Computers & Graphics 34 (2010) 719–728

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Example-based interactive illustration of multi-field datasets

Stef Busking *, Charl P. Botha, Frits H. Post

Data Visualization Group, Delft University of Technology, Mekelweg 4, Delft, The Netherlands

ARTICLE INFO

Keywords:
Multi-field visualization
Illustrative visualization
Interactive exploration

ABSTRACT

Multi-fields are widely used in areas ranging from physical simulations to medical imaging. Illustrative visualization techniques can help to effectively communicate features of interest found in a given field. Current techniques for multi-field visualization are mostly focused on showing subsets of local attributes such as single values or vector directions, e.g., using colors, texture, streamlines or glyphs. Instead, we present an approach based on highlighting areas with similar characteristics, considering all attributes of the field.

Our approach is example-based and interactive. A user simply selects a point within the field, upon which the system automatically derives the characteristic combination of attributes for that point. Our system then automatically creates a visualization highlighting areas within the field which are similar to the example point with respect to these characteristics. The visualizations are presented using sparse, illustrative techniques, using contours and colors to clearly delineate and identify separate areas. Users can interact with the visualizations in real-time, by moving the example point or, optionally, by changing the characteristics or adjusting other parameters used to determine similarity.

1. Introduction

Detection and extraction of features is an important step in the process of visualizing any dataset. Features are patterns of interest within the data. As interest in and specification of such patterns vary between applications, features are often segmented from the data using application-specific approaches.

With this work, our contribution is the following: We present a method whereby user interest in any multi-field dataset can be indicated by simply pointing somewhere in the dataset. Based on a real-time analysis of the high-dimensional data attributes at this example point and a point of contrast, the system then determines and illustrates all similar regions as feature objects. The user can interactively manipulate existing or add new feature objects, resulting in illustrations such as shown in Fig. 1. This enables an explorative approach to the specification and investigation of features in multi-field datasets.

Our techniques can be used to create an *illustration-by-exploration* approach: Rather than creating a single visualization that shows all features, we create sparse, user-guided visualizations of specific features, where an overview of the entire dataset and the context of features is obtained through exploration rather than being given in a single, potentially cluttered image. Illustrative rendering techniques are well-suited to visualize features found through such exploration, due to their sparseness and ease of interpretation.

These techniques can supplement existing visualization and segmentation techniques by offering a quick and intuitive way to explore a given dataset, and assist in locating and examining the characteristics of features in the dataset. This information can then be used by existing techniques to extract and/or quantify such features.

In the following, we first discuss work related to multi-field visualization and our proposed approach. We then present our interactive exploration approach, followed by a detailed description of the feature extraction and rendering techniques involved. We illustrate our approach with several example visualizations of extended vector and tensor fields. Finally, we present our conclusions and directions for future work.

The complete implementation of our techniques and the prototype application have been released as open source software and are available at http://multifieldexplorer.googlecode.com/.

2. Related work

Fields are functions that are defined on the spatial domain. Multi-fields or multi-field datasets simply refer to datasets consisting of two or more fields on the same spatial domain. This definition includes data consisting of any combination of multiple scalar fields, vectors and tensors. For an extensive overview on the visualization of this type of data, we refer the reader to the survey

* Corresponding author.
E-mail addresses: s.busking@tudelft.nl (S. Busking), c.p.botha@tudelft.nl (C.P. Botha), f.h.post@tudelft.nl (F.H. Post).

0097-8493/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2010.07.004

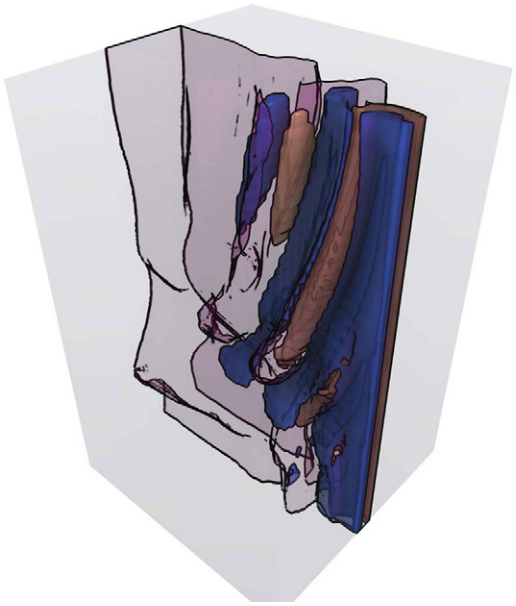

Fig. 1. Illustrative visualization, created using our approach, of a multi-field representing the flow behind a tapered cylinder, placed vertically on the right side just outside the field. The field consists of the velocity vector field and its derivatives. The illustration highlights the alternating vortices characteristic for such flow. The transparent surface shows the area where flow velocity is altered by the cylinder.

by Fuchs et al. [1]. In the remainder of this section, we give a compact summary of current work on multi-field data visualization, grouping and ordering techniques according to their level of data abstraction.

In feature-tracking, features are objects, local behaviors or other patterns of interest in a dataset, whereas in many other fields they are the dimensions of a feature space or components of an image. In order to resolve this ambiguity, we refer to the former as *feature objects*, while the components of a multi-field are referred to as *attributes*.

Current techniques for vector field visualization often focus on showing directions [2]. In tensor visualization, second order symmetric tensors can be represented by ellipsoids or superquadrics [3], or reduced to their principal directions and visualized as a vector field. Analogously, many techniques for multi-scalar visualization are also based on representing each sample point with a single visual element. For example, in multi-scalar volume rendering, the transfer function is extended to map from the multi-dimensional input space to the RGBA output space [4]. These techniques all attempt to visualize multi-field data as directly as possible.

Visualization of quantities derived from a field can often show more about the field than a visualization of only the *direct attributes* that are part of the field. For example, the inclusion of image gradients has been shown to lead to better transfer functions [4], and Busking et al. [5] visualized deformation vector fields by highlighting areas of significant change in volumes based on a measure derived from the field's Jacobian. However, different derived quantities may be required for different applications. Van Walsum et al. [6] presented a framework which allowed users to select and visualize arbitrary quantities derived from vector fields. Their system also allowed users to combine the resulting feature objects using Boolean set operations. Similar compositing techniques were applied to the visualization of multi-field and time-varying data by Woodring and Shen [7], although only direct attributes of the fields were used in their work.

A multi-field can also be seen in terms of its feature space, where each attribute is a dimension. Henze [8] presented a visualization approach using multiple views of "linked derived spaces", which are essentially projections of this feature space. Kniss et al. [4] presented a dual-domain interaction approach for exploring multi-field volume data, where interactions in the spatial domain are mapped to a linked view of the feature space. This view can then be used to manipulate transfer functions (defined on this feature space), which are in turn used for direct volume rendering of the volume data in the spatial domain. This idea was further developed by linking multiple scatter plots of the feature space with a direct volume rendering showing selected subsets of the data [9–11].

Finally, feature extraction techniques attempt to reduce complexity in the data by extracting physically meaningful patterns, hence lifting the resultant visualization to a higher level of abstraction [12]. This is also the goal of visualization techniques based on topology [13], clustering [14,15] or the extraction of physical properties, such as vorticity. Vortex extraction, the most common type of feature extraction technique for vector fields, is often based on a combination of physical and mathematical criteria [12].

The efficacy of topology techniques on vector and tensor data is highly dependent on the presence and configuration of, respectively, critical and degenerate points. Results of clustering are often too dense for illustration. Furthermore, control is limited to selecting clustering heuristics, which may be non-intuitive. Both techniques are also computationally intensive, which makes them less suitable for interactive use. Extraction of physical properties is highly domain-specific.

Ma proposed using machine learning techniques, neural networks for example, to extract meaningful physical feature objects from the multi-field data, based on user interaction [16]. This is a good attempt towards automatic generation of feature detection criteria, but is difficult to control and not directly and transparently linked to field attributes.

In this paper, we combine the direct visualization of arbitrary combinations of field attributes with a feature space approach and example-based interaction. Rather than visualizing attributes directly, we introduce the use of *local similarity* as the basis for feature selection. We use example-based, user-adjustable similarity measures computed in feature space to visualize feature objects consisting of all points within a dataset that are similar to a user-defined example. This way, rather than specifying the characteristics of feature objects directly (e.g., extracting an iso-surface of attribute X at value Y), the user can simply point at interesting areas in the data.

In our work, illustration plays an important role in visually summarizing the data. Illustrative visualization is often used to give a deliberately unbalanced view of a complex scene, by emphasizing objects of interest, de-emphasizing or suppressing other objects, and showing context only for orientation. Various techniques have been proposed to integrate such visualizations of focus+context in a single image, including cut-away views or importance-driven visualization [17]. One could also see example-based feature specification as a flexible and interactive method of specifying an importance field. For our work, however, we take advantage of the interactivity of our techniques to create a strongly selective approach, where the objects shown are only those selected by the user. Context should be provided either by interactive exploration, the addition of other user-specified objects, or by integration of our techniques with existing, possibly domain-specific techniques.

In terms of the example-based specification of feature objects, our work is related to the stroke-based transfer function specification of Ropinski et al., where the user specifies interesting

edges by indicating them directly in the rendering by means of a stroke, resulting in the automatic specification of a transfer function that emphasizes the visual appearance of all stroked edges [18]. Our approach is differentiated by catering for multi-fields and utilizing the direct calculation of similarity versus indirect application through a transfer function.

3. Interactive exploration

We aim for *interactivity*, as this enables our visualization to be used as a tool for interactive exploration as well as for creating illustrations that are effective in conveying meaningful feature objects identified during exploration. Globally, the work-flow of a user using our system to illustrate a field is as shown in Fig. 2:

1. As a *pre-processing* step, a multi-field is constructed from the input field, containing any number of attributes. For example, a vector field can be extended by adding derived quantities such as magnitude, divergence or curl. Adding such attributes allows any feature objects characterized by these attributes to be identified and visualized by our system. It also allows the system to better describe such feature objects to the user by presenting the defining characteristics in terms of these attributes.
2. *Selection*: the user selects a point of interest within the field, or a pair of points which is to be compared. The values at the first point determine the example on which the feature object is based. The second point, if given, determines values that should be outside the feature object.
3. *Abstraction*: the system automatically determines characteristic attributes for the given point or pair of points and presents these to the user.
4. *Generalization*: Based on these characteristics the system determines an appropriate similarity measure. This measure combined with the example point defines a similarity field over the dataset.
5. *Filtering*: the resulting feature object, consisting of all points in the volume that are similar to the selected point with respect to the chosen characteristics, is visualized interactively by thresholding the similarity field.
6. *Feedback*: moving the selected point of interest updates the visualization in real-time, allowing interactive exploration of the data. The user can also refine the feature object by manipulating the selection of characteristics, or by selecting from a number of visual styles to add additional semantics to the illustration.

The process can be repeated to add any number of additional feature objects to the illustration. In this way, a user can select multiple feature objects by using multiple example points, each with its own selection of characteristics and visual styles, thereby creating an illustration combining all feature objects of interest.

Section 4 describes our techniques for example-based selection of meaningful feature objects. These objects are visualized interactively using illustrative styles, which are detailed in Section 5.

4. Example-based selection of feature objects

Our illustrative exploration approach is based on extracting similarity-based feature objects from the data. These objects are defined as those areas which are similar to a user-defined point with respect to a combination of user-defined characteristics. We use a similarity measure defined on the points of the multi-field in combination with a pair of user-selectable thresholds to derive such areas. By adjusting the thresholds, the user can decide when points are considered "similar enough" to be included in the feature object.

Our approach is based on the notion of feature spaces. A sampling of any multi-field consisting of N variables can also be interpreted as a set of points in a *feature space* of N dimensions. Section 4.1 describes our definition of similarity in such a feature space. The methods for creating a compact feature space in which these similarity measures can be applied are described in Section 4.2. Finally, Section 4.3 describes how we use user-defined examples to derive appropriate parameters in order to enable our interactive exploration approach.

4.1. Similarity

Various similarity metrics for multi-fields or vector fields have been proposed in literature [19,20]. Most of these, however, only compute global similarity between whole fields. Furthermore, some metrics are expensive to compute.

As noted above, we base our similarity metric on the feature space representation of the multi-field. Similarity (or rather dissimilarity) is equivalent to distance within this feature space. We transform the original feature space using projection and scaling in order to provide user control and example-based refinement of the similarity measure. To keep our application simple and interactive, we use the p-norm to measure distances in this transformed space:

$$d(\mathbf{a},\mathbf{b}) = \left(\sum_{i=0}^{N} |a_i - b_i|^p\right)^{1/p}$$

Here, **a** and **b** are feature vectors consisting of the values of all attributes of the multi-field, representing the two points that are being compared, after applying the feature space transformations

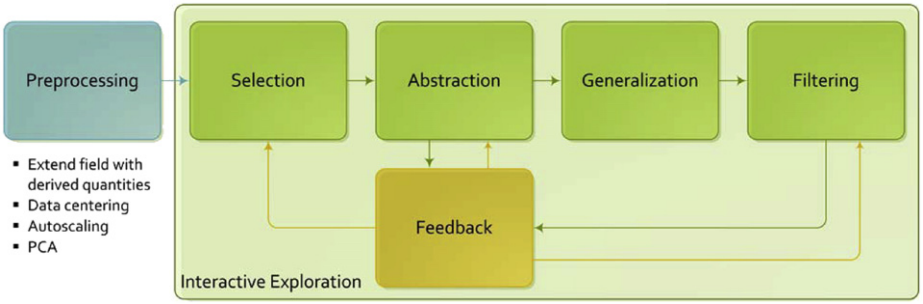

Fig. 2. The workflow in our interactive illustration system. Defining a feature object consists of selecting an example point in the field (selection). The values at this point are used to automatically select characteristics of interest (abstraction). These are then used to generate a similarity measure (generalization), from which the feature objects are extracted (filtering). The system is interactive, allowing a user to provide feedback and immediately observe the results in the visualization.

described below. The parameter $p \geq 1$, can be used to change the type of norm computed. This is generally set to 2 for the Euclidean norm.

4.1.1. Projection

Not all attributes may be of interest when determining similarity. In fact, a user may want similarity to be invariant with respect to certain attributes. If those attributes directly correspond to dimensions in the feature space it suffices to simply omit the respective values from the similarity computation. However, we reduce the dimensionality of the feature space before visualization using principal component analysis (as described in the next section). In this case attributes can be eliminated by *projection* onto a subspace containing only the remaining attributes. To achieve this, the transformation between the original and reduced feature spaces is stored during pre-processing. The projection of a feature space point is computed by transforming the point's feature vector back to the original space, setting the values corresponding to the attributes that are to be ignored to zero and transforming the result back to the reduced feature space.

4.1.2. Non-uniform scaling

Furthermore, we enhance the similarity computation by introducing a *bias vector*. The bias vector is a vector in feature space, determined automatically based on user input (see Section 4.3), along which this space can be stretched before determining similarity. The effect is that changes proportional to the linear combination of attributes specified in the vector have more influence on the value of the similarity measure than other changes, as illustrated in Fig. 3. Stretching is implemented as simple linear scaling along the bias vector with user-adjustable scaling factor s. Adjusting the scaling factor allows the similarity measure to be shifted between neutral ($s = 1$) or any amount of bias ($s > 1$). Values $s < 1$ are also allowed; the resulting similarity measure favors similarity in ways uncharacteristic of the selected example. Such measures may be used to uncover patterns normally obscured by trends in the most characteristic attributes for a point.

The scaling operation is made volume-preserving with respect to the user-selected feature subspace by pre-scaling the entire space uniformly by factor $s_u = s^{-1/(N-1)}$, then scaling along the bias vector by $s_n = s/s_u$. Here N is the number of dimensions of the original feature space remaining after projection. This way, given a uniform distribution of points in the feature space, feature objects will include approximately the same number of points regardless of the value of s. In the case where s approaches infinity, the similarity measure becomes equivalent to distance along the bias vector.

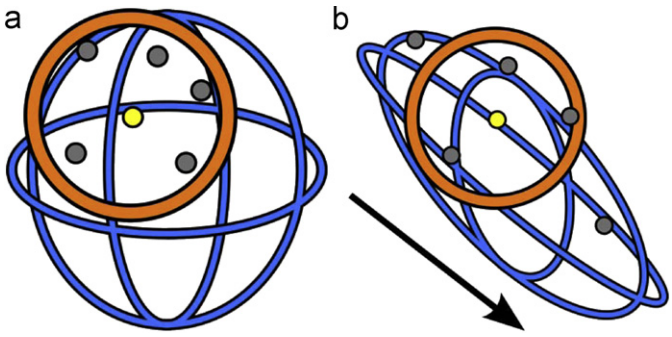

Fig. 3. Non-uniform scaling of the feature space. The orange circle represents the thresholded similarity measure, and the small circles represent the example (yellow) and other feature vectors. Stretching the feature space makes the measure more sensitive to changes along the bias vector (represented by the black arrow) and less sensitive to other changes. (a) Normal feature space, (b) biased feature space. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2. Creating the feature space

Multi-fields can potentially have large numbers of attributes, which directly affects performance and memory requirements in an interactive application. Furthermore, feature space dimensions may be heterogeneous in scale, or contain statistically dependent behavior.

In order for our techniques to find interesting relations, however, it is good to have a large number of attributes between which such relations may be found. Additionally, feedback (e.g., on user selections and bias vectors) is given and projections are defined in terms of these attributes. This means a good selection of attributes can help a domain expert to understand features found in the dataset in terms of the application domain. For this reason we pre-compute the feature space for a given multi-field in four steps:

1. Build the feature space from the points in the original multi-field. Optionally add various derived quantities as attributes of the field, such as local derivatives (e.g., gradients) or application-specific measures relevant to the user's interest.
2. Center the feature space on the origin by computing the mean value over the data for each dimension and subtracting this mean vector from each point.
3. Scale the centered feature space such that the variance over each dimension is 1 by dividing each dimension's values by their standard deviation.
4. Perform principal component analysis (PCA) on the scaled space, then transform the data to a basis of eigenvectors in order to reduce the dimensionality while maintaining as much data variance as possible.

PCA creates a basis of eigenvectors in order of how much they account for the variability in the original data. This means dimensionality can be reduced by selecting only a subset of the first N eigenvectors. However, if the original feature space is heterogeneous in nature, the PCA may favor larger scale dimensions. To alleviate this, we use *autoscaling* (i.e., division by the standard deviation for each dimension) to pre-scale the feature space dimensions, such that PCA is based on correlation rather than covariance. Pre-treatment techniques such as autoscaling can strongly influence the results of the PCA. The selection of an appropriate technique is therefore highly application-dependent. Van den Berg et al. [21] give a good overview of autoscaling and alternative techniques, and their effects on the results of the PCA.

PCA is a commonly used technique in pattern recognition for the purposes of reduction and decorrelation of high-dimensional data. While our experiments (see Section 6.2) have shown that the resulting feature spaces enable good selections of features within the data, alternative approaches exist as well. Heimann and Meinzer [22] give a good overview of alternative techniques for dimensional reduction in the context of statistical shape models, which could also be adapted to this application.

4.3. Example-based similarity

Feature objects are extracted by thresholding the similarity field created by comparing each point in the data with the values at the user-defined example location. We use the bias vector to steer the similarity measure to not only consider the values at this

location, but also their relative proportions. The bias vector is automatically determined based on the *example vector* of values at that location and a second *contrast vector*. We provide two options for selecting the second vector:

- The contrast vector can be set to the mean feature vector computed over the entire field. This way, deviations from the mean are emphasized as characteristic attributes for any user-selected point.
- The contrast vector can be sampled at a user-defined location. For instance, by selecting a location outside of the objects of interest in the field, characteristic attributes can be determined based on comparison to background values. Alternatively, such a selection can be used to emphasize the differences in characteristics between different feature objects in the field by placing example and contrast points in the objects to be compared.

While the resulting bias vector is normalized before being used to stretch the feature space, its original length is used (together with the stretch factor) to adjust the user-defined thresholds. This way, a threshold of 1 will lead to the feature object including all points with similarity values up to the difference between the two points that were selected.

In summary, given an arbitrary point $\vec{p}$, example feature vector $\mathbf{x}$ (sampled at a user-defined location) and contrast feature vector $\mathbf{c}$ (which is either the mean of the data or the feature vector at a second user-defined position), the corresponding feature object is extracted as follows:

1. Sample the multi-field to obtain the feature vector $\mathbf{p}$ corresponding to $\vec{p}$.
2. Project feature vectors $\mathbf{p}$, $\mathbf{x}$ and $\mathbf{c}$ to a subspace which eliminates a user-defined subset of "irrelevant" attributes.
3. Transform the projected feature vectors $\mathbf{p}'$ and $\mathbf{x}'$ to the stretched space given by the bias vector $\mathbf{v} = \mathbf{x} - \mathbf{c}$ and a user-defined bias factor s.
4. Compute the p-norm distance between the resulting vectors $\mathbf{p}''$ and $\mathbf{x}''$.
5. Compare the resulting value to the user-defined thresholds to classify point $\vec{p}$ as being inside or outside of the feature object.

5. Illustrative visualization

Our objects of interest are areas similar to the user-defined examples. We visualize each of these feature objects using an iso-surface of the similarity measure at a user-defined threshold. As we aim for our visualization to be used in interactive exploration scenarios, we use a sparse visualization style which is easy to comprehend, but provides enough detail to help the user gain insight into the data. Our style is inspired by schematic drawings as used in biology and engineering, which make heavy use of contours and simple textures to delineate and annotate feature objects in an image.

In order to enable interactive exploration and to avoid the complexities of surface extraction, we render the iso-surfaces directly using a ray casting approach. Using current generation GPU hardware, ray casting approaches can be implemented with real-time performance.

We use a GPU ray casting algorithm based on the technique introduced by Krüger and Westermann [23]. Rendering involves using a rasterized bounding volume to determine ray entry and exit points. These are then used to trace each ray in parallel using a fragment shader, stepping through the multi-field in fixed intervals. At each point along a ray, we sample from volumes containing the pre-processed field values to obtain the feature vector for that point. The similarity measures for all feature objects are then evaluated as described in Section 4 in order to find intersections with the feature objects' surfaces. If multiple surface intersections are found, the positions of these intersections are refined and then sorted. The front-most intersection determines the surface visualized for that ray.

We use a deferred shading approach [24] in order to create our illustrative visualization style. Rather than performing surface shading directly during ray casting, our ray casting pass outputs surface information to a G-buffer, including feature object number, surface normal and the position of the ray–surface intersection. The deferred shading pass uses only the information in this buffer to shade the corresponding pixels. Furthermore, in order to support the use of transparent surfaces, we use depth-peeling [25] in order to apply the deferred shading enhancements to each of the surfaces encountered along the rays. The results of each deferred shading pass are composited to create the final image.

The deferred shading pass is used to create the illustrative style of our visualizations. The use of transparent surfaces can help show the interior structure of feature objects, but multiple nested transparent surfaces are often hard to interpret. To alleviate this, object contours are enhanced using image-based detection of silhouettes, surface intersections and sharp edges. This enables the viewer to easily distinguish the shapes of feature objects even if transparent surfaces are used. Fully transparent surfaces (showing only contours) can be used as well, and may be helpful for adding contextual information to a visualization.

By default, simple colors are used to identify the feature objects in the image. Additionally, users can select from a number of screen-space texture patterns to be overlaid on the feature object surfaces. These textures can be used as a form of annotation, for example, a pattern consisting of "plus" glyphs can be used to indicate expansion or growth, while "minus" glyphs can indicate decreases in local volume.

Furthermore, we apply screen-space ambient occlusion (SSAO) to simulate global illumination (Ritschel et al. give a good overview of related work in this area [26]). It has been argued [27] that correct global illumination improves perception of surface shape and depth in visualizations, and helps the user better understand the 3D positions of feature objects relative to each other. As it is straight-forward to integrate existing SSAO techniques in a deferred shading pipeline, their description falls outside the scope of this paper.

6. Results

We implemented our techniques in C++, using OpenGL and GLSL for all GPU-based algorithms. A user interface was created (see Fig. 4) which, in addition to our visualization of the feature objects, includes several controls to manipulate the parameters of each feature object.

A 2D slice view of the field allows users to specify example points within the field. The example position is updated interactively by simply hovering over the slice viewer, allowing for easy exploration of the field. The values for the field's attributes at the selected point are visualized using a set of colored rectangles, where values are mapped to colors relative to the mean value over the field. In case the field has been reduced in dimensionality (such as described in Section 4.2), the stored transformation is used to convert values back to the original feature space before visualization. The normalized bias vector resulting from the user's selection is visualized in a similar way. Clicking the individual rectangles in this view allows the user to

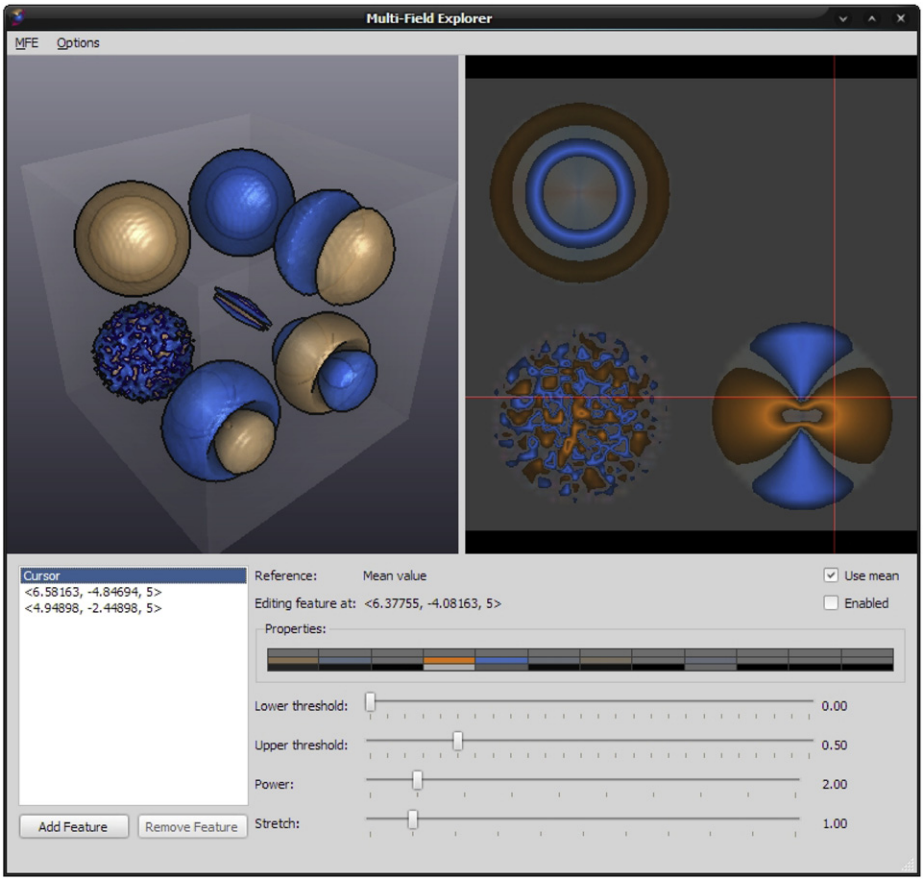

Fig. 4. The prototype application visualizing a synthetic dataset in real-time ($64 \times 64 \times 64$ points). The interface consists of our visualization (top left), a slice viewer for selecting example points (top right), a list of all current feature objects (bottom left), colored rectangles representing the values in the contrast vector, the values at the current point and the resulting bias vector (middle) and a set of sliders for manipulating the thresholds and bias factor for the current feature object. In this case, two feature objects have been defined with different example points.

further manipulate the definition of similarity by selecting the corresponding attributes to be ignored for this feature object's similarity measure.

6.1. Performance

As our techniques are aimed at interactive exploration, we measured their performance using several datasets. Using a current generation GPU (NVIDIA GeForce GTX 280), performance is generally suitable for interactive exploration. As our techniques are based on ray casting, performance is strongly dependent on the resolution of the vector field under consideration. Using a dataset of $512 \times 512 \times 80$ (Fig. 6(b)), average frame rates were around 12 frames per second when rendering at 600×600 resolution. Image resolution does not influence performance as strongly, as modern GPUs are highly parallelized. Performance and data sizes can likely be improved by integrating existing optimization and large-data-handling techniques from volume rendering literature.

As discussed in Section 5, we used depth-peeling to enable transparent surfaces to be used. This adds a separate shading pass for each layer in the scene, as well as requiring ray casting to proceed beyond the first surface hit, with both leading to decreased performance. With up to 6 layers of transparent feature object surfaces, performance of the same dataset was around 7 fps, which is still suitable for interactive use. If no transparent surfaces are used, depth-peeling is not required.

Our current implementation is limited to data sets which fit in GPU memory. This means both the dataset size in voxels as well as the number of attributes of the (PCA-reduced) multi-fields are limited. However, it would be straightforward to adapt current work on volume rendering for large datasets (e.g., Kniss et al. [28]) to our approach, in order to enable larger volumes to be explored interactively. Similarly, our implementation of the pre-processing stage currently uses an in-core approach. Processing of the largest dataset used in the paper took around 20 min and up to 7 GB of RAM on a recent machine. However, unless the user needs to refine the selection of attributes, this only needs to be done once per dataset.

6.2. Examples

We visualized several example datasets in order to demonstrate our approach. These datasets from different application domains contain well-known features, which are typically extracted and shown using application-specific techniques. This section shows how our techniques can be used to quickly find and visualize such features in an application-independent manner, thereby demonstrating the flexibility of our approach and validating its usefulness as a first step in exploring multi-field datasets.

Our first example is a computational fluid dynamics dataset representing the flow behind a tapered cylinder. The full dataset is time-dependent and can be obtained online from NASA. As our techniques do not yet handle time-dependent data, we used

a single frame from the dataset. Additionally, we cropped and re-sampled the (originally $64 \times 64 \times 32$ curvilinear) dataset to a regular grid of $128 \times 128 \times 204$ points. We extended this vector field dataset into a multi-field during pre-processing by adding a simple set of derived attributes: In addition to the vector data, we added the normalized vector, the vector magnitude, and several vector calculus derivatives of the field. The derivatives were all computed from the Jacobian of the field: the trace of the Jacobian matrix is the divergence of the field, while its determinant indicates changes in volume. We also included curl, a vector-valued attribute describing local rotation. The Jacobian can be computed at a user-defined scale, which enables filtering of small-scale details such as noise.

At certain flow speeds, such as the one used in the simulation which created the dataset, a pattern of vortices forms behind the cylinder. These vortices alternate between clockwise and counter-clockwise directions. We used our techniques to illustrate this flow behavior.

By using a single feature object and simply moving the example point in the area behind the cylinder, two opposing patterns can be located. Fig. 5(a) shows one of these feature objects. The visualization of the example feature vector for this object, shown below the figure, indicates that the directional components of the field (the first six attributes) are highly characteristic of the selected example point. Masking out the non-directional attributes of the field and increasing the bias factor to further emphasize the characteristic attributes reveals more of the pattern. The visualization of the example vector also shows that directions in the second feature object, shown in Fig. 5(b) are opposite to the first.

As the yellow and blue areas indicate regions with directions opposite to each other, we can assume the vortices exist between these areas. To highlight these, we use a pair of feature objects with the similarity measure made invariant to direction, and example points set between the yellow and blue feature objects. Fig. 5(c) shows how these new feature objects (pink and green) capture a similar recurring pattern between the yellow and blue objects. The visualization of the example vector confirms that these are the vortices, as curl (represented by the rightmost three rectangles) is the characteristic attribute for the new objects. Again, values for these attributes are opposite to each other, indicating rotation in opposite directions. Fig. 5(d) shows the final visualization of the extracted feature objects, revealing the 3D structure of these vortex patterns. Fig. 1 shows another visualization of these vortices, with an added feature object representing the area where the flow velocity is altered by the presence of the cylinder.

Fig. 6 shows visualizations of a vector field resulting from the non-rigid registration of two MRI scans of a human knee. The use of such deformation fields is common in medical image analysis.

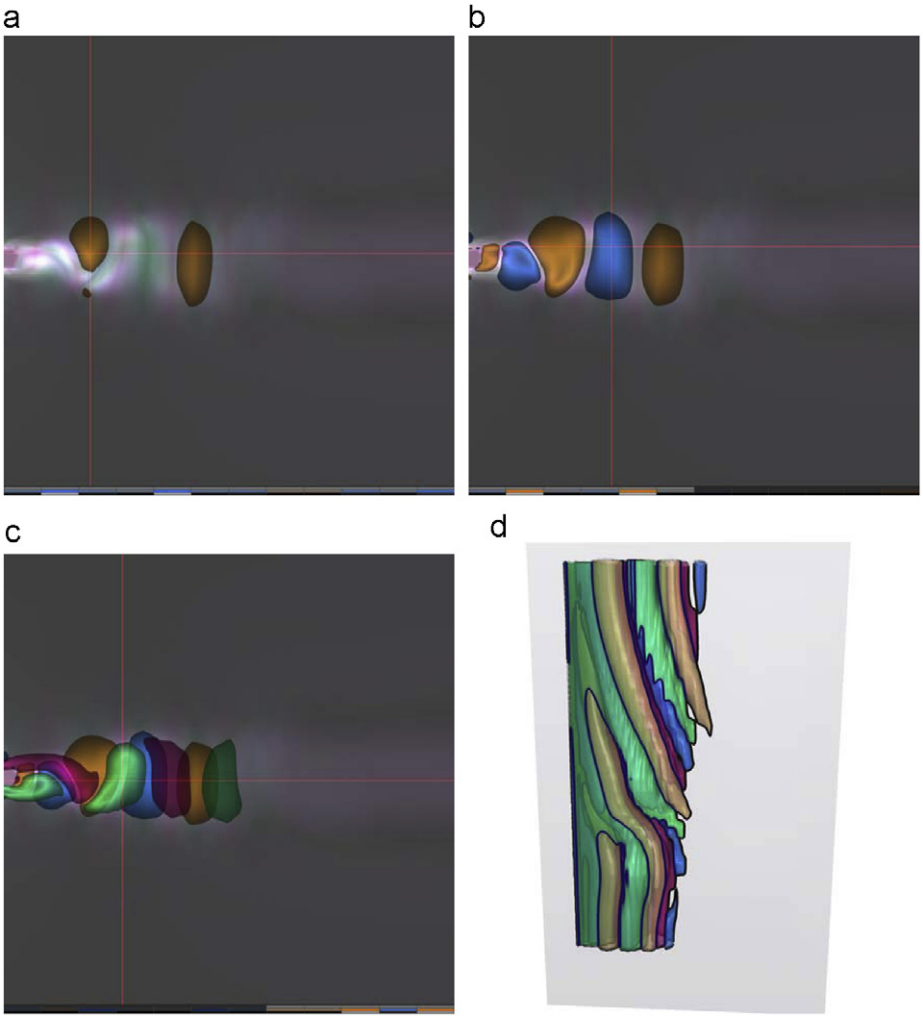

Fig. 5. Using our techniques to explore the flow behind a tapered cylinder. Slice views show steps in the exploration process. The crosshairs indicate example positions, the colored rectangles below each image visualize (from top to bottom) the contrast and example feature vectors as well as the bias vector for the corresponding feature object. (a) Single feature object, (b) opposing flow directions, (c) vortices, (d) 3D visualization.

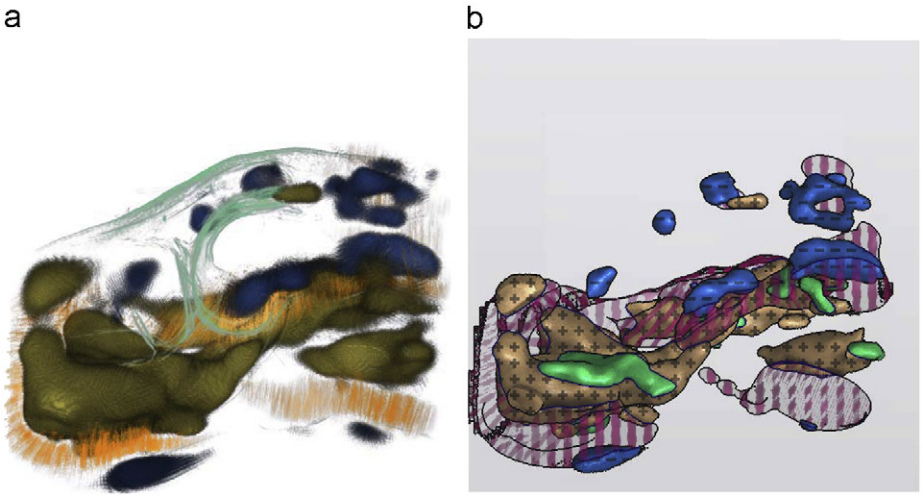

Fig. 6. Visualizations of a deformation field ($512 \times 512 \times 80$ points) representing changes between two MRI scans of a human knee. (a) Direct visualization of deformation [5], (b) similarity-based feature objects.

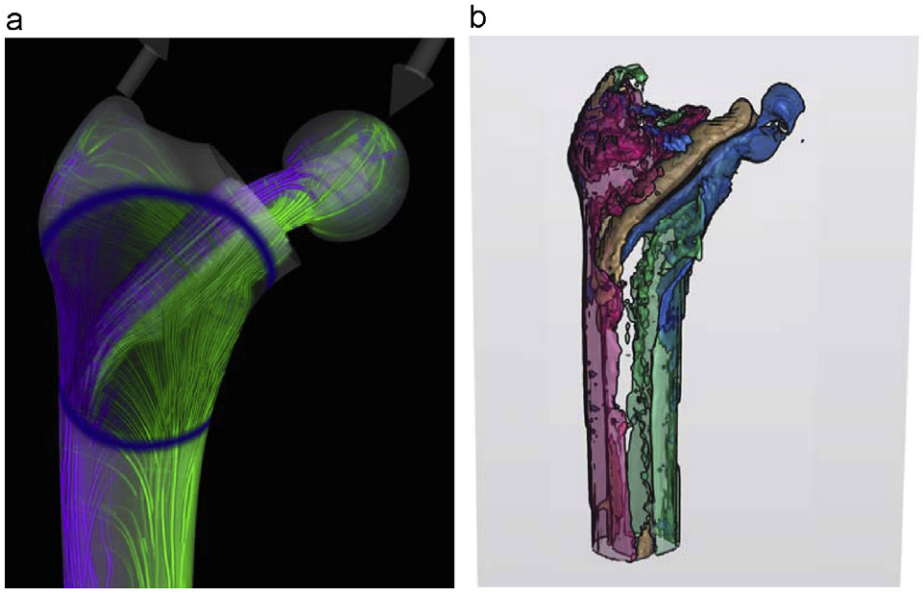

Fig. 7. Visualizations of a dataset ($85 \times 79 \times 101$ points) consisting of simulated stress and strain tensors in a human femur with a hip joint replacement implant. (a) Stress tensor streamlines [29], (b) similarity-based feature objects.

Busking et al. [5] created direct visualizations of such fields by visualizing a growth measure derived from the Jacobian determinant of the field (Fig. 6(a)). A similarity measure based on a subspace containing only this growth attribute can capture the same feature objects (Fig. 6(b)). Furthermore, by exploring the data using feature objects with different feature-subspaces, areas were found with low magnitude but a significant increase in volume. These were highlighted in green using a feature object based on both magnitude and growth.

Missing from our current implementation is a visualization of relevant context information. However, the techniques could easily be combined with a visualization of contours such as used in the earlier work mentioned above (shown in green in Fig. 6(a)). Context information could also be visualized with our feature objects if the relevant data is included in the multi-field.

Fig. 7 shows a visualization of a dataset used for hip joint replacement planning. This dataset consists of two tensor fields representing simulated stress and strain resulting from pressure being applied to the implant and femur bone. Dick et al. [29] recently presented techniques for visualizing such fields interactively with streamlines following the first principal component of the stress tensor, shown in Fig. 7(a). Fig. 7(b) shows similar structures highlighted using our techniques, applied directly to the tensor field. The visualization clearly shows the opposing tension (yellow) and compression (blue) parts of the implant (shown in purple and green in Fig. 7(a)), as well as the similarly opposing effects on the sides of the bone itself (shown in pink and green).

7. Conclusions

Our contribution in this paper is a new method for the interactive visualization and exploration of multi-fields. Our technique is based on the direct visualization of similarity,

defined as distance in a feature space, where the similarity measure is automatically derived from user-specified examples.

While our prototype implementation offered the concepts presented as the *only* option for exploring a dataset, we expect similarity-based techniques for exploration and visualization to be integrated with traditional direct visualization of field attributes. This way, traditional visualizations can guide a user in locating points of interest. Our techniques then automatically highlight similar areas within the dataset, the details of which the user can further explore using different visualization techniques.

Our techniques lead to an intuitive approach to interactive exploration, as little interaction (only point selection) is required in order to explore a given field. Furthermore, the concepts and techniques presented are generic, and can be applied to the visualization of any type of multi-field.

In this work we have presented a proof of concept for a new idea, supported by several application examples demonstrating the utility of the proposed method. When these concepts have been integrated into a real visualization system for such an application, that would present a good opportunity to perform user-oriented evaluation.

7.1. Future work

While our techniques can be applied to any type of multi-field, extension to time-varying data is more involved. Feature tracking techniques could be applied to track user-defined features over time. New visual representations should be developed to visualize the evolution over time of any detected feature objects. Stompel et al. [30] presented various illustrative techniques for enhancing the visualization of features in multi-field and time-varying datasets, which could be applied to our feature objects for this purpose.

The techniques proposed could be applied to segmentation of features from a given dataset. Boolean combinations of feature objects, as used by van Walsum et al. [6] and Woodring and Shen [7], could be used to specify areas to segment. More advanced filtering techniques could also be added, including the detection of connected components to distinguish between separate parts of the same feature object. Alternative similarity metrics should also be investigated. A variety of metrics in combination with a system for combining feature objects would create a powerful system for segmentation by example.

The definition of examples could also be extended. For example, tracing techniques like streamlines are frequently used to visualize vector fields, while brushing techniques are frequently used to make selections in other example-based systems. Such selections consisting of multiple points could be used as examples in our approach by, e.g., averaging attributes over the set, or by considering similarity of points as compared to the example point closest to the point being evaluated. This way, our approach can be integrated with application-specific techniques for feature selection. For instance, in a medical brain dataset, areas with similar characteristics could be selected around fibers traced from DTI data.

Finally, we plan to integrate our approach with other visualization techniques. In addition to highlighting areas using our current rendering techniques, similarity-based feature objects could also be used to indicate importance or user interest in an importance-based visualization approach.

Acknowledgements

We are grateful to Dipl.-Inf C. Dick of the Computer Graphics and Visualization group, Technische Universität München, for providing the hip joint replacement datasets. This research is supported by the Netherlands Organization for Scientific Research (NWO), project number 643.100.503 "Multi-Field Medical Visualization".

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.cag.2010.07.004.

References

[1] Fuchs R, Hauser H. Visualization of multi-variate scientific data. Computer Graphics Forum 2009;28(6):1670–90.
[2] Laramee RS, Hauser H, Doleisch H, Vrolijk B, Post FH, Weiskopf D. The state of the art in flow visualization: dense and texture-based techniques. Computer Graphics Forum 2003;23(2):203–21.
[3] Kindlmann G. Superquadric tensor glyphs. In: Proceedings of IEEE TVCG/EG symposium on visualization; 2004. p. 147–54.
[4] Kniss J, Kindlmann G, Hansen C. Multidimensional transfer functions for interactive volume rendering. IEEE Transactions on Visualization and Computer Graphics 2002;8(3):270–85.
[5] Busking S, Botha CP, Post FH. Direct visualization of deformation in volumes. Computer Graphics Forum 2009;28(3):799–806.
[6] van Walsum T, Post FH. Selective visualization of vector fields. Computer Graphics Forum 1994;13(3):339–47.
[7] Shen HW, Woodring J. Multi-variate, time varying, and comparative visualization with contextual cues. IEEE Transactions on Visualization and Computer Graphics 2006;12(5):909–16.
[8] Henze C. Feature detection in linked derived spaces. In: Proceedings IEEE visualization; 1998. p. 87–94.
[9] Gresh DL, Rogowitz BE, Winslow RL, Scollan DF, Yung CK. WEAVE: a system for visually linking 3-D and statistical visualizations applied to cardiac simulation and measurement data. In: Proceedings of IEEE visualization 2000; 2000. p. 489–92, 597.
[10] Doleisch H, Gasser M, Hauser H. Interactive feature specification for focus+context visualization of complex simulation data. In: VISSYM '03: Proceedings of the symposium on data visualisation 2003, Eurographics Association; 2003. p. 239–48.
[11] Blaas J, Botha CP, Post FH. Interactive visualization of multi-field medical data using linked physical and feature-space views. In: Museth K, Ynnerman A, Möller T, editors. Proceedings of eurographics/IEEE-VGTC eurovis; 2007. p. 123–30.
[12] Post FH, Vrolijk B, Hauser H, Laramee RS, Doleisch H. The state of the art in flow visualisation: feature extraction and tracking. Computer Graphics Forum 2003;22(4):775–92.
[13] Hege HC, Theisel H, Seidel H-P, Weinkauf T. Saddle connectors—an approach to visualizing the topological skeleton of complex 3D vector fields. In: Proceedings, IEEE visualization; 2003. p. 225–32.
[14] Griebel M, Preusser T, Rumpf M, Schweitzer M, Telea A. Flow field clustering via algebraic multigrid. In: Proceedings, IEEE visualization; 2004. p. 35–42.
[15] McKenzie A, Lombeyda S, Desbrun M. Vector field analysis and visualization through variational clustering. In: Proceedings, eurographics/IEEE-VGTC symposium on visualization, vol. 2005; 2005. p. 29–35.
[16] Ma K-L. Machine learning to boost the next generation of visualization technology. IEEE Computer Graphics and Applications 2007;27(5):6–9.
[17] Viola I, Kanitsar A, Gröller ME. Importance-driven volume rendering. In: Rushmeier H, Turk G, van Wijk J, editors. Proceedings of IEEE visualization 2004; 2004. p. 139–45.
[18] Ropinski T, Praßni JS, Steinicke F, Hinrichs KH. Stroke-based transfer function design. In: IEEE/EG international symposium on volume and point-based graphics; 2008. p. 41–8.
[19] Dinh HQ, Xu L. Structural, Syntactic, and Statistical Pattern Recognition. In: Lecture notes in computer science, vol. 5342; 2008. p. 187–96.
[20] Seidel H-P, Theisel H, Rössl C. Using feature flow fields for topological comparison of vector fields. In: Proceedings vision modeling and visualization; 2003. p. 521–8.
[21] van Den Berg RA, Hoefsloot HCJ, Westerhuis JA, van Der Werf MJ, Smilde AK. Centering, scaling, and transformations: improving the biological information content of metabolomics data. BMC Genomics 2006;7:142.
[22] Heimann T, Meinzer H-P. Statistical shape models for 3D medical image segmentation: a review. Medical Image Analysis 2009;13(4):543–63.
[23] Krüger J, Westermann R. Acceleration techniques for GPU-based volume rendering. In: IEEE visualization proceedings; 2003. p. 287–92.
[24] Saito T, Takahashi T. Comprehensible rendering of 3-D shapes. In: Proceedings, ACM SIGGRAPH; 1990. p. 197–206.
[25] Diefenbach PJ. Pipeline rendering: interaction and realism through hardware-based multi-pass rendering. PhD thesis, University of Pennsylvania; 1996.
[26] Ritschel T, Grosch T, Seidel H-P. Approximating dynamic global illumination in image space. In: Proceedings symposium on interactive 3D graphics and games; 2009. p. 75–82.

[27] Westin CF, Banks DC. Global illumination of white matter fibers from DT-MRI Data, mathematics and visualization. Berlin, Heidelberg: Springer Berlin Heidelberg; 2008. p. 173–84 [chapter 10].
[28] Kniss J, McCormick P, McPherson A, Ahrens J, Painter J, Keahey A, et al. Interactive texture-based volume rendering for large data sets. IEEE Computer Graphics and Applications 2001;21(4):52–61.
[29] Dick C, Georgii J, Burgkart R, Westermann R. Stress tensor field visualization for implant planning in orthopedics. IEEE Transactions on Visualization and Computer Graphics 2009;15(6):1399–406.
[30] Stompel A, Lum EB, Ma K-L. Visualization of multidimensional, multivariate volume data using hardware-accelerated non-photorealistic rendering techniques. In: Proceedings of 10th pacific conference on computer graphics and applications 2002; 2002. p. 394–402.