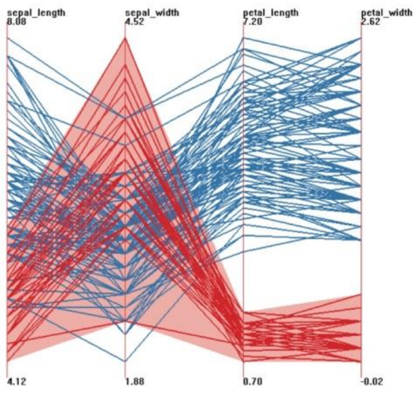
Many visualizations today support several types of visualizations at the same time because a single type of visualization cannot capture an effective view of all the tasks that the user wants to perform on data. Each of the visualizations is best suited for a certain subset of data types, their properties, and user-specified tasks.

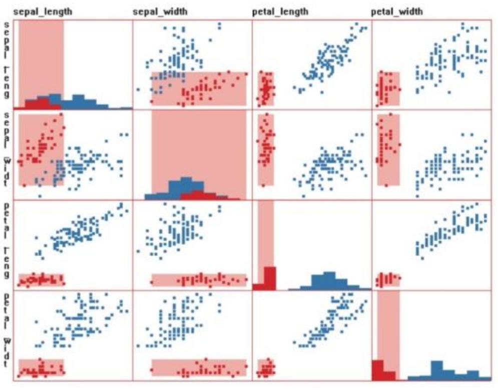
Data encoding is performed by various types of mappings and views of the data, by which the user can properly examine the data. Other forms of coding operations may, for example, modify the color map, the size of the graphic entities, or their shape. This can be considered as different variations of a given type of visualization and helps to reveal areas of interest. By using various variations of the technique, we can overcome several limitations of the visualization technique used. For example, the problem of overlapping points in point diagrams can be eliminated by shaking the points or selecting the size of the individual points in such a way that the size of the point is derived from the number of points in the same position. Other attributes of graphic entities that we can affect include transparency, texturing, line or fill style, but also dynamic attributes, such as loss of intensity or blink rate.

It should be noted that these effects can often be mimicked by using transformations directly on the data instead of on their graphical representation.

Aggregation operators

A common use of selection operations is to combine selected data in one view with corresponding data in other views. There are many ways to connect between the panes of a given application, however, probably the most used form of communication between windows in modern visualization tools is the so-called **linked selection**. The popularity of this form is mainly because each of the data views can reveal interesting properties and that highlighting one of these properties in one view can help build a "richer" mental model of this property when viewed in other views (see figure). In parallel coordinates, we select the examined cluster, which is displayed in the matrix of corresponding point diagrams using a rectangle.





If it is allowed to change the data selection interactively, this operator is called **brushing**, where the user can continuously change the selection in one view, and the corresponding combined data in the other views are highlighted. Another strength of the linked brushing technique is the specification of complex constraints for a given selection. Each type of view is optimized to emphasize a certain type of information. For example, we can specify time constraints when using a visualization that includes a timeline, restrictions on field names when using a list view, and geographic constraints for maps.

In certain cases, the user may want to unlink some visualizations, leaving the view, but we want to explore a different area of data or a different dataset. Some systems allow the user to specify for each window whether information should be transmitted to other windows or from which windows that window receives input.

Some types of interaction can be local to a given window (e.g., zoom), while others are shared between all windows (e.g., reordering dimensions).

Abstraction/concretization operators

When displaying a large amount of data, it is often useful to focus only on a certain subset of data, about which we display details (concretization), while on other parts of the data we reduce the degree of detail (LOD) (abstracting). One of the most popular techniques of this type is distortion operators. While some scientists classify these distortions as a visualization technique, they are in fact a transformation that can be applied to any type of visualization. Like zooming or panning, distortion is suitable for interactive data exploration. Many **distortion operators** (also called functions) have been proposed in the past. This includes methods that deform the entire analyzed space or methods that apply deformations only locally.

The distortion can be part of the original visualization or can be displayed in a separate window. Distortion differs in the ratio of the properties they retain to the input data. For example, text distortion techniques seek to be as legible as possible in a given area of interest, and the rest of the text is given mainly to maintain the structure of the document but may not be legible.

Distortion operators can be linear or nonlinear, with zero, first or second order continuity (discontinuous operators can also be used). Operators can be applied to structures instead of contiguous spaces, and therefore can be specific to a given type of operand (see below).

Different operators have different footprints, such as the shape or extent of space affected by a transformation. Common fingerprint shapes are rectangular or circular, to which hyperboxes and hyperellipses correspond in higher dimensions. The extent of the affected space is usually specified by a distance function within the deformed space and is often multidimensional. These ranges can be fixed or variable, user controlled, or information semantics.

Interaction operands and spaces

The parameters of the interaction operators we have described so far will be discussed even further. Before that, we will show the categorization of interaction operands, which will help clarify the role that these parameters play in the interaction process and their semantics within different spaces.

The interaction operand is the part of the space to which the interaction operator is applied.

To determine the result of an interactive operation, we must know within which space the interaction will be performed. In other words: if the user clicks on a given place or area on the screen, which entities do they actually want to mark? These can be pixels, data values or records mapped to a given location or part of a visualization structure (for example, an axis).

Several different classes of interaction spaces have been identified. We will now describe these spaces, including examples of existing interaction techniques that fall into each class.

We distinguish several basic interaction operands.

- Screen space (Pixels)
- Data value space (Multivariate data values)
- Components of Data Organization
- Components of Graphical Entities
- Object space (3D Surfaces)

Screen space (pixels)

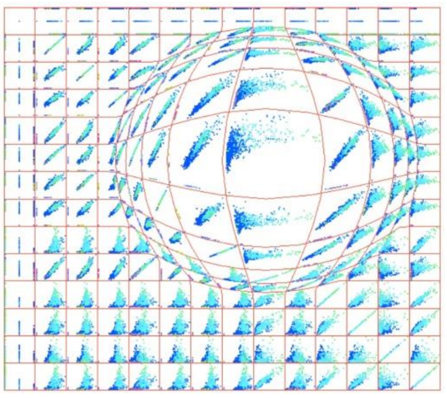
Screen space navigation typically consists of actions such as panning, zooming, or rotating. In none of these cases do we use any new additional data - the process consists of pixel-level operations such as transformation, sampling, and replication.

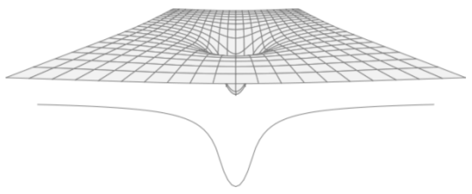
Pixel-level selection means that at the end of a given operation, each pixel is classified as selected or unselected. As previously mentioned, the selection can be made over individual pixels, rectangular or circular areas of pixels, or over areas of any shape that the user defines. Selection areas can also be continuous or discontinuous.

Distortion in screen space involves a transformation on pixels, for example $(x', y') = f(x, y)$.

Magnification $m(x, y)$ at a given point is simply a derivative of this transformation and is useful for switching between transformations and their magnifications during the distortion process. Examples of screen space techniques are fisheye, rubber sheet (relief).

An example of this type of distortion is shown in the figure. The matrix of graphs is shown on the left, and the rubber sheet on the right.





Let's take a closer look at one of the typical examples of this distortion: the fisheye. The implementation of the fisheye is relatively straightforward. It is necessary to specify the center point (c_x, c_y) of the transformation, the radius of the lens (magnifying glass) r_l and the amount of deflection d. Then the coordinates of the image are transformed into polar coordinates relative to the center point. The magnifying glass effect is obtained by a relatively simple transformation applied within the radius r_l.

One of the popular transformations of this type is given by the formula:

$$r_{new} = s \log(1 + d(r_{old}))$$

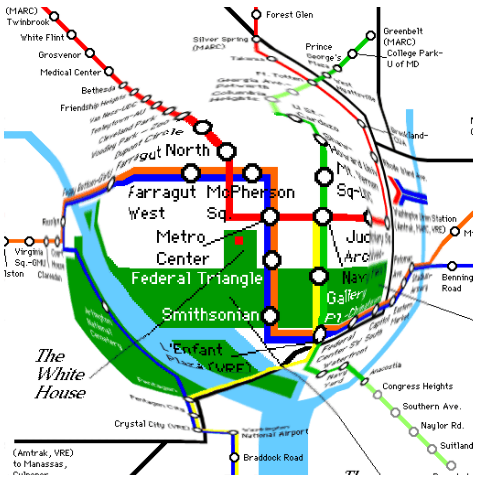
where

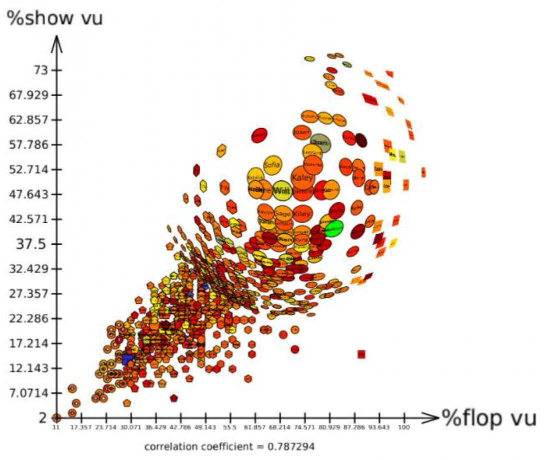
$$s = \frac{r_l}{\log(1 + d * r_l)}$$

This formula ensures that the radius of the points located at the edge of the magnifying glass is maintained at its original value. Now let's look at the pseudocode of the whole algorithm:

```
1. Clean the output image.
2. For each pixel of the input image:
    1. Calculate the corresponding polar coordinates.
    2. If the radius is smaller than the radius of the magnifying glass:
        1. Calculate the new radius $r_{new}$.
        2. We get the color of this place from the original image.
        3. Set this color as the pixel color in the output image.
    3. Otherwise, set the resulting pixel of the image to the same value as in the
       original image.
```

Depending on the type of transformation used for the distortion in the screen space, we have to deal with the holes or, conversely, the pixel overlap that occurs during the transformation. Pixel overlap does not cause as much problems as holes. Nevertheless, overlapping pixels are often averaged in specific implementations. Holes must be solved using interpolation. Here it depends on the type of magnifying glass used - for text visualization, a linear function with a "flat" center part is most often used for interpolation, so that the text in this part can be read without problems.

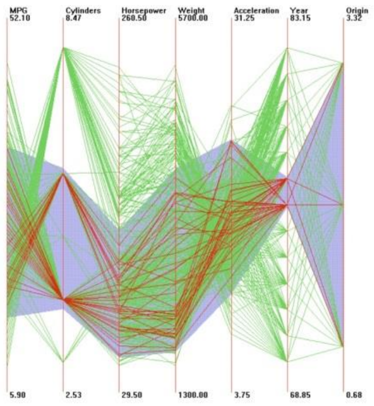




Space of data values (Multivariate data values)

Navigation in the data value space involves the use of these data values as a view specification mechanism. Analogous panning and zooming operations are used to change the displayed data values - panning shifts the starting position of the range of values for display, while zooming increases / decreases the size of this range.

Selection in the data value space is similar to database queries, where the user specifies a range of data values for one or more dimensions. This can be achieved by direct data manipulation, such as data-driven brushing (see figure) or by using sliders or other query specification mechanisms.

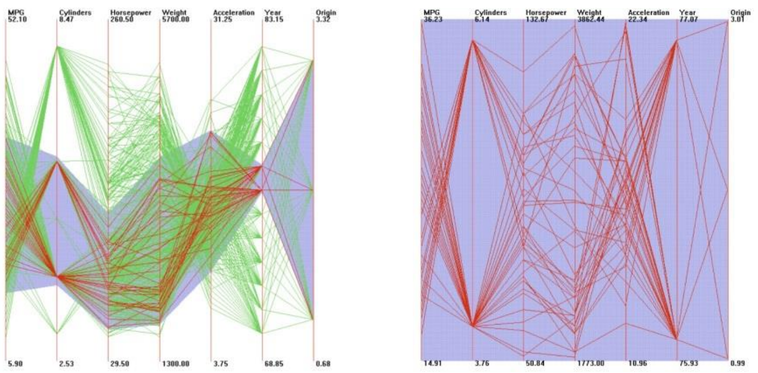


The data value space is probably the most intuitive space in which filtering is performed. When visualizing large data sets, it is common practice to reduce the data first. For spatial data, this is analogous to trimming data falling outside the viewing region. For non-spatial data, this reduction corresponds to deleting some records, dimensions, or both.

Dimensions can also be filtered to allow users to explore a subset of dimensions with similar properties or to select representatives for individual cluster clusters.

When distorted in the data value space, the data values $(d_0, d_1, ..., d_n)$ are transformed using the j: $(d'_0, d'_1, ..., d'_n) = j(d_0, d_1, ..., d_n)$ function. This transformation takes place before the visualization itself. In fact, each of the dimensions may be subject to its own transform function $j_i : d'_i = j_i(d_i)$. In the most general case, the function can depend on any number of dimensions, although in such a case the possibility of controlling the filtering by the user is problematic.

An example of data space distortion is the previously mentioned XmdvTool tool, where each dimension of a selected subset of data is scaled in such a way that the subset can be displayed in screen space (see figure).



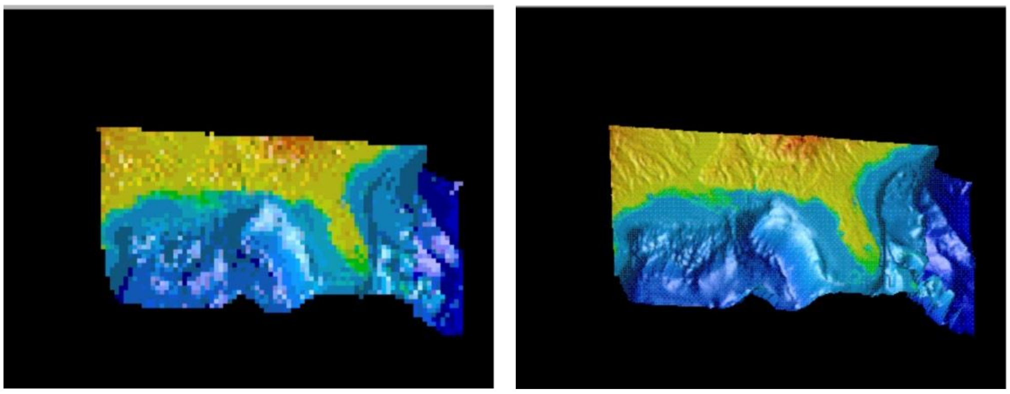
In the picture, the N-dimensional hyperbox is selected and then scaled across all dimensions, thus filling the unit hypercube.

Space of data structures (Components of Data Organization)

Data can be structured in many ways, such as into lists, tables, grids, hierarchies, and graphs. For each of these structures, it is possible to develop a special interaction mechanism that determines which parts of the structure we will manipulate and how this manipulation will manifest itself.

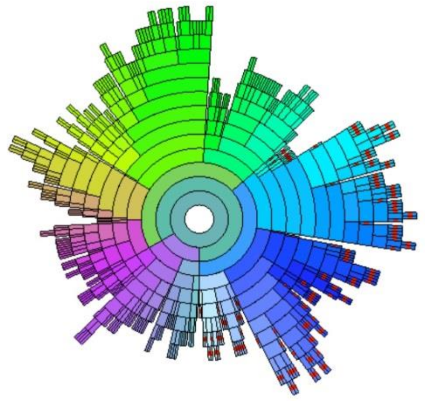
Navigating in the space of data structures involves moving the view specification along the structure, such as when displaying sequential groups of records or navigating a hierarchical structure (operations of moving up or down in the structure).

As an example, let's take a picture that shows the difference between zooming in screen space (left - involving pixel replication) and zooming in data structure space (right - involving getting more detailed data at the desired resolution).



Selection in the space of a data structure generally involves displaying the structure and allowing the user to identify areas of interest within the structure. An example is structure-based brushing, which allows you to control the selection of data stored in a cluster hierarchy. An example of an interaction in this case is to highlight data that falls into a particular branch of a tree.

Another example is the InterRing visualization tool, which displays the hierarchy radially and by filling a space. This tool allows semi-automatic selection of nodes, due to their hierarchical structure.



The figure shows an example where end nodes are automatically selected. This selection was created by querying the end nodes above their common ancestor.

Here, too, filtering is often used to reduce the amount of information displayed. For example, for visualizations over time, it is common practice to define a range on the timeline that we want to focus on. Exploring the environment in graph visualizations, in turn, often involves filtering nodes and edges that are further than the user-specified number of "hops" (meaning the number of edges) from a given point of interest. For hierarchical methods, filtering is based on the level of the hierarchy.

Distortion on hierarchical structures is very common, especially due to the density of information, which is derived from broad or deep hierarchical structures. Several scientists have focused on techniques based on mapping the hierarchy radially.

In all these cases, the data is stored in the structure instead of in the data values themselves or in the mechanism as they are visualized.

Formalization of this procedure is more complicated than in other cases. However, most distortions can be defined by mapping a vector (D, S), where D is data and S is a structure that stores that data, to a vector (D', S'), where the transformation can modify the data, the structure, or both.

Now consider the arrangement of dimensions for the visualization of multivariate data. Fully manual techniques in this case may involve manipulating the text entries in the list (using a move-up and down operation or using a drag-and-drop technique). In the case of parallel coordinates or matrices of scatter plots, we can manipulate the axes directly. For a matrix of scatter plots, moving one element can cause changes in other rows and columns if, for example, we want to maintain symmetry along the main diagonal.

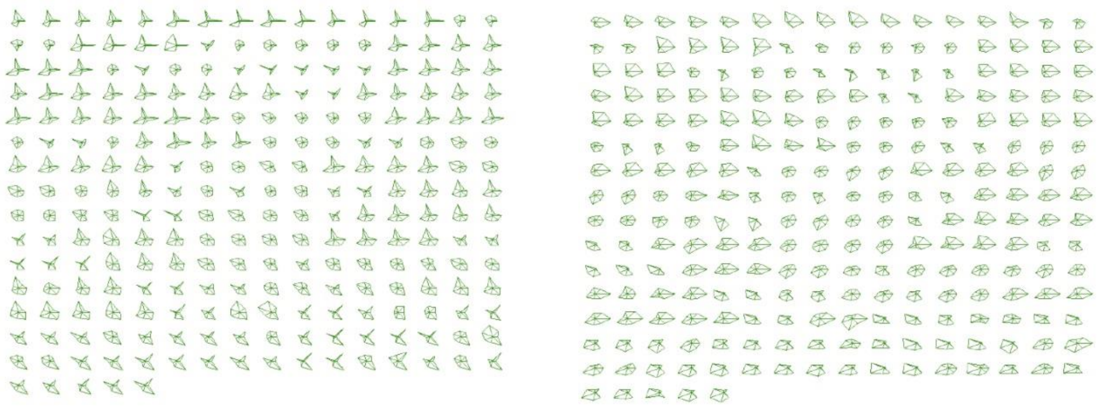
In contrast, automatic dimensioning requires at least two basic design decisions: how to measure the quality of an arrangement and what strategy to choose to find those quality arrangements. Different metrics can be selected for these decisions. One of the commonly used is the sum of the correlation coefficients between each pair of dimensions.

This correlation coefficient between the two dimensions is defined as follows:

$$\rho_{X,Y} = \frac{\sum (x_i y_i - n\mu_X \mu_Y)}{(n-1)\sigma_X \sigma_Y}$$

where *n* is the number of data points, X and Y are two dimensions, x_i and y_i are the values for the i-th data point, μ_X is the mean value in X, and σ_X is the standard deviation for X.

Another approach to measuring the quality of an arrangement may involve simplicity of interpretation. Different dimensional arrangements can lead to representations with larger or smaller visual clusters or structures. For example, it is stated that using glyphs representing data points makes it easier to analyze simple shapes instead of complex ones. Therefore, if we are able to measure the average or cumulative complexity of a shape (e.g., by counting depressions or vertices of a shape), we can use this knowledge to compare the visual complexity of different dimensional arrangements.



An example is in the picture, where the left part shows the original arrangement, while the right part shows the results after rearranging the dimensions, where we try to reduce the concave areas of glyphs and increase the percentage of symmetrical shapes.

If we have selected the required quality of arrangement, the next task is to find an effective search strategy for finding these quality arrangements. The evaluation of all possible arrangements of dimensions is very demanding, if the number of dimensions is not very small (the number of individual arrangements is N!). This number can be divided by 2 if we consider symmetric solutions, but we still have to evaluate a huge number of possibilities. A typical strategy in these situations is to use optimization techniques. The problem of arranging dimensions is very similar to the problem of a business traveler, so we can directly use the algorithms used for this problem.

One of the simplest algorithms works as follows:

1. Select any two different dimensions
2. Swap their positions and calculate the quality of the arrangement

3. If the quality is lower than the quality of the original layout, we will cancel the swap

4. Repeat steps 1-3 with a fixed number of iterations or until a certain number of tests performed show no improvement in quality

These heuristic approaches are definitely not optimal, but they often lead to finding an acceptable solution. These approaches can also be combined with a manual approach, where the user can enter some arrangements manually based on his knowledge of the data set and then let the system automatically calculate a quality arrangement on his modified set.

Space of attributes (Components of Graphical Entities)

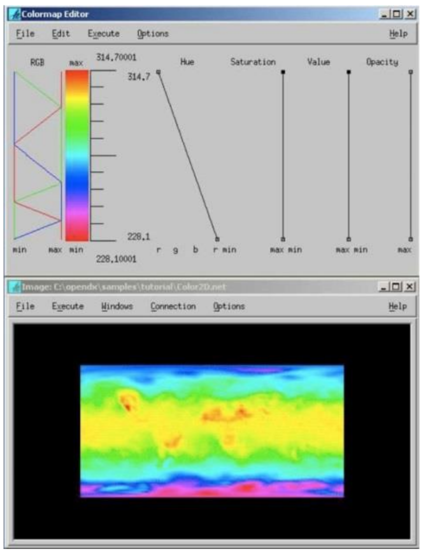
Navigating in the attribute space is very similar to navigating in the data value space. Panning involves shifting the range of values of interest, while zooming can be achieved either by scaling attributes or increasing the range of values of interest.

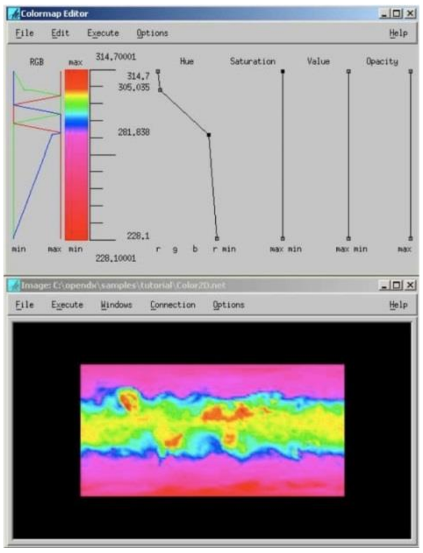
As with data-driven selection, selection in the attribute space requires the user to specify a subset of those attributes of interest. For example, if we have a color map representation, the user can select one or more entries to be highlighted. Similarly, if data records contain attributes such as quality or uncertainty, then a visual representation of those attributes accompanied by appropriate interaction techniques allows the user to filter or highlight data based on those attributes.

In the attribute space, remapping often occurs - either by selecting different ranges of a given attribute, or by selecting different attributes for a given input set. For example, in GlyphMaker, a user can select to map a given data dimension from a list of possible graphical attributes.

Let's have the A attribute of a given graphic entity. We can perform a distortion transformation by applying the function k: a’= k (a). We can assume that A can take values from the range [a_0 → a_1] or that A is specified as a vector. For example, color map distortion can allocate a wider or narrower range of values for some subsets, increasing the readability of subtle deviations (see figure). The figure shows the attribute distortion in the form of a color map modification, which was generated using the color map editor in the OpenDX system. The color map is distorted in such a way that a larger range of values is assigned to the center of the data range.

This form of distortion is often used in the analysis of medical images to identify areas of interest.


Colormap Editor
File Edit Execute Options Help
RGB max 314.70001 Hue Saturation Value Opacity
314.7
228.1
min max min 228.10001 r g b r min max min max min max
Image: C:\opendx\samples\tutorial\Color2D.net
File Execute Windows Connection Options Help


Colormap Editor
File Edit Execute Options Help
RGB max 314.70001 Hue Saturation Value Opacity
314.7
305.035
281.838
228.1
min max min 228.10001 r g b r min max min max min max
Image: C:\opendx\samples\tutorial\Color2D.net
File Execute Windows Connection Options Help

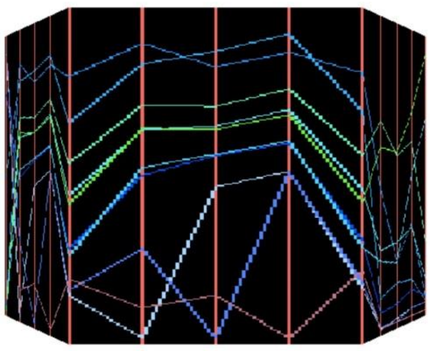
Space of objects (3D Surfaces)

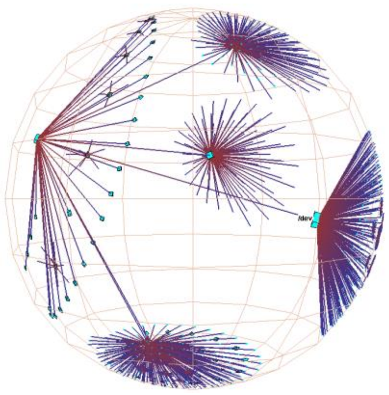
In this view, the data is mapped to geometric objects, and this object (or its projection) is subsequently subject to interactions and transformations. Navigating in the space of objects often consists of moving around objects and observing the surfaces on which the data has been mapped. A system that supports navigation in the object space should allow a global view of the object as well as detailed views. These can be limited in some way to quickly offer the user "good views" of the object.

Selection involves clicking on objects of interest or selecting target objects from a list.

A typical example of remapping in object space is to change the object to which the data is mapped, such as switching the mapping of geographic data from a plane to a sphere and vice versa.

Examples of distortion in this form of interaction are the so-called perspective walls (left) and hyperbolic projections (right). These methods can be seen as variants of the method based on the screen space, where the object on which the data is projected encapsulates the distortion function. However, after mapping is applied, surfaces may undergo other 3D transformations, such as rotation, scaling, and perspective distortion.





The process of distortion in the space of objects can be represented as a sequence of two functions:

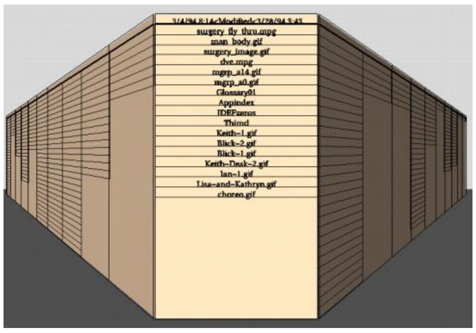
- The first maps the data (generally parameterized in two dimensions) to a 3D structure:

$$(x, y, z) = g(a, b)$$

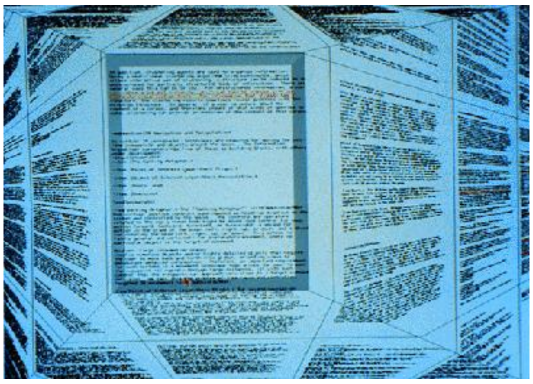
- The second transforms this structure and projects it on the screen:

$$(i, j) = h(x, y, z)$$

One of the methods for navigation in the visualization of a large number of documents and data are the so-called **perspective walls**. This approach shows one panel of the view of the surface of the object located orthogonal to the direction of view and the other panels are oriented in such a way that they disappear with distance, in a way defined by perspective deformation.



A simplified version of a perspective wall can be created so that the front wall implements horizontal scaling of a portion of the mapped 2D image, while adjacent segments are subjected to horizontal and vertical scaling proportional to their distance to the edge of the front wall. In addition, a chamfer is applied to these segments.



Thus, if the left, middle, and right sections of the original image to be drawn on the perspective wall are delimited by (x_0, x_{left}, x_{right}, x_1) and the left, middle, and right panels of the resulting image are defined as (X_0, X_{left}, X_{right}, X_1), then the applied transformation is as follows:

for $x < x_{left}$:

$$x' = X_0 + (x - x_0) * \frac{(X_{left} - X_0)}{(x_{left} - x_0)}$$

$$y' = (X_{left} - x') + y\left(1 - \frac{(X_{left} - x')}{(X_{left} - X_0)}\right)$$

for $x_{left} \le x < x_{right}$:

$$x' = X_{left} + (x - x_{left}) * \frac{(X_{right} - X_{left})}{(x_{right} - x_{left})}$$

$$y' = y$$

for $x \ge x_{right}$:

$$x' = X_{right} + (x - x_{right}) * \frac{(X_1 - X_{right})}{(x_1 - x_{right})}$$

$$y' = (x' - X_{right}) + y\left(1 - \frac{(x' - X_{right})}{(X_1 - X_{right})}\right)$$

When using a perspective wall, the user can interact with it by sequentially scrolling through the "pages" (forward and backward). It is also possible to use indexes for direct access (jumping) to the area of interest - often this is implemented as a tab protruding at the top of the page at the beginning of each section.

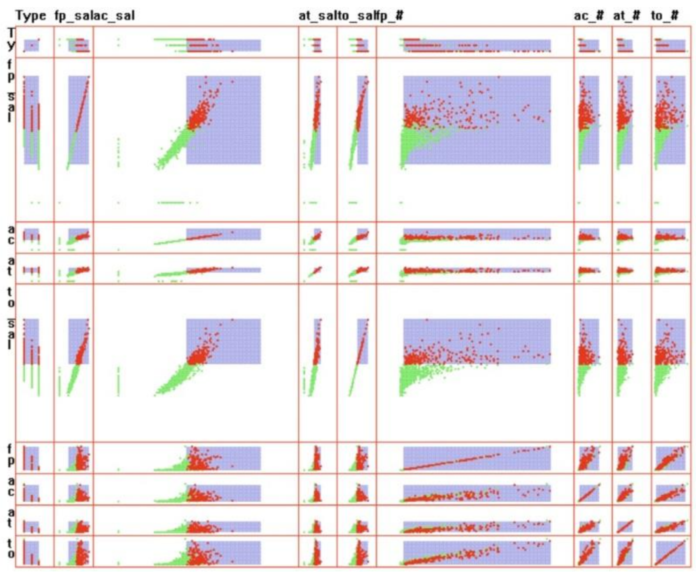
Space of structure visualization

Visualization focuses on a structure that is relatively independent of values, attributes, and data structure. For example, a grid into which a matrix of scatter plots is drawn, or axes displayed in different types of visualizations, are components of the visualized structure on which the interaction focuses.

An example of navigation when visualizing a structure space is scrolling pages in a tool based on tabular visualization or zooming to individual graphs in a matrix of scatter graphs.

When selecting, typical operations involve selecting components to be hidden, moved, or rearranged. For example, a user can select an axis of parallel coordinates and drag it to a different position to discover different relationships between data dimensions.

An example of distortion in this space is the so-called table lens technique, which allows the user to transform rows and / or columns of a table to provide multiple LODs. This process applied to a matrix of scatter plots is shown in the figure.



The image generated by the TableLens tool shows a matrix of scatter plots in which two cells (and their corresponding rows and columns) are enlarged at the expense of the other cells.

Animation transformations

Basically, all interactions within visualization systems lead to a change in the displayed image. Some of these changes are quite dramatic - for example, when opening a new dataset. Other changes may preserve some aspects of the view and change others. In case the user wants to keep the context while focusing on the changing area, it is desirable to provide a smooth transition between the initial and target visualization. For example, when rotating with a 3D object or dataset, a smooth change of orientation is generally much better than a step transition to the final orientation. In some cases, it is sufficient to use a simple linear interpolation between the initial and final configuration. In other cases, however, linear interpolation does not lead to a constant rate of change (for example, when the camera moves along a curve). In addition, in most cases we get a much more attractive result using gradual acceleration and slowing down of change. In this section, we will focus on the algorithms we must use to achieve this change control.

The first step in algorithms for controlling changes in data is to obtain a uniform parameterization of the variable or variables that we want to control during the animation. For some variables, such as straight line positioning or scaling, we use linear interpolation to provide consistent changes over time. For other variables, such as positions along a curved path, we need to reformulate the problem by introducing a new parameter.

Suppose that the original parameter is a function of the variable *t*, which takes values from 0 to 1. For example, we can use a cubic polynomial to calculate the (x, y) position for different values of t:$x(t) = At^3 + Bt^2 + Ct + D$ (same for y). Now we can create a list of positions p_i for $0 <= i <= n$, where *n* is the number of steps between the start and end positions. Division of *t* into *n* same subintervals is reached by simple assignment of the corresponding t values to the parametric equation.

Then we can estimate the length of the arc A by the sum of the distances between successive points:

$$A = \sum_{i=1}^{i=n} dist(p_{i-1}, p_i)$$

Obviously, the smaller the step between adjacent points, the more accurate the estimate of the arc length.

However, for most curves, the distance between adjacent points is different. Therefore, if we always used the approach described above, the speed of the resulting animation would be variable.

When calculating the length of the arc, it is also useful for each point p_i to calculate the distance di from the beginning of the curve to this point.

Then we calculate the function A(i), which represents the percentage of the distance the point travels in the i-th time step.

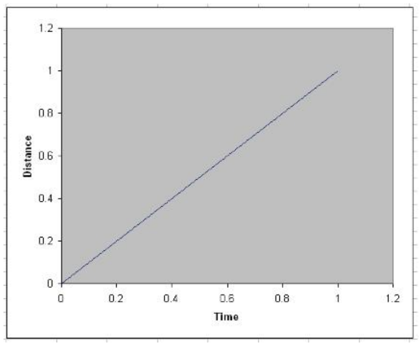
For simplicity, we will use the variable t (0.0 <= t <= 1.0) instead of the variable. Next, we define a new parameter s = A(t). We store the results in a table, so for each value of t we know its corresponding value s = A(t). We use the value of s to determine the uniform velocity (using linear interpolation).

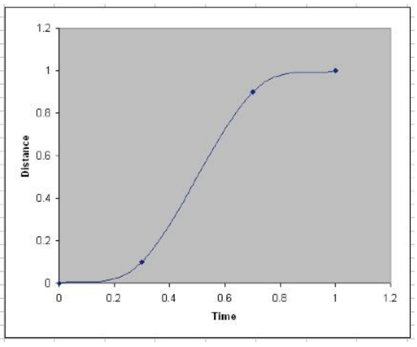
The above procedure is referred to as **reparameterization**. The s parameter is now used to control the speed. When we draw the parameter s as a function of time, we get a straight line leading from the beginning (0.0, 0.0) to (1.0, 1.0). In other words, at the beginning of the animation we start in the original position and when the time reaches 1.0, we are in the end position. The speed simply corresponds to the slope of this curve. But what about cases where the curve is not straight but curved? Then the parts with a small slope represent a low speed and, conversely, a large slope represents a high speed.

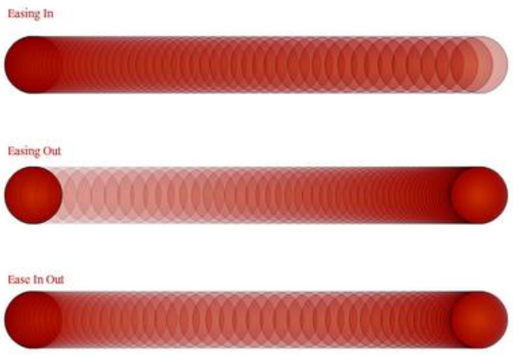
Because the start and end points are fixed, we are assured that we end where we want.

To specify the animation between the start and end point, we have an infinite number of settings. We can even stop the animation for a while. The main assumption is that the curve increases monotonically, and now let's also assume that it cannot go back.

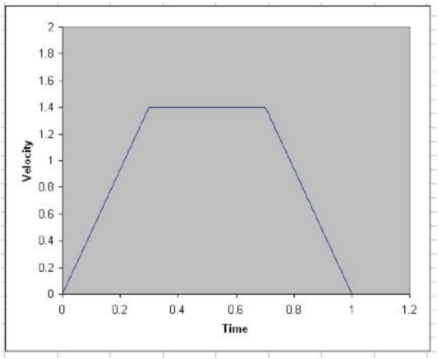
A common type of animation control curve is a curve that represents a gradual increase in speed at the beginning of the animation from zero to the desired speed, and then again, a gradual decrease in speed to zero at the end of the animation. This behavior is exactly represented by the sine curve. The key requirement is to maintain a smooth curve.





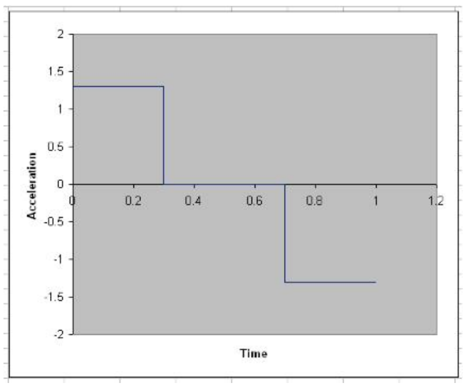


Sometimes it is easier to specify motion using a **velocity curve**. Velocity is simply the first derivative of the position curve. The speed curve for the case of gradual increase and decrease of speed consists of a segment that increases evenly from zero, then a straight segment and at the end of a descending segment ending at zero (see figure).



The space under the curve must correspond to a value of 1.0 - because if the required speed is too large, we have to spend more time in the ascending and then descending part.

A third type of curve that is sometimes used to control motion is the **acceleration curve**. Corresponds to the second derivative of the position curve or analogously to the first derivative of the velocity curve. The shape of the curve representing the gradually increasing and then decreasing velocity is composed of three horizontal line segments - one above the axis (positive acceleration), one on the axis (constant velocity) and one below the axis (deceleration).



The relative positions and lengths of the lines above and below the axis can be used for different effects and are not necessarily symmetrical. However, the spaces defined by the ascending and descending phases must be equal.

The position curve and the velocity and acceleration curve can be used to control any attribute that changes during the animation.

Virtual reality

Interaction in three-dimensional space is much more complicated than on a 2D screen, because we have to take the depth of the scene into account when interacting. Navigation in such an environment must work with six degrees of freedom. In addition to the virtual scene, we must also visualize the user's position and direction. Also, the selection of objects in the virtual scene is different - you need to work with the 3D menu.

The virtual environment has great advantages in terms of:

- Navigation - we can also work with head movements

- Interactions - data gloves, optical motion tracking can be used,...

- Stereoscopic projections and depth perception - polarizing glasses, active glasses, head mounted displays,...

- Nesting the user into the scene - the user is completely surrounded by the virtual world (glasses, specialized rooms - CAVE)