Feature Extraction and Iconic Visualization

Theo van Walsum* Frits H. Post† Deborah Silver‡ Frank J. Post†

Abstract

We present a conceptual framework and a process model for feature extraction and iconic visualization. The features are regions of interest extracted from a data set. They are represented by attribute sets, which play a key role in the visualization process. These attribute sets are mapped to icons, or symbolic parametric objects, for visualization. The features provide a compact abstraction of the original data, and the icons are a natural way to visualize them. We present generic techniques to extract features and to calculate attribute sets, and describe a simple but powerful modeling language which was developed to create icons and to link the attributes to the icon parameters. We present illustrative examples of iconic visualization created with the techniques described, showing the effectiveness of this approach.

1 Introduction

The main goal of visualization is the extraction of meaningful features from large data sets. Direct presentation of data leaves the extraction of these features to the eye and brain of the user. But often the number of features is small compared to the amount of data, and it is useful to support the process of visual feature extraction by algorithmic techniques. In this article we will describe an approach to visualization based on algorithmic feature extraction, and visual representation of these features using symbolic objects, or icons. This article is an extension of our earlier work [18]. In this article, we elaborate on feature extraction and icon design, and we give new examples.

The two important aspects of this visualization approach are feature extraction and the mapping of features to icons. A *feature* can be defined as a region in a data set that is of interest for its interpretation. The features are extracted and represented by a set of characteristic parameter values: an attribute set. The attribute sets are

*Currently at: Laboratory for Clinical and Experimental Image Processing, Department of Diagnostic Radiology, Leiden University Hospital, Rijnsburgerweg 10, Building 1, C3Q50, Leiden, The Netherlands

†Delft University of Technology, Faculty of Technical Mathematics and Informatics, Julianalaan 132, 2628 BL Delft, The Netherlands

‡Dept. of Electrical and Computer Engineering, Rutgers University, P.O. Box 1390, Piscataway, NJ 08855-1390, USA

an abstract representation of the original data, as they represent the data at a higher level. Feature extraction can proceed in multiple stages, resulting in higher levels of abstraction. The next step is the mapping of features to icons. The attribute sets are linked to the parameters of symbolic objects. We call these symbolic objects *icons*.

An icon is an object with parametric geometry and appearance that can be arbitrarily linked to data quantities. The function of an icon is to act as a symbolic representation, which shows essential characteristics or features of a data domain to which the icon refers. This reduction to essentials is the main purpose of iconic visualization: to replace the original data by a symbolic representation that is more clear, compact, and meaningful, and which can be related to the physical concepts of an application [17].

The paper presents a conceptual framework and a process model of feature extraction and visualization using icons. The main purpose of this paper is to introduce the notion of a data abstraction process. In Section 2 we discuss the concept of icon, and give a classification of icon types. In Section 3 we describe how a visualization process model can be adapted for feature extraction and iconic visualization and in Section 4 we show techniques for data selection and feature extraction. Section 5 presents examples of feature attribute calculation using volume integrals. In Section 6, we present a simple iconic mapping and modeling technique. How feature extraction can be represented as a multi-level process of data abstraction is described in Section 7. In Section 8 the use of these techniques is demonstrated with some applications. Finally, in Section 9, we present conclusions and directions for further research.

2 Iconic Representation

2.1 The icon concept

The general meaning of the term 'icon' is an image or sign which has a characteristic in common with the thing it signifies [15]. Icons have been studied extensively in many fields, including theology, art history, logic, the theory of signs or semiotics [9], and in pictorial information systems [5]. In the context of scientific visualization, Hesselink and Delmarcelle [12] related the icon concept to classical sign theory, and have given a taxonomy of icon types for vector and tensor field visualization.

The icon concept is related to Abstract Visualization Objects [10], and Parametric Geometric Objects used for computational steering [14]. The term 'glyph' has also been frequently used [19], which is roughly comparable to the term 'icon' as it is used here. We do not draw a sharp distinction between these terms, but we prefer the term icon, especially when it refers to distinct, macroscopic, multi-parameter objects.

Early applications of iconic visualization can be found in statistical data visualization, such as Kleiner-Hartigan trees [4] and Chernoff faces [6]. These techniques are used for high-dimensional data that are defined in an abstract data spaces, rather than in Euclidean physical space. The use of symbolic representations may be more obvious in abstract data or information spaces, as objects and patterns in physical spaces can also be visualized in more 'naturalistic' ways. The notion of an iconic visualization may

be common to physical visualization, statistical data visualization, and information visualization.

Examples of iconic representations in scientific visualization are glyphs used by Helman and Hesselink [11] to indicate location and type of critical points in a vector field, and cylindrical objects to visualize vortex tubes extracted from flow fields [2, 25]. Ellipsoids have also been used as icons for feature visualization [22]. Our purpose in this paper is to demonstrate the generation and use of macroscopic icons for visualization of features and feature attributes.

2.2 Icon types for visualization

In scientific visualization, several types of icons have been used. To characterize the different types, we will briefly discuss some aspects of iconic representations.

A first aspect is the number of *parameters*, or *degrees of freedom*, that can be varied for the icon, and separately bound to data quantities. The parameters can be divided into three groups: *spatial* parameters (position and orientation), *geometric* parameters which control the shape of the object, and *descriptive* parameters such as color, texture, transparency, or sound.

Icons can be *fixed-template*, whose basic shape is predefined, or *amorphous-template* without a predefined shape. A fixed-template icon contains a set number of defining parameters that are used to vary the basic shape. For example, the length and direction of an arrow icon in a 3D vector field is determined by the vector value at a single point. The position, direction, and length of the arrow are varied to reflect the direction and magnitude of the local vector; however, the shape of the arrow will always be similar. Other examples of fixed icons are stick figures [8, 24], the ellipsoid [10, 22], and the flow probe [13]. The basic shape of a fixed-template icon must be designed. The shape of amorphous-template icons is not predefined, but is fully determined by the local properties of a data field. Examples of such icons are streamlines, isosurfaces, stream surfaces, tensor field lines or hyper-streamlines [12], and vortex tubes [2, 25]. All of these icons do not have a fixed set of defining parameters.

The *reference domain* is the area that is represented by a single icon. Local means a point and its immediate environment; global means throughout the domain. Intermediate is the level of a sub-volume, at a scale between local and global, which can apply to features or selections [27].

The *dimension* of an icon depends on the dimension of the object space in which it exists. Time is considered as a separate dimension. Time-dependent variations of icon parameters are shown as animated motion, deformation, or changes in color of an icon.

The *display scale* refers to relative size and distinctness. Micro-icons are discrete but are not individual objects. They can only exist in large numbers, as elements of dot or line patterns or as texture elements; meaning emerges from them only in large numbers. Examples at this level are point markers, arrows, stick figures [8, 24], and particles. At the macro scale, icons are individual objects representing data attributes from a given reference domain. Examples of this type are 3D solids [19], wedges [7], ellipsoids [22],

and the flow probe [13].

A last aspect is the use of icons for *interaction*. A passive icon is used for display purposes only. Position and appearance are dictated by the data. An icon that can be interactively positioned at a chosen reference point can be used as a probe [13]. If the data binding is in two directions, icons may be used as input objects to control model parameters in the interactive steering of simulations [14].

The icons discussed in this paper are fixed and amorphous template, with an intermediate-scale reference domain. They are mostly 3D static or animated, macroscopic, and passive (non-interactive).

3 Iconic visualization process

The visualization pipeline [10] is a generally accepted model of the visualization process. The stages of this model are data enrichment and enhancement, visualization mapping and rendering and display. In the data enrichment and enhancement stage, data are preprocessed to be more suitable for visualization. Visualization mapping is the stage at which data quantities are mapped to visual quantities. At the rendering and display stage, an image of these visual primitives is generated and displayed on a screen. For the iconic visualization process, a similar process model can be used (see Figure 1). The goal of the iconic representation is to get a 'summarized' visualization, therefore the first step in the iconic visualization process is to find items in the dataset that need to be represented in the summary, i.e. regions of interest for interpretation (*features*). Simultaneously, or in the next processing step, characteristic parameter values of the features (*attributes*) are calculated. This is what we call *feature extraction* and *attribute calculation*. The result of this feature extraction is a set of attribute values that characterize the feature. This is called an *attribute set*; each characteristic value is an attribute of the feature. As the attribute set represents the data at another, higher level, this is a process of data-abstraction.

The feature extraction and attribute calculation stage can be implemented in different ways. Either special algorithms can be developed to extract specific features from data and calculate the attribute sets (see e.g. [2, 25]) or general selection and segmentation techniques can be used to identify features in the data [22, 23, 27].

In the iconic mapping stage, the attributes are mapped onto the parameters of *icons*. The purpose of this mapping is to visualize features by objects, that display the characteristics of a feature in a clear and understandable way: there should be some similarity relation between the features and their iconic representations. The way the attributes are mapped onto the icon's parameters determines the appearance of the icon.

One important consequence of this process is a vast data reduction. The original data field is replaced by a usually small number of attribute sets, visualized by simple geometric objects. The reduction is caused by the data selection and abstraction, and is necessary for understanding the information inherent in very large data sets.

For example, the size of a typical CFD data set will be in the order of 10-100 Mbytes

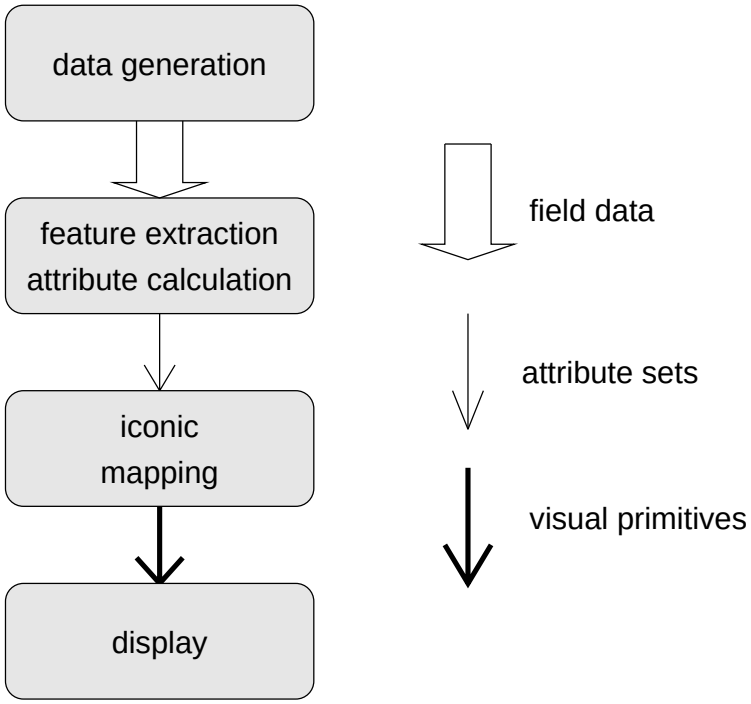


Figure 1: Iconic visualization pipeline.

per time step. For 100 features of 20 data items, the feature/attribute dataset will be in the order of 10 Kbytes. This attribute set represents one specific view of the data set, which may for its purpose replace the original data set, and thus means a reduction by a factor of at least 1000. An icon will be in the order of 100s of polygons, so the total visualization may be about 10^4 to 10^5 polygons, which can be displayed by a graphics workstation in real time.

The size of the intermediate feature/attribute data set is very small, and this provides an excellent opportunity for distributed processing. The computationally and data intensive first part of the process can be executed on a remote high performance computer, and the results can be easily transferred to the visualization workstation by a low-bandwidth link.

4 Feature extraction

Features are regions of interest in the dataset. These regions can be a single grid location or a set of nodes from the dataset. There are many different ways to extract these feature-sets from a dataset. In this section, we focus on generic techniques for the extraction of features from a dataset. A subset of nodes X is created by evaluating some feature criterion, $H(x)$ over all grid nodes x of the dataset. A node is included if the result of $H(x)$ is true for that node.

The function $H(x)$ depends upon the scientific domain and the features of interest. It can be a Boolean criterion function, consisting of a logical combination of scalar

thresholds, applied to data values or to derived data values such as gradients [27, 26]. For example, a criterion can be informally stated as: "select the data points where velocity is high and pressure is low". This criterion can be formalized to: the magnitude $|\mathbf{v}|$ of the velocity vector $\mathbf{v}$ is greater than a given threshold t_1, and the pressure value p is less than a given threshold t_2. The thresholds can be expressed as fractions of the global maximum values for the data set: $t_1 = r_1\,|\mathbf{v}|_{\max}$, and $t_2 = r_2\,p_{\max}$, with r_1 and r_2 between 0 and 1. Then $H(x) = (|\mathbf{v}| > t_1)$ and $(p < t_2)$. The criterion function $H(x)$ can be applied to all grid nodes, yielding a selected subset X of the grid nodes. The set of grid nodes then defines the feature.

Alternatively, we can combine threshold and connectivity criteria in $H(x)$. This results in common segmentation techniques (adapted from image processing for this purpose), that can be used to extract connected components from a dataset [21]. The subset X resulting from the above criterion evaluation consists of a collection of nodes. X can be segmented into coherent regions by applying an additional connected component algorithm, to group adjacent selected grid nodes. From a selected seed node, the neighboring selected nodes are visited and recursively added to a group if a given connectivity criterion is satisfied. This will yield a number of distinct regions, that can be labeled for reference, and used individually in further processing.

The region selection operation generally requires checking all values in the dataset. However, when certain operations are performed regularly, data structures (such as the octree) can be used to speed up the operations. In addition, the connectivity criteria can be implemented with region filling techniques.

5 Attribute calculation

Attribute sets are sets of characteristic values computed from the extracted features in a dataset. An attribute set can consist of a combination of scalars, vectors, and tensors. When a feature is larger than a single grid position or cell, relevant attributes such as volume, center, mean data value, and moments can be calculated by integration over this region using volume integrals. Thus volume integrals offer the possibility to calculate many different aggregate data values for selected regions. Attributes can be classified as being either purely geometric or a combination of geometry and the underlying data value. Below are a selection of attributes which can be computed with volume integrals (S is a region that belongs to a selection, V_S is the volume of that region, $\mathbf{x}$ a position vector, ρ a density (scalar) field and $\mathbf{v}$ a vector field and $\mathbf{v}^T$ the transpose of $\mathbf{v}$):

- *Geometric / morphological attributes*: describe the spatial properties of a feature, such as width, height, volume, centroid, and shape

geometric attributes of S	**integral**
volume (V_S)	$\int_S dS$
center	$\frac{1}{V_S}\int_S \mathbf{x}\,dS$

$$\text{2nd moment} \qquad \frac{1}{V_S}\int_S \mathbf{x}\,\mathbf{x}^T dS - \frac{1}{V_S}\int_S \mathbf{x}\,dS\ \frac{1}{V_S}\int_S \mathbf{x}^T dS \tag{1}$$

- *Combined morphological / data attributes*: these are related to both spatial and data properties of a feature. Examples include the center of gravity in a density field, average velocity, etc.

combined attributes of S	**integral**
mass (M_S)	$\int_S \rho\,dS$
center of gravity	$\frac{1}{M_S}\int_S \rho\,\mathbf{x}\,dS$
average	$\frac{1}{V_S}\int_S \mathbf{v}\,dS$
variance/covariance matrix for $\mathbf{v}$	$\frac{1}{V_S}\int_S \mathbf{v}\,\mathbf{v}^T dS - \frac{1}{V_S}\int_S \mathbf{v}\,dS\ \frac{1}{V_S}\int_S \mathbf{v}^T dS$

(2)

For the calculation of these volume integrals, we use quadrature rules. The integral $\int f(x)dx$ is approximated by evaluating $f(x)$ at sample points $x_1 \ldots x_n$, so that $\int f(x)dx = \sum W_i f(x_i)$. The number of samples, the sample positions, and the values for W_i are determined by the type of quadrature (e.g. Newton-Cotes and Gauss-Legendre). The one-dimensional integration process can be extended to volumes (grid cells), by calculating the function to be integrated at $n \times n \times n$ sample points around a selected node and summing over the volume.

6 Icon modeling

The purpose of iconification is to transform a feature in a dataset into an iconic object. The feature and its icon are linked by the feature's attributes: the attributes are mapped to parameter values for iconic objects. The type of icon and the binding determine the appearance of the feature in the visualization. For iconic visualization, one needs either a large set of icons, sufficient for the visualization of features for a specific application area, or an icon design system, with which researchers can develop their own iconic representation of features, and bind the feature attributes to the icon parameters.

6.1 Icon modeling language

In order to facilitate the construction of icons, we developed a simple but powerful icon modeling language, with which we were able to generate a wide variety of icons. With this language, geometric primitives can be defined and the parameters of these primitives can be bound to the attribute set. The geometric primitives are constructed with this language in three steps:

1. A 2D contour in the x-y plane is designed and colors or color-maps are bound to this contour. This contour can consist of line-segments or parametric functions. The coordinates of the line-segments, the constants of the parametric functions and the colors or color-maps can be bound to the attribute set.

2. This contour is extended to a 3D object by performing a rotation sweep around the x or y axis, performing a translation sweep by moving the contour along an axis over a given length, or performing a general sweep along a arbitrary 3D trajectory [3]. The parameters of the function that defines the 3D trajectory can also be bound to the attribute set.

3. The 3D object is scaled, translated, rotated or deformed. In this way the objects can be put on their location in the dataset or to their relative location in a complex object. The values defining the rotation, translation, scaling or deformation are also bound to the attribute set.

For each step in the construction of an icon, the language provides a set of commands. Furthermore, there are macros that can bind elements from the attribute set to parameters for the icon. In these macros, an array is used for representing the attribute set. The elements of this array are used as parameters in an arbitrary parametric function. A symbolic name is assigned to the evaluation result of this function, and this symbolic name can be used as a parameter in the icon geometry definition macros. In this way, a complex relation can be established between the attribute set elements and the parameters of the icon.

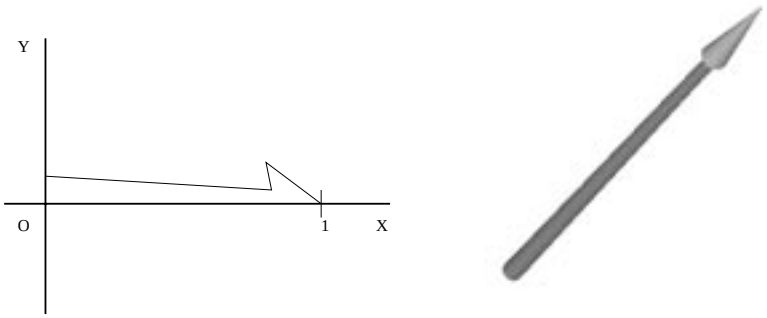
Figure 2 shows how an arrow is constructed using this procedure. The parameters of the arrow are the base position in space (p_x, p_y, p_z), the length l, and two angles (α, β). First a contour is drawn in the x-y plane. The base of the arrow is drawn at $(0, 0)$, the tip at $(1, 0)$. This contour is rotation-swept around the x-axis, scaled by l in the x direction, rotated around the y-axis with α, rotated around the z-axis with β and translated over (p_x, p_y, p_z).

A wide range of icons can be generated with the modeling language. A collection of icons is shown in Figure 4.

6.2 Ellipsoids

The ellipsoid (Figure 3) is a good icon for multiparameter fields. An ellipsoid is defined by a position (p_x, p_y, p_z), the three lengths of the main axes (l_1, l_2, l_3) and three angles (α, β, γ) which define the orientation of the ellipsoid. Therefore, ellipsoids are good icons for mapping to regions which contain three orthogonal vector attributes. One example is a 3D second-order symmetric tensor field (which contains a 3×3 matrix at each grid location). The eigenvalues and eigenvectors of the tensor define the axes and orientation of the ellipsoid centered at that grid location. When the tensor field is defined at every grid location, many small ellipsoids result much like an arrow-vector field.

Larger regions can also benefit from ellipsoid iconification. An obvious way of repre-




```
Bind_XY(ArrowTop, 1, 0)            /* bind the positions */
Bind_XY(ArrowHat1, 0.8, 0.04)
...
Bind_RGB(YELLOW, 255, 255, 0)         /* bind the colors */
...
/* bind transformations, a[i] is attr. vector ... */
Bind_Scale(ArrowScaleX, sqrt(a[3]*a[3]+a[4]*a[4]+a[5]*a[5]))
Bind_Trans(T012,a[0],a[1],a[2]) /* binding of T012        */
...
XY_SetPen(ArrowTop,YELLOW),  /* construct the 2d geometry */
XY_LineTo(ArrowHat,YELLOW),  /* construct the 2d geometry */
...
Sweep_R_X(15),                  /* sweep contour around    */
                                /* x-axis in 15 steps      */

/* perform 3d transformation */
XYZ_Scale(ArrowScaleX),         /* scale 3D object         */
XYZ_Rotate(AnglesFromVec_3_4_5),/* rotate 3D object        */
XYZ_Translate(T012),            /* translate 3D object     */
```

Figure 2: Construction of 3D arrow: 2D contour, 3D object and parts of the code to generate this icon

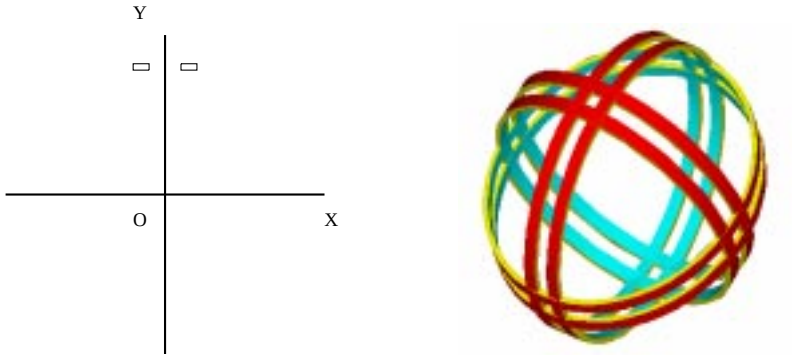


Figure 3: Construction of a 3D ellipsoid

senting a connected region in a data set is to construct an enclosing surface of this region, which amounts to computing an isosurface. This isosurface will have an irregular shape, of which the details are not always relevant to understanding the data. The ellipsoid icon can also be used to show a more simplified or abstract view of a connected region. The ellipsoid can be computed using just the geometric information (position of extracted nodes, or the boundary nodes) or with the density information. The tensor of second moments of the region defines an approximate spatial distribution of the grid points in the region. As before, the eigenvalues and eigenvectors of this tensor can be used for the ellipsoid parameters. Examples are presented in Section 8 and in [22]. Higher order moments can determine the 'goodness' of the ellipsoid fit for classification and pattern recognition. In addition, higher order moments can define additional parameters with which to 'deform' the ellipse for a better fitting icon.

To construct an ellipse, the contour shown at left in Figure 3 is rotation-swept around the x-axis. This geometry is rotated around the y and z axes to get three double bands. The object is then scaled by l_1, l_2 and l_3, rotated by (α, β, γ) and translated by (p_x, p_y, p_z). The shape of an ellipsoid can be visually represented in many ways. Figure 4 shows several variants that have been constructed with the modeling language.

7 Multilevel data abstraction

The process of feature extraction and attribute calculation in iconic visualization is a process of data abstraction since the attribute set represents the data at a higher, more abstract level. The abstraction process can be applied recursively, by using the attributes calculated at the previous stage as input for the next, e.g. for calculation of derived quantities of higher order (such as moments, or gradients). This can result in a series of attribute sets, each representing the data at a different level of abstraction. The data at each level can be visualized with a corresponding icon. The abstraction level of the symbolic representation is likely to increase if the level of abstraction of the data increases; a more abstract data representation gives a more abstract visual representation. Also, the semantics of an attribute set can be enriched in the process of abstraction.

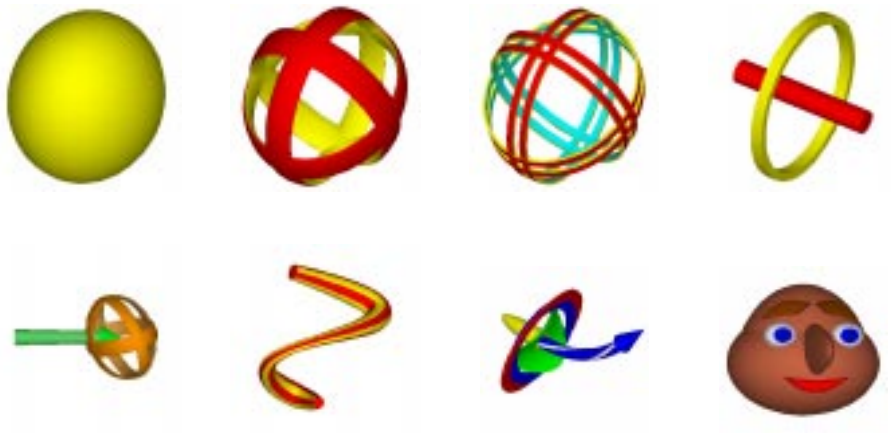


Figure 4: Example icons: four ellipsoid icons, an average velocity arrow with velocity distribution ellipsoid, an interpolated tube through five positions, a velocity gradient probe, and an 18 parameter 3D Chernoff face.

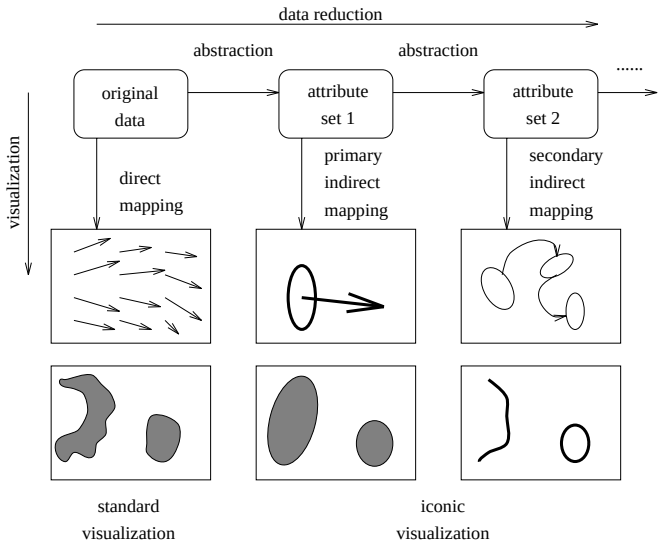


Figure 5: Multilevel feature extraction.

This is depicted in Figure 5. The upper line shows the multilevel abstraction process, of which data reduction is an important side-effect. At each abstraction level, the attribute sets can be mapped onto symbolic representations (icons), yielding an iconic visualization of the data. Visualization mapping without a distinct abstraction is called direct mapping (e.g. arrow plot, isosurface). Mappings with one or more levels of abstraction are called (primary, secondary, ...) indirect mappings. The lower pictures show examples of visualizations that correspond to the data abstraction process.

In the abstraction process, often some kind of simplification is applied by removing details that are considered irrelevant. In many cases, the precise geometric shape of a curve or surface is not of prime importance. A curve defined by a series of line segments (such as a streamline), or a surface defined by a maze of polygons (such as an isosurface) can be simplified by reduction of the number of line segments or polygons.

Abstract representation of connected regions include simple shape fitting and skeletal representations. An ellipsoid is an example of simple shape fitting. Cylinders, spheres, boxes, and cones are other shapes which may be useful for this purpose. The skeleton or medial axis provides a good abstraction for cylindrical or finger-like regions. The skeletons can show the topological structure of a network, such as the connectivity of blood vessels. Depending upon the type of dataset, the medial axis can be computed by thinning or tracking. In a thinning algorithm, distance from the boundary is used to remove all the points within a region until just a "core" set of points is left. In a numerical tracking algorithm, a local search is performed starting from a seed point, such as a local maximum. The search continues along in the particular direction (given by a gradient or other vector field) until the end of the region [2, 25].

The attribute sets generally contain less data than the original dataset, since only *interesting* regions are extracted. Therefore, the attribute calculation provides a means for data reduction and data compaction (See also Section 3). At each level of data abstraction, further data reduction occurs. When the computed attributes are mapped to icons, a reduced visual representation also results. This is especially useful for indexing images or viewing many images together on one page (postage/button size). Instead of globally reducing a visual representation of a dataset, one can abstract the essential information and then reduce that. This provides a more coherent method to automatically downsize visualizations and create better small images for linked documents. In Figure 6, postage-size images are created from the visualization. The original image is 128^3 scalar dataset of vorticity magnitude. The regions are isosurfaces of 38% of the maximum, colored by the interior local maxima of each region. There are 19 separate regions. For the simplified image, regions above a certain volume are fit to ellipsoids to remove the "clutter" as the figure is downsized. This is done during the segmentation with rules to specify a volume threshold. While the original dataset cannot be recreated from the iconified representation, quantifiable information, such as volume, mass, and the number of distinct regions is still available.

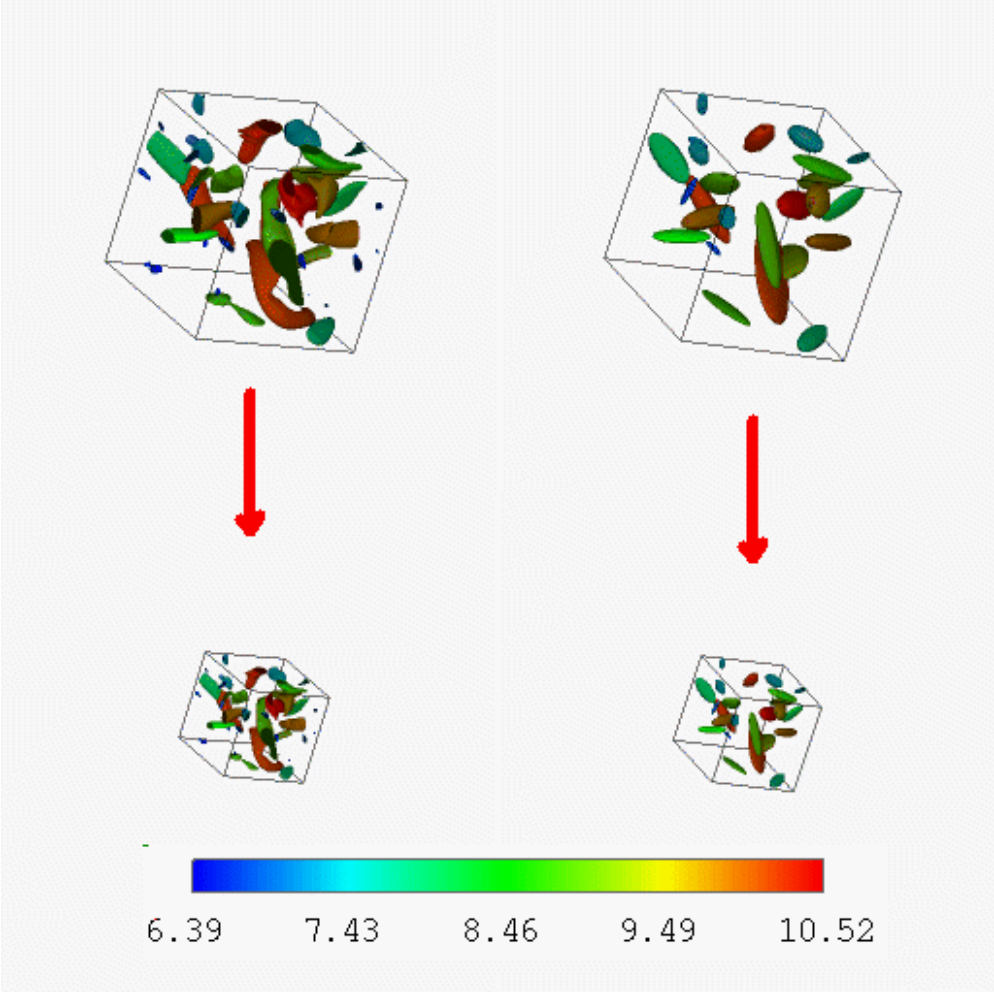


Figure 6: Downsizing images: regions are colored by their interior local maxima

8 Implementation and examples

The techniques described in the previous sections have been implemented as modules in AVS [1]. There are modules for region selection, attribute set calculation (by volume integration, streamline integration or interpolation), for further abstraction of the attribute sets (e.g. by ellipsoid fitting), and for design and generation of the icons. Most visualizations in this section have been created using these modules.

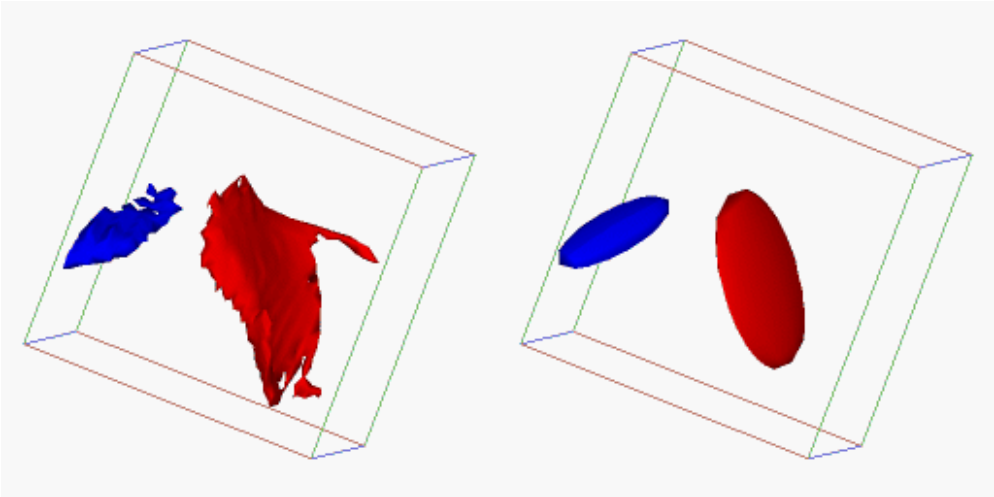


Figure 7: Ellipsoid fitting around regions of large vorticity magnitude.

Figure 7 is a 32^3 subdomain of a 512^3 turbulence simulation. The variable being visualized is vorticity magnitude. The isosurfaces at left are segmentations resulting from a threshold of 35% of the maximum. Each region is partitioned and an ellipsoid is fit to the region by calculating the weighted second moments (see Section 5) and normalizing the ellipsoid (eigenvalues) by the volume of the extracted region. In the left image, the isosurfaces are shown. The right image shows the ellipsoids. The ellipsoids demonstrate a reduction in complexity as compared to the isosurfaces in addition to providing meaningful information in a simple format. While the ellipsoids do not contain all of the information in the orginal regions, certain information is maintained, such as volume, general position, and basic shape. Note that the aspect ratios of the ellipsoids are clearly evident, enabling the scientist to identify flat or 'sheet-like' structures easily.

The extracted regions contain a total of 608 nodes, and the bounding surfaces contain 3010 triangles. Another example is given in Figure 6, which is a 128^3 dataset. There are 44 extracted regions with a total of 179501 nodes. The local maxima attribute is used to determine the color of the regions.

Another example of attribute mapping to ellipsoids is given in Figures 8 and 9, where a turbulent atmospheric flow is visualized. The data comes from a numerical simulation of a chemical reaction in an atmospheric flow. Air velocity, temperature, and concentrations of O_3, NO, and NO_2 are given on a 42^3 Cartesian grid. The goal of the visualization is to show the velocity distribution in areas of high reaction speed. In these areas, NO_2 is formed, and the researchers wanted to know whether NO_2 would flow downwards to the earth, or move upwards and disperse in the atmosphere.

Both images were created to answer the above question. For both, regions with a reaction speed higher than 50% of the global maximum were selected. In Figure 8, a low abstraction level visualization was created by showing arrows at all grid nodes that belong to a region with high reaction speed. The resulting image thus displays the velocity in these regions. In Figure 9, the iconic approach is shown. The centroids and second moments of the regions with high velocity were calculated and mapped to ellipsoids (the yellow ones). For these regions the air velocity distributions were also calculated: the mean and the variance/covariance matrix. The arrow icons show the mean velocity for each region, and the ellipsoids at the heads of the arrows show the velocity variances.

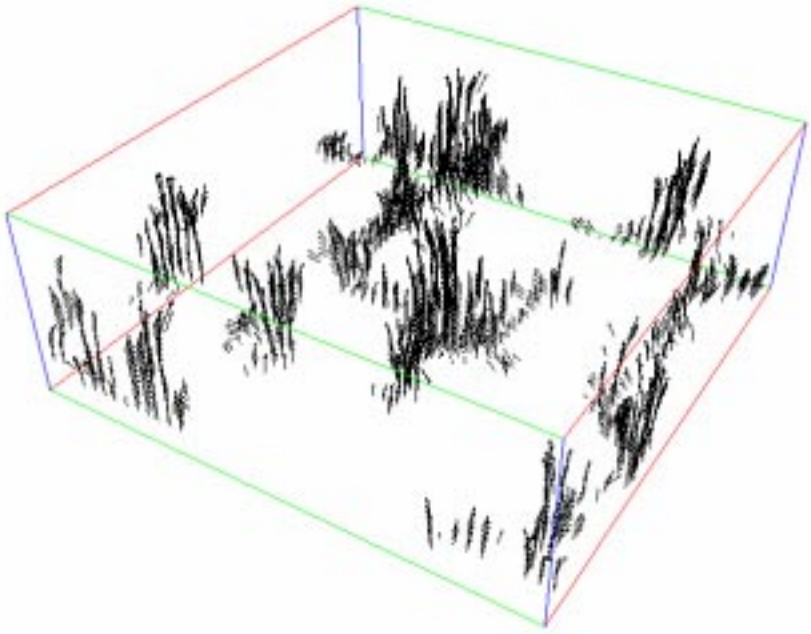


Figure 8: Velocity in regions with high reaction speed, visualized using arrows at grid nodes where the reaction speed is high.

The two approaches clearly represent two levels of data abstraction: the arrows are a direct mapping, the icons are an indirect mapping. The first visualization is intuitive, but the image is rather cluttered, and it is difficult to see the general flow direction. For the second image, statistical attributes have been calculated for each selected region, and ellipsoid icons are used to show the distributions. The image contains all information that is needed, yet a much smaller number of objects is shown. To obtain this result, the data is visualized in a more abstract, but less intuitive way.

In Figure 10, a steady, laminar flow in a backward facing step geometry is visualized. Velocity and pressure data are defined on a $25 \times 37 \times 9$ curvilinear grid. For this visualization, the features of interest were the regions with the spiraling flow pattern behind the step. These regions were extracted by thresholding the normalized helicity density at 66% of the global maximum. The attributes calculated for these regions were the centroids and second moments, which were used for fitting ellipsoid icons. Two streamlines were generated through starting points in these regions, and visualized

using tubular icons. The streamlines show the characteristic spiraling pattern. Each tube icon is a generalized cylinder, of which the axis is defined by two consecutive points on the streamline, and the two direction vectors at these points. The radius of the circular cross section at the end points is bound to the inverse of the square root of velocity magnitude. In this way, a smooth continuous tube is generated, which is an approximation of a constant-flux stream tube; the velocity magnitude can be inferred from the tube diameter. The local pressure is bound to an icon parameter that determines the tube's color.

The number of positions along the streamline was reduced prior to mapping the positions to icons. It appeared that, although many positions along the streamline must be generated in the streamline integration process, only few of them are necessary to adequately visualize the essential shape of the streamline. The combined visualization of the regions of high helicity density and the stream tubes gives a summarized image of this flow.

In Figure 11, we show the application of iconic feature visualization techniques for another CFD dataset. The data is from a Navier-Stokes simulation of a hypersonic flow of Mach 5 along a blunt fin and a wedge configuration. Velocity, vorticity, pressure, temperature and Mach number are defined on a curvilinear grid of $101 \times 56 \times 81$ nodes. Several features appear in this dataset, such as vortices and shock waves [16]. In our visualization, we focus on the vortices, as they are important for the evaluation of the flow field. Two vortex cores were extracted using particle tracing techniques, carefully placing particle sources in high vorticity areas. Along these cores, local rotation, temperature and Mach number were calculated. All of these values are mapped onto tubular vortex icons with a star-shaped cross section. Rotation (scaled with a factor) is shown by the grooves and temperature by color. This example shows how a combination of feature extraction and iconic techniques can create simple but instructive visualizations from complex data sets.

The last example is shown in Figure 12. There are two vector datasets (one $252 \times 60 \times 60$, and the other $128 \times 30 \times 30$) of vorticity. The isosurfaces of vorticity magnitude are shown (the transparent tube-like structures) with the skeletons. The skeletons are found using a predictor-corrector scheme, similar to Banks and Singer [2]. The direction of the skeleton is determined by the vorticity vector field. A purely geometric skeleton can be generated by erosion of a 3D region, using medial axis algorithms from computer vision [20]. Other attributes can be mapped onto the skeletons.

9 Conclusions and research directions

Data sets generated from scientific computing are still rapidly increasing in size and complexity. Feature extraction and iconic visualization enable the scientist to quickly focus on important features, and and cogently assimilate the immense amount of information.

In this paper, we have presented a framework to classify existing iconification techniques and have provided the building blocks for creating new levels of icons and abstractions.

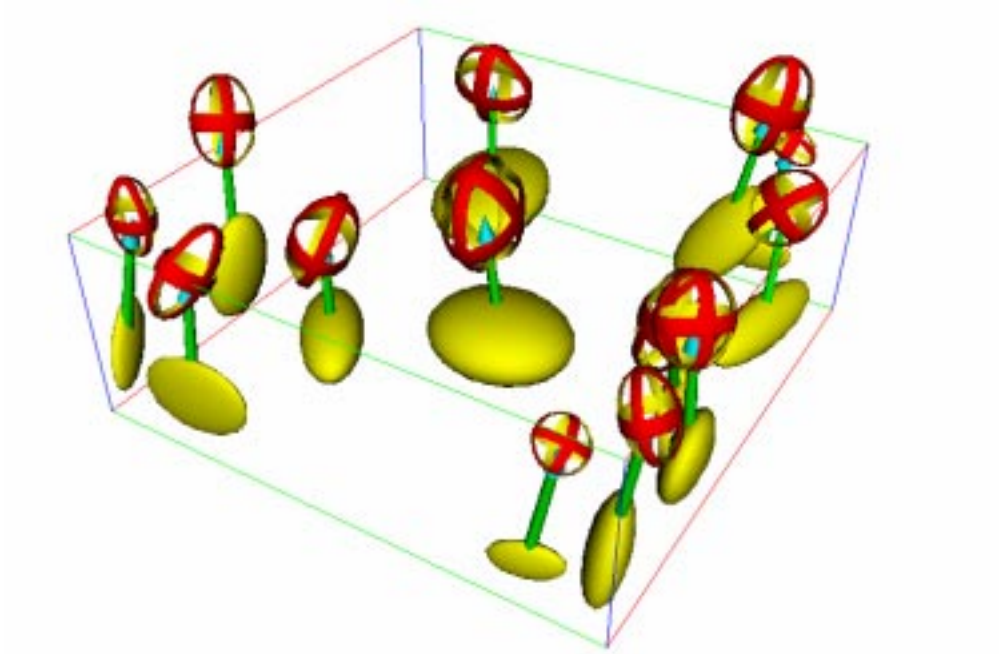


Figure 9: Regions with high reaction speed, and icons to visualize these regions (bottom ellipsoid icons), and average velocity (arrow icon) and velocity distribution (ellipsoid icon) over these regions.

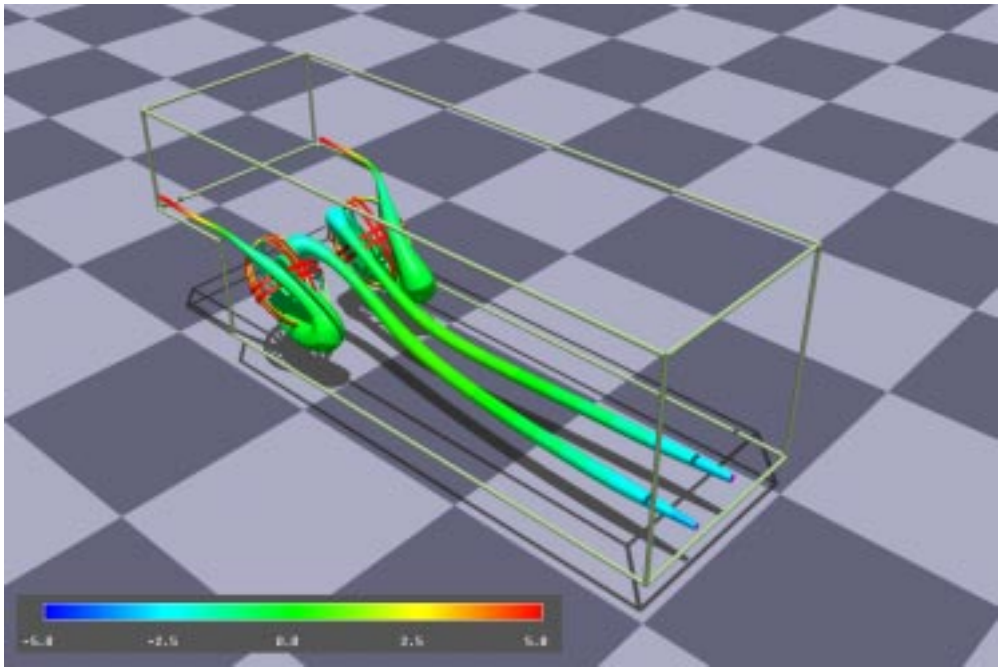


Figure 10: Regions with high normalized helicity density, and stream tubes through these regions, in a backward facing step flow.

We have showed how volume integrals and other techniques can be used to calculate attribute sets for features, and how attribute sets themselves can be the subject of additional feature extraction techniques. These techniques, combined with an icon modeling language and a mapping mechanism to map attributes to icon parameters, are useful for visualizing features.

Most important is the notion of a multi-level process of data abstraction, resulting in representations at ever higher levels of abstraction and corresponding iconic visualizations. We believe that this process allows the concepts from an application to be used in visualization. As a side effect, the data abstraction also results in a vast data reduction to aid both memory allocation and visual clutter.

The area of feature extraction and iconic visualization is still full of interesting research opportunities. Future efforts in this area must be directed towards new feature extraction criteria and algorithms, use of icons in interactive exploration and interactive steering of simulations, new icon designs and interactive design facilities. An interesting open issue is the semantic relationship between the icons and their physical features. This is based upon the cognitive process of attaching meaning to symbolic objects.

Acknowledgments

The flow dataset of the backward facing step (Figure 10) is provided by Guus Segal (Department of Technical Mathematics, Delft University of Technology) and the atmospheric flow dataset (Figure 8) by Ivo Bouwmans and John Meeder (Department of Fluid Dynamics, Faculty of Mechanical Engineering, Delft University of Technology). The vortex core data for the blunt fin and wedge flow were kindly provided by H-G. Pagendarm, German Aerospace Research Establishment in Göttingen. We thank Ari Sadarjoen of Delft University for his help in producing the image of figure 11. The turbulent datasets were provided from the Laboratory for Visiometrics and Modeling, CAIP Center, Rutgers University, courtesy of N. Zabusky, V. Fernandez, X. Wang and N. Gagvani. The laboratory is supported by NASA NAG2-829, DOE (DE-FG02-93ER25179), and ARPA HPCD.

References

[1] Advanced Visual Systems Inc. *AVS User's Guide, Release 4*, May 1992.

[2] D C Banks and B A Singer. Vortex visualization of an unsteady flow. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):151–163, 1995.

[3] W F Bronsvoort, P R van Nieuwenhuizen, and F H Post. Display of profiled sweep objects. *The Visual Computer*, 5(3):147–157, June 1989.

[4] J M Chambers, W S Cleveland, and P A Tukey. *Graphical Methods for Data Analysis*. Wadsworth International Group, Wadsworth International Group, 1983.

[5] S-K Chang. *Principles of Pictoral Information Systems Design*. Prentice-Hall, Englewood Cliffs, 1989.

[6] H Chernoff. Using faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68(342):366–368, 1973.

[7] R Ellson and D Cox. Visualization of injection molding. *Simulation*, 51(5):184–188, 1988.

[8] R Erbacher and G Grinstein. Issues in the development of 3D icons. In M Göbel, H Müller, and B Urban, editors, *Visualization in Scienti c Computing*, pages 109–123. Springer Verlag, Wien, 1995.

[9] J K Feibleman. *An introduction to the Philosophy of Charles S Pierce*. MIT Press, Cambridge, 1969.

[10] R B Haber and D A McNabb. Visualization idioms: A conceptual model for scientific visualization systems. In G M Nielson, B D Shriver, and L Rosenblum, editors, *Visualization in Scienti c Computing*, pages 74–92. IEEE Computer Society Press, Los Alamitos, California, 1990.

[11] J L Helman and L Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, 1991.

[12] L Hesselink and T Delmarcelle. Visualization of vector and tensor data sets. In L. Rosenblum et al., editor, *Scienti c Visualization: advances and challenges*, pages 419–433. Academic Press, London, 1994.

[13] W C de Leeuw and J J van Wijk. A probe for local flow field visualization. In G M Nielson and D Bergeron, editors, *Proceedings Visualization '93*, pages 39–45, Los Alamitos, 1993. IEEE Computer Society Press.

[14] J.D. Mulder and J.J. van Wijk. 3D computational steering with parameterized geometric objects. In G M Nielson and D Silver, editors, *Proceedings Visualization '95*, pages 304–311, Los Alamitos, CA, 1995. IEEE Computer Society Press.

[15] *Concise Oxford Dictionary, 8th Ed.* Oxford University Press, 1991.

[16] H-G Pagendarm and B Walter. Competent, compact, comparative visualization of a vortical flow field. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):142–150, 1995.

[17] F H Post and J J van Wijk. Visual representation of vector fields. In L Rosenblum et al., editor, *Scienti c Visualization: advances and challenges*, chapter 23, pages 367–390. Academic Press, London, 1994.

[18] F J Post, T van Walsum, F H Post, and D Silver. Iconic techniques for feature visualization. In G M Nielson and D Silver, editors, *Proceedings Visualization '95*, pages 288–295, Los Alamitos, CA, 1995. IEEE Computer Society Press.

[19] W Ribarsky, E Ayers, J Eble, and S Mukherja. Glyphmaker: creating customized visualizations of complex data. *IEEE Computer*, 27(7):57–64, jul 1994.

[20] J Serra. *Image Analysis and Mathematical Morphology*. Academic Press, New York, 1988.

[21] D. Silver. Object-oriented visualization. *IEEE Computer Graphics and Applications*, 15(3):54–63, may 1995.

[22] D Silver, N Zabusky, V Fernandez, M Gao, and R Samtaney. Ellipsoidal quantification of evolving phenomena. In N M Patrikalakis, editor, *Scienti c Visualization of Natural Phenomena*, pages 573–588. Springer Verlag, Tokyo, 1991.

[23] D Silver and N J Zabusky. Quantifying visualizations for reduced modeling in nonlinear science: Extracting structures from data sets. *Journal of Visual Communication and Image Representation*, 4(1):46–61, March 1993.

[24] S Smith, G Grinstein, and R D Bergeron. Interactive data exploration with a supercomputer. In G M Nielson and L Rosenblum, editors, *Proceedings Visualization '91*, Los Alamitos, 1991. IEEE Computer Society Press.

[25] J Villasenor and A Vincent. An algorithm for space recognition and time tracking of vorticity tubes in turbulence. *CVGIP: Image Understanding*, 55(1):27–35, March 1992.

[26] T van Walsum. *Selective visualization on curvilinear grids*. PhD thesis, Delft University of Technology, The Netherlands, 1995.

[27] T van Walsum and F H Post. Selective visualization of vector fields. *Computer Graphics Forum*, 13(3):C339–C347, September 12–16 1994.

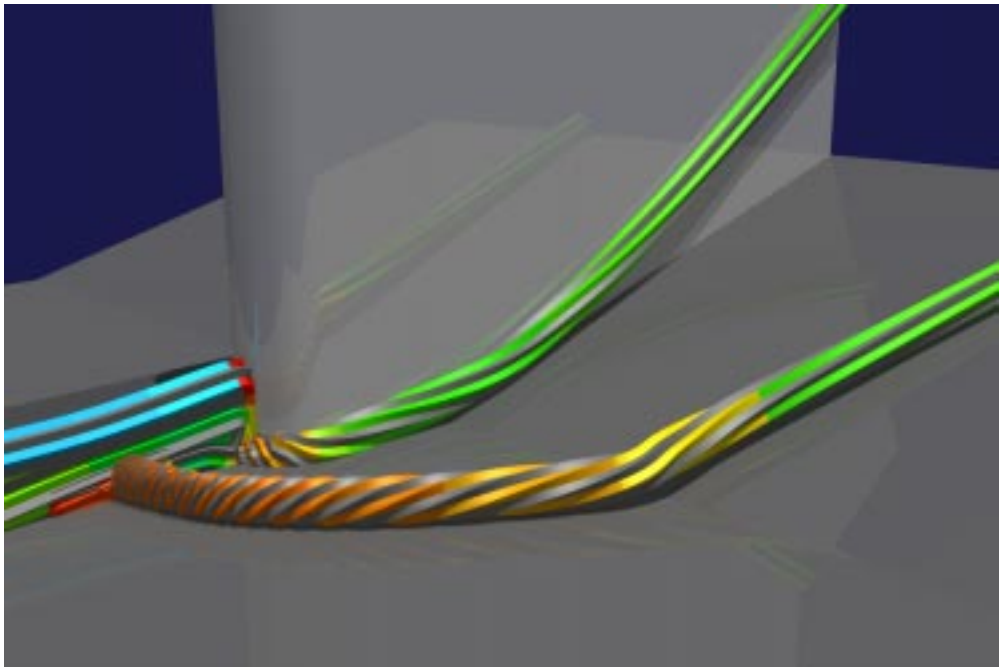


Figure 11: Vortex cores in a supersonic flow around a fin. The tubular icons show local rotation (grooves) and temperature (color).

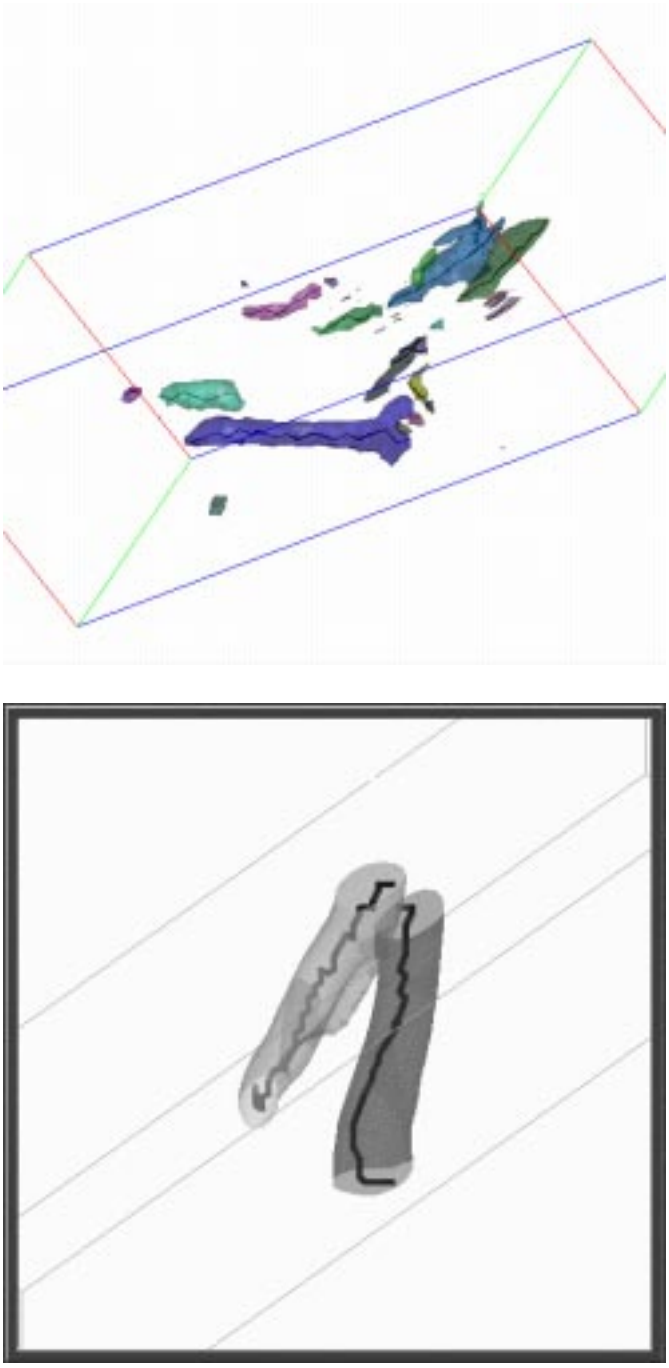


Figure 12: Vortex cores reduced to a skeleton representation.