Vis Comput (2011) 27: 347–363
DOI 10.1007/s00371-010-0541-z

ORIGINAL ARTICLE

Image-based rendering of intersecting surfaces for dynamic comparative visualization

Stef Busking · Charl P. Botha · Luca Ferrarini · Julien Milles · Frits H. Post

Published online: 9 December 2010

Abstract Nested or intersecting surfaces are proven techniques for visualizing shape differences between static 3D objects (Weigle and Taylor II, IEEE Visualization, Proceedings, pp. 503–510, 2005). In this paper we present an image-based formulation for these techniques that extends their use to dynamic scenarios, in which surfaces can be manipulated or even deformed interactively. The formulation is based on our new layered rendering pipeline, a generic image-based approach for rendering nested surfaces based on depth peeling and deferred shading.

We use layered rendering to enhance the intersecting surfaces visualization. In addition to enabling interactive performance, our enhancements address several limitations of the original technique. Contours remove ambiguity regarding the shape of intersections. Local distances between the surfaces can be visualized at any point using either depth fogging or distance fields: Depth fogging is used as a cue for the distance between two surfaces in the viewing direction, whereas closest-point distance measures are visualized interactively by evaluating one surface's distance field on the other surface. Furthermore, we use these measures to define a three-way surface segmentation, which visualizes regions of growth, shrinkage, and no change of a test surface compared with a reference surface.

Finally, we demonstrate an application of our technique in the visualization of statistical shape models. We evaluate our technique based on feedback provided by medical image analysis researchers, who are experts in working with such models.

Keywords Comparative visualization · Image-based rendering · Surface comparison · Nested surfaces

Electronic supplementary material The online version of this article (doi:10.1007/s00371-010-0541-z) contains supplementary material, which is available to authorized users.

S. Busking (✉) · C.P. Botha · F.H. Post
Data Visualization Group, Delft University of Technology, Delft, the Netherlands
e-mail: s.busking@tudelft.nl

C.P. Botha
e-mail: c.p.botha@tudelft.nl

F.H. Post
e-mail: f.h.post@tudelft.nl

C.P. Botha · L. Ferrarini · J. Milles
Division of Image Processing (LKEB), Department of Radiology, Leiden University Medical Center, Leiden, the Netherlands

C.P. Botha
e-mail: c.p.botha@lumc.nl

L. Ferrarini
e-mail: l.ferrarini@lumc.nl

J. Milles
e-mail: j.r.milles@lumc.nl

1 Introduction

In this paper, we examine one class of solutions to the problem of comparing the shapes of 3D surfaces. Comparison of data plays an important role in many areas of scientific research. Visualization can be useful to support comparative data analysis. The most common approach to comparative visualization of surfaces is a simple side-by-side display (with similar viewing conditions) of the two surfaces under consideration. Such an approach relies on memory to compare details of the surfaces, and local distances between surfaces are hard to estimate. We identify the following requirements for an effective comparative visualization:

– Differences should be made *explicit*, alerting the user to the presence and nature of all differences present.

– Visualization of differences should be *local*, showing not only the presence but also the precise location and extent of all differences. In medical applications, for instance, local information is required for understanding differences between patients or studying changes in specific biological structures over time.
– The visualization should be able to show *relevant* differences and hide irrelevant ones.

As an example of what is meant by relevance, consider the alignment of surfaces prior to comparison. Inaccuracies in the registration process can result in misalignment of the resulting surfaces. This misalignment, however, is usually not relevant to the researcher's problem. An ideal visualization could automatically distinguish between relevant and irrelevant differences, and show only the former. However, the notion of relevance is a highly application-dependent property, which can generally only be decided by the researcher. The use of user-feedback and *interactivity* is therefore essential to deal with this issue. Various applications can benefit from interactive visualization:

– User interaction or guidance in the registration process.
– Local registration for exploring differences between specific parts of the objects under consideration, ignoring more global differences.
– Analysis and comparison of dynamic or deformable surfaces.

In this paper we present a visualization for the comparison of 3D surfaces. Our visualization is based on the proven *intersecting surfaces* technique, first introduced and evaluated by Weigle et al. [1]. Our contribution consists of three aspects: We present an alternative, image-based implementation of this technique, which enables interactive performance even when manipulating alignment or dealing with dynamic objects. Furthermore, we present enhancements designed to address specific limitations of the existing technique. Finally, we present a case study, evaluating the suitability of intersecting surfaces and our enhancements for the visualization of statistical shape models [2].

Figure 1 shows the comparison of two segmentations of the same brain ventricle MRI scan using our technique. Such a visualization may give important information on the characteristics of a new segmentation algorithm. The yellow surface represents the baseline segmentation, while the blue surface shows a different segmentation. This means blue areas and yellow glyphs represent areas not covered by the new segmentation, while yellow areas and blue glyphs represent areas covered by this segmentation which are not in the baseline segmentation. Differences are remarkably symmetric in overall shape, but the lack of coloring due to fog indicates distances are small and contain little local variation. This may indicate these could be due to variations in the segmentation process. In Fig. 1(d) we applied a threshold at a distance of 1 mm, which reveals an area of considerable difference in the shape of the segmented ventricles.

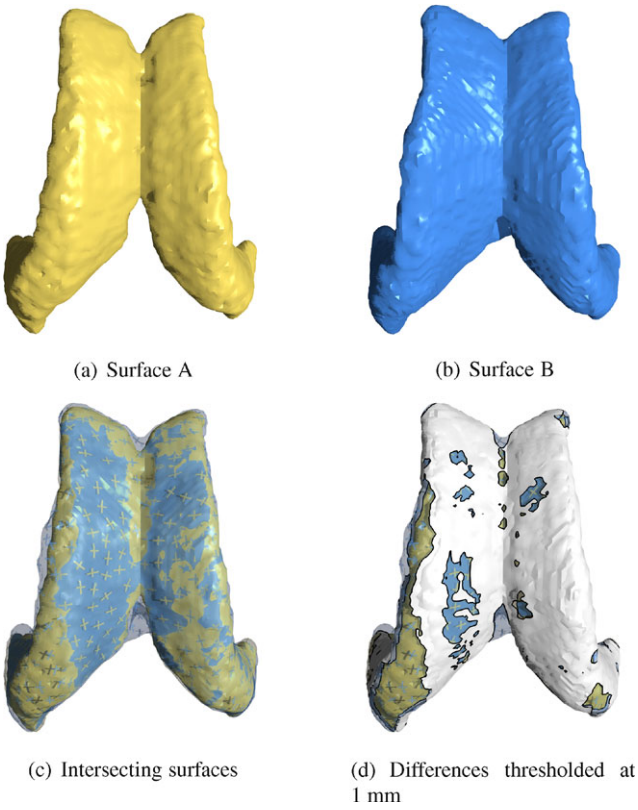


Fig. 1 Comparative visualizations of two partial brain ventricle surfaces segmented from MRI data. The *top figures* show the separate surfaces, illustrating the difficulty to locate differences without an intersecting surfaces approach. The basic intersecting surfaces visualization [1] (*bottom left*) clearly shows some symmetry in the differences. Our visualization adds intersection contours and thresholding to suppress smaller differences. The resulting 3-way visualization (*bottom right*) reveals an area of considerable change

Our implementation of the techniques described in this paper is available as part of the open-source NQVTK library (http://nqvtk.googlecode.com/).

2 Related work

In this section we first define the position of our technique in the comparative visualization process. We use this as a framework to present and discuss work related to our technique, including the intersecting surfaces visualizations by Weigle et al., on which our approach is based. As a meaningful discussion of work related to our implementation of these techniques depends on the details thereof, such work is discussed in Sect. 4.1.

2.1 Domain matching and comparison

Comparison can occur at any stage of the standard visualization pipeline [3]. However, the process always requires data sets to be aligned before differences can be determined and/or visualized. Many solutions exist for this *domain matching* step, the details of which are mostly application-dependent. In part this is because the processes of aligning data and extracting differences cannot always be cleanly separated. For instance, a body of work exists which applies non-rigid registration to volume data [4, 5] or to features extracted from it [6]. The deformation field resulting from this process is then analyzed to determine differences. This deformation field will typically also contain irrelevant differences due to imperfect alignment, such as tissue deformations caused by patient movement rather than by pathological processes.

Our technique can be applied after domain matching in order to directly visualize the remaining differences between two surfaces. This enables such differences to be studied in detail, but also gives insight into the quality of the matching itself. We aim to be independent of the choice of domain-matching technique. We only assume that a good (possibly application-specific) solution for this stage is available: i.e., one that does not remove any of the relevant differences and preferably leaves a minimum of irrelevant differences. As perfect matching (i.e., separation of relevant and irrelevant differences) is often impossible, our technique should provide enough information to assess the relevance of the differences.

As stated in Sect. 1, our visualization of such differences should be explicit and local. This means a suitable technique should determine and extract localized differences after the matching step rather than simply visualizing the aligned data sets. These differences should then be mapped to clear elements in the resulting visualization. Many existing comparison techniques are implicit in that they skip this extraction step, which means the act of comparison is left to the user.

2.2 Comparative visualization of 3D surfaces

Numerous measures have been proposed that can be used to express the difference (or similarity) in shape between two surfaces; most notably in the area of image retrieval [7–9] and shape retrieval [10, 11]. However, most of these measures, such as the commonly used Hausdorff distance, only express similarity at a global level.

The local visualization of differences can yield important insights which might not be obvious from global measurements. However, only a few techniques have been presented for comparing shape locally [12, 13]. Local distance measures often require establishing some form of *correspondence* between the surfaces (i.e., domain matching). A commonly used method for visualizing local distance measures is to display these on one of the two surfaces using an appropriate color map [12, 14, 15]. This has the disadvantage that only one of the surfaces is shown; the shape of the second surface is not obvious.

A way to overcome this is to make one or both surfaces transparent and overlay them in the visualization. Such a *nested surfaces* approach (used by, e.g., Tory et al. [16]) shows all context information. Similar approaches have been proposed in uncertainty visualization (see, e.g., Johnson and Sanderson [17]). However, in these visualizations the identification of differences is left to the user. This is complicated by the fact that overlaying multiple transparent and potentially intersecting surfaces results in an image which is not always clear to a user. In particular, understanding the shapes of transparent surfaces and classifying surfaces as being either in front or behind other surfaces can be difficult perceptual tasks.

Attempts have been made to resolve these issues. Textures are commonly applied to improve shape perception of transparent surfaces [18]. Interrante et al. [19] proposed using stroke textures based on the directions of principal curvatures. Bair and House [20] investigated the use of several types of grid textures, and performed user studies on their effects on perception of surface shape. Bruckner et al. [21, 22] approached similar problems in volume rendering using illustrative techniques. They proposed using interactive control over a special opacity function that allows a user to focus on specific objects within a volume while also keeping contextual surfaces in view.

Weigle et al. [1, 23] proposed the use of *constructive solid geometry* (CSG) operations to solve the perceptual problem of inside/outside classification. Their intersecting surfaces technique, described in detail in the next section, forms the basis for the visualization presented in this paper. User studies performed by Weigle et al. have shown these visualizations to be effective for the comparison and understanding of surface shapes.

2.3 Base visualization

Because of its proven effectiveness, we use the intersecting surfaces technique by Weigle et al. as the basis for our visualization. The technique uses CSG operations to extract differences between two surfaces: The intersection of the closed objects formed by the surfaces represents the volume in common between both objects, and is rendered as an opaque object. The remaining parts of the two surfaces represent differences, and are rendered transparently in order to show the intersection behind them. This solves the inside/outside classification problem, as parts of each surface inside the other are always opaque and outside parts are always transparent.

As suggested by Interrante [19], Weigle's visualization includes curvature-aligned glyphs on the transparent parts

of the surfaces to better illustrate surface shape. In order to visualize local distances between the two surfaces, Weigle's method relies on shadows cast by these glyphs. As an alternative to shadows, point-correspondence glyphs can be used [23]. These are line segments which connect corresponding points on the two objects that are being compared.

In their extensive user studies, Weigle et al. compared their visualization to a number of previously existing techniques [1, 23]. The studies show that the intersecting surface visualizations, both with shadows and with point-correspondence glyphs, are more effective than the existing techniques. The contribution of this paper includes improvements which address specific limitations of Weigle's technique.

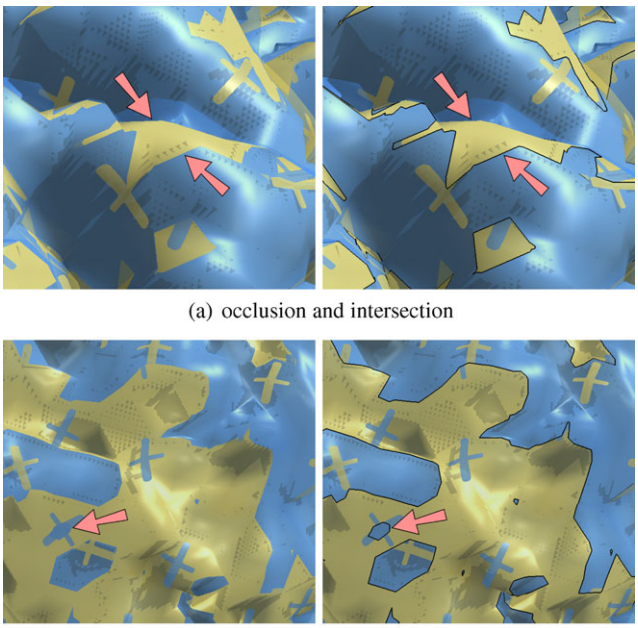
3 Enhanced intersecting surfaces

In this section we present our enhanced visualization for the comparison of 3D surfaces. We first identify and discuss limitations of the existing technique, and use these to introduce and motivate our enhancements: an image-based implementation for increased flexibility and interactive performance, dynamic intersection contours, integration of distance cues and suppression of irrelevant differences.

3.1 Limitations of the original visualization

In their work, Weigle et al. identified a number of shortcomings of their intersecting surfaces implementation. We summarize these issues below, and add two more issues (5 and 6) found when applying our requirements, described in Sect. 1:

1. Folded surfaces and other forms of self-occlusion can hide differences. Weigle proposed interactive control of the camera as a possible solution, which is also possible in our technique. However, similar issues occur when glyphs are placed near intersections, which may cause confusion regarding the shape of those intersections (see Fig. 2).
2. Due to their fixed spacing, surface glyphs can only illustrate shape at a fixed scale. Because of this, small-scale details can be missed. Weigle proposed making glyph spacing adaptive to local shape characteristics. However, we found that the sparse nature of the glyphs also means that the distance between surfaces can only be estimated around glyph/shadow pairs, or around point-correspondence glyphs.
3. The visualizations are not useful when objects are too different, or are not aligned properly during domain matching. In this case, Weigle recommends that explicit correspondence information be included in the visualization.
4. The approach is not easily extended to more than two surfaces.



(a) occlusion and intersection

(b) small intersection hidden due to glyph placement

Fig. 2 Without contours, intersections between the *blue surface* and the *yellow surface* are not always obvious. Marking all intersections differentiates them from occlusions and highlights smaller features, which might otherwise be hidden. The small triangular artifacts are side-effects of our shadow mapping implementation and not a result of our technique

5. In the base visualization, differences are shown regardless of their relevance, with no way to distinguish between relevant and irrelevant differences.
6. Importantly, Weigle et al. did not apply their techniques to the comparison of dynamic surfaces and the original implementation therefore does not support scenarios where the objects are not static in shape or in their relative positions and orientations.

3.2 Enhancements

While simultaneous comparison of more than two surfaces is considered outside the scope of this paper, we present enhancements to the base visualization that address each of the remaining issues in the list above. The following sections discuss the motivation for and design of these enhancements. The details of their implementation can be found in Sect. 4.

3.2.1 Image-based implementation

In this paper, we present an image-based implementation of the intersecting surfaces visualization. Our implementation provides interactive control of the viewpoint, suggested by Weigle as a possible solution to the first issue. It also enables the visualization to be applied to dynamically deforming

objects with interactive performance (issue 6), as the intersection is not computed geometrically. This in turn enables new applications for the visualizations. For instance, occlusion issues could also be solved by a dynamic peel-away approach. Additionally, our technique could enable interactive domain-matching tools, or interactive local alignment of corresponding features. Such tools could help in dealing with objects with significant global differences but local similarities, such as described in issue 3. While we consider such solutions in future work and do not discuss them further in this paper, we do present an example of dynamic objects in our case study (Sect. 5).

In Sect. 4.1 we present a generic rendering pipeline on which we base our image-based implementation. This pipeline enables combinations of CSG rendering and transparent surfaces in a straightforward and extensible way, which can also be useful in other visualization scenarios.

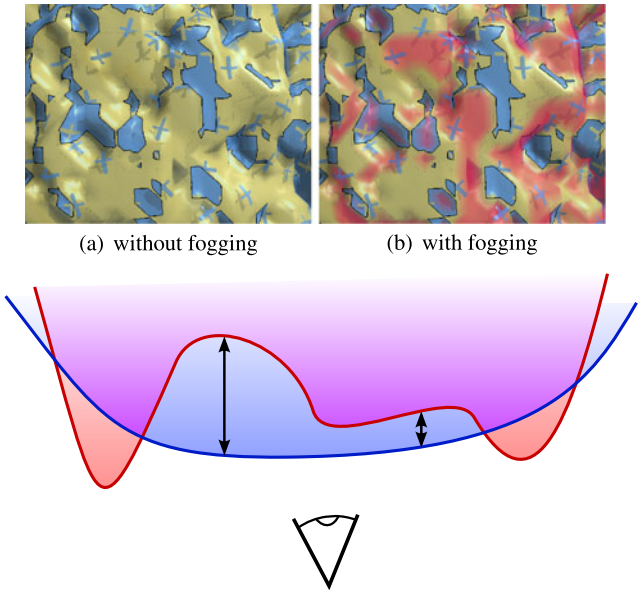
3.2.2 *Intersection contours*

As described in issue 1 and shown in Fig. 2, the complex appearance of the intersecting surfaces can sometimes cause confusion. Specifically, occlusions might be interpreted as intersections and vice versa (Fig. 2(a)). Glyph placement can also cause confusion when this occurs on the intersection curve, potentially distorting the shape of intersections or hiding them completely (Fig. 2(b)). If the intersections between the surfaces are complex, the visualization becomes more cluttered. In this case, such issues are likely to become problematic, as small but possibly relevant differences may be missed.

Explicitly marking the intersection curves in the visualization removes any ambiguity in these cases. Contouring is a commonly used technique in illustrative rendering. Lines can be drawn to emphasize important boundaries in a visualization. While it may seem more natural to enhance occlusions rather than intersections, such contours are view-dependent and can therefore make the visualizations harder to perceive. Furthermore, in our case the intersections between objects are actual features of interest, and contouring them serves the dual purpose of highlighting these features. Similarly, changing the color of the outer surface (and glyphs) would solve problems as shown in Fig. 2(b). However, as color is used to identify surfaces, such a change would complicate the visualization. Contours provide a solution that works independently of the choice of surface coloring and texturing. The intersection curves of the surfaces provide clear contours for the "difference" areas in the visualization. They also enable clear distinction between intersection and occlusion.

3.2.3 *Local distance cues*

In addition to the well-known difficulties of illustrating the shape of transparent surfaces, the second issue is caused by



(a) without fogging (b) with fogging

(c) cross section showing area with varying depth

Fig. 3 The choice of viewing direction and placement of glyphs may hide important information about the size and shape of differences. By including local distance feedback in the visualization (here using *red fog* added between the surfaces), larger differences are made more obvious. The *bottom figure* shows a simplified schematic cross section, where adding fog would highlight the larger distance on the left

local distance information being visualized only sparsely in the original intersecting surfaces technique. Such information is important for understanding the shape and size of differences, and can also help in understanding the shape of the objects themselves.

In Weigle's technique, the user has to rely on shadows cast by glyphs in order to judge local distances between the surfaces. However, on many complex surfaces it may not be straightforward to match up glyphs with their shadows. It may also be the case that there simply is no glyph present in an area of interest. One example of problematic shape is shown in Fig. 3. In this case, areas with increased depth (a simplified example is shown in the cross section, Fig. 3(c)) are not immediately obvious in the visualization (Fig. 3(a)). When using glyphs, a smart placement of these glyphs (and their shadows) might solve such problems. However, this could lead to an uneven distribution of glyphs over the surface, which causes other perceptual issues.

We integrate two forms of local distance cues to alert the user to these areas in an intuitive way. Specifically, we simulate fogging between the inner and outer surfaces as a viewpoint-dependent distance cue and integrate closest-point distance for viewpoint-independent feedback. These enhancements also provide a way to visualize the shape of the surfaces at any scale, albeit relative to each other.

In fogging, we add a third color between surfaces, the amount of which is computed in terms of the distance be-

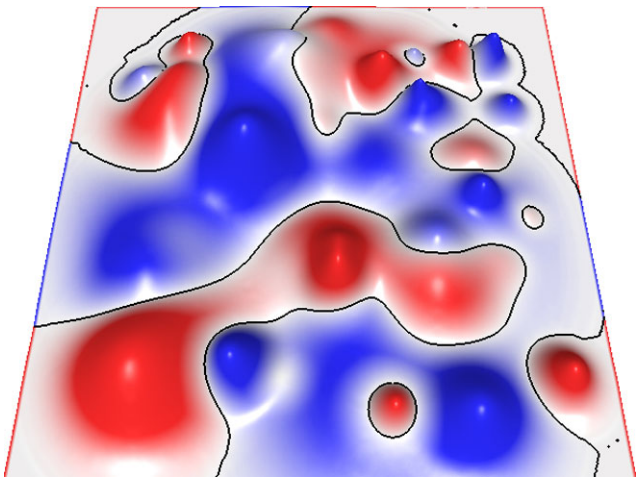


Fig. 4 Distance fields can be used to color each surface by its distance to the other, even when surfaces are moved relative to each other

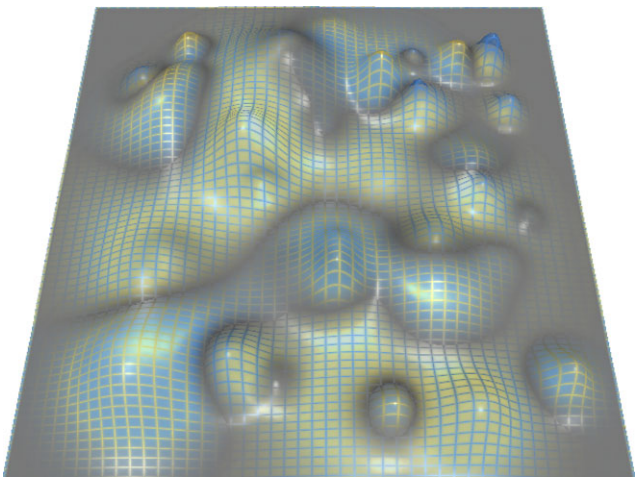


Fig. 5 Mapping distance to saturation of the surface colors reduces the prominence of small differences, but causes perceptual issues due to the blurry appearance

tween these objects along the viewing direction and a user-controllable density parameter. The effect is that the areas of difference between the objects are highlighted more for larger distances. As can be seen in Fig. 3(b), differences in depth between surfaces can be seen clearly after fog is added. Interactive manipulation of fog density can help a user compare relative depths of these differences. This interactivity also eliminates the problem that fog obscures the shape of the inner surface, as the fog can simply be removed or made less dense in order to examine surface shape after differences have been identified.

As fogging is view-dependent, results may change when the viewpoint is moved. We integrate closest-point distance as a view-independent alternative (Fig. 4), and enable coloring of either surface based on this distance. Our implementation, described in Sect. 4.3.2, achieves interactive performance even when objects are moved relative to each other, meaning that this technique could also be used in an interactive domain-matching solution such as described earlier.

3.2.4 Relevance filtering

Issue 5 originates directly from our requirements. If objects differ in some areas and are identical in others, the intersecting surfaces visualization clearly shows such differences. In real-world data, however, noise and other inaccuracies are often a concern, which can cause many small differences between objects. In intersecting surfaces, each of these will be shown with similar visual impact as other differences, cluttering the resulting images and distracting users from those other, possibly more interesting differences.

We use the distance measures introduced in the previous section in order to allow users to suppress smaller differences in the visualization. We selected to filter based on size, as this is a common criterion for considering the relevance of differences. We experimented with mapping distance to the saturation of the surface color (Fig. 5). This has the effect of both surfaces fading to gray as the distance between them gets smaller. Small differences such as those caused by noise will therefore also appear gray, rather than the sudden change of object color they cause in the original visualization. Unfortunately, changing saturation over the surface negatively affects the ability to perceive object shapes.

A simpler solution is thresholding the local distance measure. This way, we allow users to suppress differences that are considered too small to be relevant. Essentially, this has the effect of widening the intersection contours into areas of insignificant differences. We render such areas in an opaque neutral white color. The result, shown in Fig. 6 and Fig. 1, is a less cluttered visualization where the user is less distracted by small fluctuations between the surfaces. The threshold value can be controlled interactively by the user, to control the distance from which differences are considered large enough. This way, we allow different thresholds to be used when examining different parts of an object, and also enable exploring distances interactively by moving the threshold value.

One limitation of the current solution is that while in white areas the objects are considered similar, the two objects are still rendered as separate surfaces. This effect can be seen in the two bottom images in Fig. 6, where thresholds on the inner surface show up through the holes representing significant differences. Especially for higher threshold values, this can cause the already complex visualizations to be harder to interpret. In future work we aim to address this limitation by using techniques from illustrative rendering in order to reduce the visual complexity of the thresholded areas.

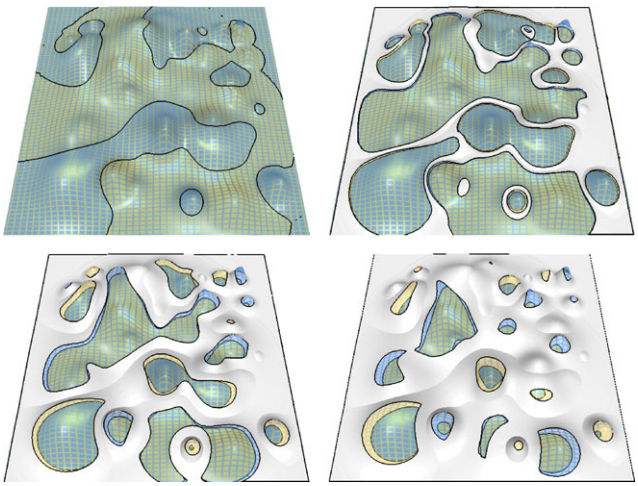


Fig. 6 Using increasing thresholds to hide small differences

4 Implementation

In this section we discuss our image-based implementation of the intersecting surfaces visualizations and of the enhancements described in the previous section.

While not the focus of this work, we implemented both opacity-modulating glyph textures (as used by Weigle et al. [1, 23]) and grid textures (see Bair et al. [20]) to help illustrate the shape of transparent surfaces. The integration of such techniques in the layered rendering pipeline (described next) is considered straightforward. Similarly, our implementation uses standard shadow mapping [24] to make the opacity-modulating textures cast shadows on the inner surface, as recommended by Weigle et al. [1, 23].

4.1 The layered rendering pipeline

We introduce the *layered rendering pipeline* (Fig. 7) as a generic basis on which we build our image-based implementation of the intersecting surfaces technique. Layered rendering combines depth peeling, deferred shading and CSG in order to create an approach for surface rendering similar to ray casting, but at interactive rates. In the intersecting surfaces visualization, this pipeline enables interactive performance even when objects are deformed. It also facilitates the addition of further enhancements which address limitations of the original visualization.

As stated in our requirements, we would like to be able to deal with objects of arbitrary complexity, and support objects which are dynamic or interactively positioned relative to each other. This requirement means that a geometric approach to computing the CSG intersection, as used by Weigle et al. [1], is infeasible due to the complexity of such a computation. We therefore investigated possibilities for an image-based approach.

While image-based CSG solutions exist [25, 26], the layered rendering pipeline is an extension of depth peeling [27–29]: All image-based CSG approaches require multiple passes over the geometry, each extracting separate layers of geometry distinguished by the number of surfaces occluding them. Depth peeling, however, also ensures that the surfaces are extracted (and rendered) in strict front-to-back order, which is required in order to properly render any transparent surfaces. We extend the pipeline with a stage in which deferred shading [30, 31] is applied to each layer extracted from the geometry. Furthermore, we extend deferred shading to allow propagation of information between layers.

Weigle [23] described an image-based inside/outside classification approach based on depth peeling. Weigle's algorithm, however, is limited to two surfaces and is not a full CSG algorithm; any fragment appearing deeper than a fragment already labeled interior is also labeled interior. This makes, for example, clipping the geometry resulting from the CSG operation or rendering the intersection with transparency difficult to achieve. Other combinations of depth peeling and image-based CSG have been presented by Guennebaud et al. [32], used to handle transparent objects in point-based rendering, and Nienhaus et al. [33], who presented a method for illustrative rendering of CSG models. These combinations were designed specifically for their applications, however, and are not easily extended.

The layered rendering pipeline takes inspiration from ray casting. The ray-casting algorithm takes an image-order approach, where rays are cast from each image pixel. These rays traverse the scene and intersect any objects within. Due to this image-order approach, achieving a correct ordering for compositing transparent surfaces is inherent in the algorithm. Similarly, CSG operations are implemented easily by simply keeping track of object intersections along a ray.

The design of the layered rendering pipeline is based on several observations. Firstly, rendering an object using traditional rasterization essentially provides the intersection points of all rays cast from the image simultaneously for a single layer of geometry. Secondly, the depth peeling algorithm can be used "as is" to process all intersections in the scene in front-to-back order.

The layered rendering pipeline, therefore (see Fig. 7), consists of repeated execution of two stages in order to process the layers in the scene. The *scribe* stage performs depth peeling using a dual-depth buffer approach [28] and stores information about the current layer which is required for shading in an *info-buffer*. The information in this buffer is created by updating the info-buffer from the previous layer with information from the geometry of the current layer. This way such information can be propagated through the scene in a way similar to ray casting. The *painter* stage then compares this information with that of the previous layer(s) in order to create the image for this layer. Similarly

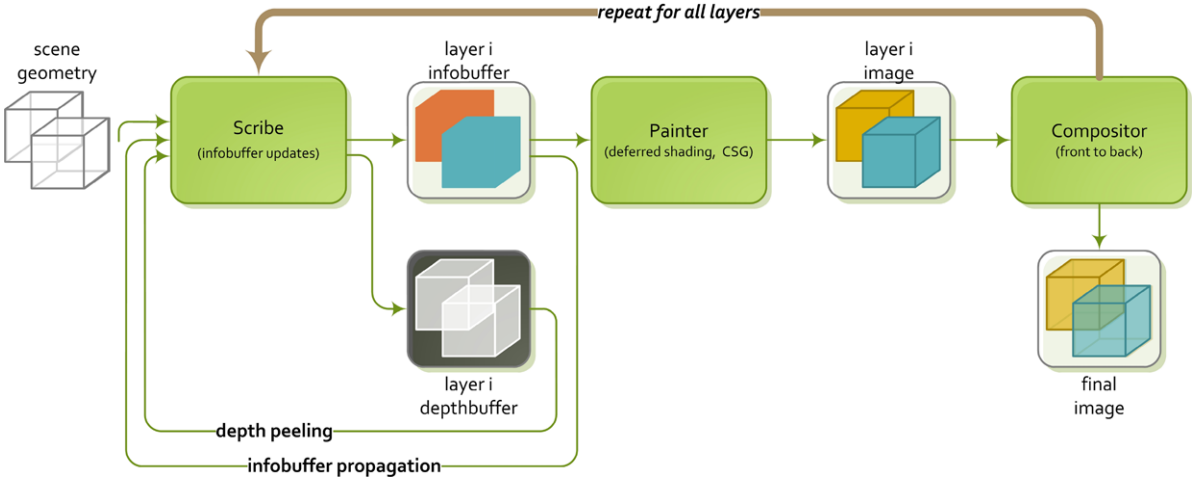


Fig. 7 The layered rendering pipeline. The scribe stage uses depth peeling to process the scene one depth layer at a time in front-to-back order. It propagates information between layers and also outputs information about each layer to the painter stage, which creates the image for that layer using a deferred shading approach. These images are composited to form the final visualization

to deferred shading, the painter stage does not require the scene geometry to be drawn, as all information required for shading is present in the info-buffers. The resulting images are composited using front-to-back blending to create the final image.

The total number of layers could be very large, depending on the depth complexity of the scene being rendered. However, because each subsequent layer affects at most the same pixels as the previous layer, and because of the front-to-back blending used, later layers may not noticeably influence the final image. Therefore rendering can be terminated after either a set number of layers is reached, or when the number of pixels in a layer is below some (small) fraction of the total image size. This number is determined using occlusion tests available in modern hardware.

In the following, we refer to the info-buffer for layer l_i as B_i. The data in B_i describes (for each pixel) properties either of the corresponding layer l_i, or of the depth slab s_i, which is the volume between l_i and l_{i+1}. An important part of B_i is the *in/out mask*, a bit mask which indicates for each pixel and for each of the objects in the scene whether s_i is inside or outside of the object. We assume all objects are closed, non-self-intersecting surfaces, although our approach still works with objects that extend into infinity in the viewing direction (e.g., landscape surfaces such as Fig. 6, viewed from above). In the scribe stage, the bit corresponding to the visible surface for each pixel is set to 1 if that surface is front-facing, and to 0 if it is back-facing. Other bits are left unchanged and are copied from B_{i-1}. The mask for B_0 needs to be initialized according to the camera's position w.r.t. the objects in the scene. However, it can generally be assumed in our visualizations that the camera is positioned outside of all objects.

The in/out mask provides a simple way to perform arbitrary CSG operations on the objects in the scene, as well as clipping objects to arbitrary geometry. Additionally, it solves a common problem in depth peeling regarding coplanar surfaces. In our implementation, the mask is stored in one of the 8-bit RGBA channels of the info-buffer texture. This means we can track up to 8 objects at a time, which is more than sufficient for the purposes of the intersecting surfaces visualization.

4.1.1 CSG rendering

The base visualization uses only the CSG intersection of objects. In future extensions, however, other CSG operations could be useful as well. For example, such operations could be used as part of area-of-interest selection in a focus + context visualization.

CSG rendering is as simple as applying the Boolean-valued CSG expression to the in/out masks for both B_i and B_{i-1} during the painter stage. Surfaces of the result of the CSG operation are characterized by a change in the value of this expression. Therefore, if the resulting value for some pixels on a layer is different from the value on the previous layer, that part of l_i is a surface of the object resulting from the CSG operation, and can be rendered as such (see Fig. 8).

4.1.2 Clipping

One advantage of this approach is that it results in the CSG surfaces on all depth layers rather than only a single layer. For instance, the Goldfeather algorithm [25] only provides the first depth layer of the CSG object. Similarly, Weigle's image-based classification technique only locates the first

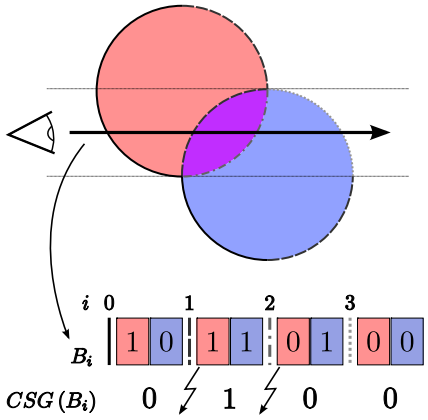


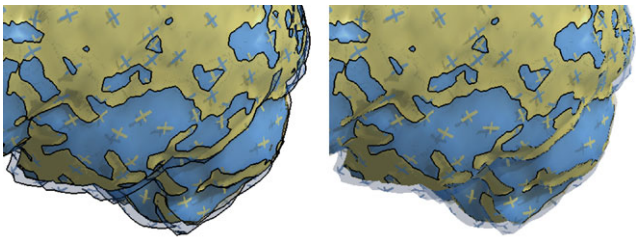
Fig. 8 Layered rendering for the CSG intersection of two spheres. In/out masks are shown for each depth slab for positions along the arrow. Changes in $CSG(B_i)$ correspond to surfaces of the CSG result

intersection surface along the viewing direction. Having correct CSG surfaces on all layers means it is easy to implement clipping techniques in the painter stage. These can be used, for instance, to enable viewing of interior structures of the CSG model. Furthermore, because we maintain an in/out mask between layers, the objects in the scene can be clipped against arbitrary geometry. Clipping against arbitrary closed objects is done trivially during the painter stage by discarding all fragments for which the bit corresponding to the clipping object is set in the in/out mask (indicating they are part of surfaces located inside this object).

4.1.3 Coplanarity

One important limitation of the depth-peeling algorithm is that it does not deal well with coplanar, overlapping surfaces. As only depth information is used to distinguish between surfaces in peeling, in the case of coplanar overlapping surfaces only one of the surfaces will be detected. In addition to causing visual artifacts when surfaces are not rendered, this will cause problems for a CSG implementation such as the one described above.

Our layered rendering approach offers an elegant solution to the problem. The problem stems from the fact that normal depth peeling only allows fragments with a depth value *larger* than the previous layer. In our *coplanarity peeling* algorithm, identical depth values are also allowed, as long as the in/out mask resulting from the previous layer indicates we have not rendered these before. Our assumption that surfaces are not self-intersecting enables us to perform only a single "greater or equal" test against the previous layer, rather than testing for equality separately, thereby simplifying the implementation.



(a) all detected contours (b) filtered contours

Fig. 9 Using depth comparison to filter the contours: silhouettes are removed while intersection contours remain

4.2 Intersection contours

Like the intersection objects themselves, intersection curves can be computed geometrically. Such computations, however, are complicated, hard to generalize and unsuited to interactive visualization. We use our layered rendering pipeline to create an image-based derivation for these contours.

In illustrative rendering, a common technique is to derive object silhouettes and feature contours from discontinuities in images containing depth values or surface normals respectively [31]. Similarly, we use the info-buffer to record, for each pixel, the identity of the object which contributed that pixel for the current layer. During the painter stage, we perform simple edge detection on this buffer by comparing pixels to their immediate 4-connected neighbors. This approach, however, leads to more than just the intersection contours (see Fig. 9(a)).

In order to filter this set of contours, we also store the depth values for each pixel in the layer. Our reasoning is as follows: if two neighboring pixels are on opposite sides of a true intersection contour, their depth values will be similar as well. While this assumption does not hold in general, we benefit from the fact that in most comparison scenarios objects are similar. If depth values on opposite sides of a detected edge fall within an epsilon range of each other, the edge is rendered in the image (Fig. 9(b)). The value of this epsilon depends on the scale of objects in the scene and is currently determined empirically on a per-application basis.

4.3 Integrating local distance information

As stated in Sect. 3.2.3, we integrated two types of local distance cues. Their implementations again take advantage of the layered rendering pipeline, and are described in the next sections.

4.3.1 Fogging

When using ray casting to render fog, colors and opacities are accumulated along the rays while they travel through

fog volumes. Similarly, we determine the amount of fog between two surfaces as a function of the distance between those surfaces. In our layered rendering pipeline, this means we can simply compare the depth values for the current layer l_i to those of the previous layer l_{i-1} in the painter stage, in order to determine the distance for each pixel. Based on this distance, we compute the accumulated fog opacity for the depth slab s_{i-1} and then blend the fog over the current layer. Distance can be mapped into opacity linearly; however, as multiple slabs may contribute to an area of fog, an exponential mapping yields better results.

As in CSG rendering, a Boolean expression can be applied to the in/out mask (in B_{i-1}) in order to specify the (combinations of) objects in which fogging should be applied. An alternative to fogging is distortion (e.g., defocussing, blurring) of the layer image due to refraction or translucent materials. To implement this, the layer should be rendered normally, after which the resulting image can be distorted in accordance with previous layers in order to produce the desired effect.

4.3.2 Distance fields

As discussed in Sect. 2, many shape (dis)similarity measures have been proposed in the literature. The values resulting from these measures change whenever either object is manipulated by the user. Recomputing the measure for each point on both surfaces may not be feasible in an interactive system. We therefore propose a solution based on distance fields.

A distance field is a sampling of a distance function, which maps positions in 3D space into the distance of that position to the surface. The simplest such function uses the Euclidean distance metric between the position in space and the position of the closest point on the surface. In the case of a closed surface, such a distance function could be signed, i.e., the sign of the distance could indicate whether a point is inside or outside of the object. Although creating such fields is computationally expensive, they only need to be computed once per object.

We use the Closest Point Transform algorithm by Mauch [34] to precompute (signed) distance fields for our surfaces. These distance fields are made available during rendering as 3D textures. When objects are rendered during the scribe stage of the pipeline, we sample the distance fields in order to obtain distances between the objects. To accomplish this, 3D locations corresponding to each surface pixel need to be transformed to the local coordinate frame of the other object, after which the corresponding 3D texture can be sampled. This effectively results in the distances of each point to the closest point on the other object. Additionally, objects can easily be manipulated in this approach; the corresponding transformations simply need to be taken into account when determining lookup locations in the distance fields.

Although this technique technically does not require the use of the info-buffer, we still benefit from the basic depth peeling and deferred shading options inherent in the layered rendering formulation, as well as allowing the visualization of the distances to be handled separately from their measurement. In our current implementation, the distance fields only work for static surfaces. In the case where one of the surfaces is dynamic, the distance field of the static surface can still be used to display the distance on the dynamic surface. In recent literature, GPU-accelerated methods have been presented [35] which could enable interactive performance in the fully dynamic case. Fogging works for static and dynamic surfaces, and combinations thereof.

4.4 Relevance

Implementing the relevance-thresholding technique in our framework is as simple as adding a threshold test after sampling the distance field in the scribe-stage. We replace our object identification in the layer information buffer by a classification. This classification can take one of three values. If for some fragment the distance is smaller than the threshold, the fragment is classified as being "equal." Otherwise, the fragment is classified as significantly belonging to one of the objects.

During the painter stage, we render "equal" fragments as opaque in a neutral color. Our contour detection approach can be performed on the new classification buffer (rather than on the object identification buffer) in order to emphasize the areas of large difference.

4.5 Performance

We created a prototype implementation of the techniques presented in this paper in C++, using VTK for data processing and OpenGL for rendering. The OpenGL shading language (GLSL) was used to implement all GPU-based algorithms, which require at least NVIDIA GeForce 7 graphics hardware (or equivalent). The resulting system allows for interactive rendering of the visualization, as well as interactive manipulation of the surfaces involved in the comparison, the viewpoint and the parameters of the various techniques.

Our layered rendering implementation makes full use of deferred shading. For this, our layer information buffer consists of three 32-bit render targets. The first two contain color and surface normals for use in deferred shading. The third is used to store surface classifications, the in/out-mask and surface depth values. As only this third buffer needs to be propagated between layers, the memory costs of the algorithm are those of five 32-bit images at the display resolution (see Sect. 4.1). The use of distance fields adds to this requirement; we used two 256^3-byte distance volumes, which provide sufficient accuracy for the purposes of our visualizations.

Tests were performed on a single core of an AMD Athlon 2.6 GHz dual core machine with 3 GB RAM and NVIDIA GeForce 8800 graphics. Images were rendered with shadow mapping enabled at 1000 × 700 resolution, resulting in frame rates between 10 and 50 frames per second for four depth layers (depending on surface complexity), which is sufficient for most intersecting surfaces visualizations. As noted, the use of shadow mapping requires scenes to be rendered twice per frame: once from the light's viewpoint and once from the camera. This means a doubling of frame rates can be achieved when shadow mapping is not required.

We used the brain ventricle segmentations (Fig. 1, 79 668 triangles), an artificial height field data set (Fig. 6, 1 044 484 triangles) and the ventricle shape models (Fig. 12, 3788 triangles) for our measurements. Both the number of triangles and the number of depth layers linearly influence the rendering performance. This is expected, as geometry is rendered once for each layer. Decreasing the maximum number of layers increases performance, but at the cost of image quality. A lower number of layers may cause display artifacts when large numbers of transparent surfaces overlap each other. In the case of the intersecting surfaces visualization, depth complexity of the scene rarely exceeds 4, due to the fact that the inner object is rendered opaque. However, the use of clipping techniques (as described in Sect. 4.1.2) may cause an increase of the number of layers required. This is because clipping is performed in the painter stages, meaning the hidden geometry and the clipping object itself both contribute to the depth complexity of the scene in the scribe stages.

The k-buffer architecture proposed by Bavoil et al. [36] could be used to greatly reduce the number of passes required to perform depth peeling. However, currently no hardware implementations of this architecture exist.

Depth peeling based algorithms are usually criticized for their high memory requirements. We note, however, that in our technique only information from the current and previous layer is required in order to render any given layer. Therefore, we require only two buffers in memory at all times. Furthermore, rather than storing each layer, we update the final image on the fly. This brings the total memory requirements of the layered rendering algorithm down to the memory required for two info-buffers and one final image. As only a subset of information needs to be propagated between layers, memory requirements can be further reduced.

The interactive performance achieved by our image-based implementation of the intersecting surfaces visualization enables new uses of such visualizations. As the intersection does not have to be computed explicitly, the objects involved in the comparison do not need to be static. The next section illustrates an application of the comparison of dynamic surfaces.

5 Case study

We performed an expert evaluation study of the techniques presented in this paper. This section describes our collaboration with two researchers in medical image analysis, who are experts in creating and analyzing statistical shape models. In this study, we applied our technique to visualize real-world statistical shape model data provided by the researchers.

The researchers are also the third and fourth authors of this paper. They contributed to this work by providing information on the problem domain and by giving feedback on our prototypes, but did not work on the implementation itself. In this context, their role as test users and evaluation subjects was not compromised.

5.1 Statistical shape models

In medical research, statistical shape modeling [2] is an important tool for understanding both differences in and variability of shapes of anatomical features. A shape model is created by first identifying corresponding points on all individuals in a population. Principal component analysis is then applied to determine the average shape of the population, as well as the principal modes of variation, ranked from most to least significant.

For this case study, the researchers provided us with a pair of statistical shape models used in research of Alzheimer disease (AD). The shape models represent the variation of the shape of the brain ventricles within two populations: a group of AD patients and a control group of healthy patients. It has been shown [37] that brain ventricles enlarge significantly due to the loss of brain tissue caused by AD. Figure 10 shows a visualization using our technique of the mean shapes of both groups, in which this shape difference is clearly visible.

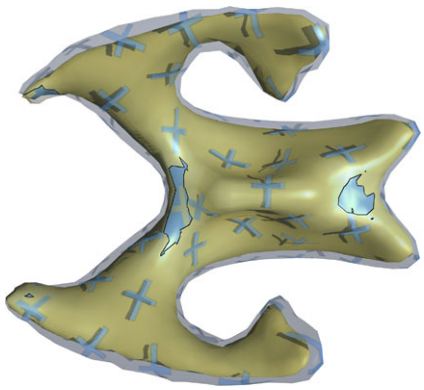


Fig. 10 Comparative visualization using our technique of the mean ventricle shape of the healthy group (*yellow*) versus the mean ventricle shape of the AD group (*blue*). The visualization clearly shows the latter shape to be larger, and therefore almost completely shown as *transparent*

Shape models are a special case of comparison, for which a known correspondence between points on both meshes is not only available but in some cases highly relevant to the comparison. Currently, such correspondence is visualized by the researchers by showing a single surface combined with *point-correspondence glyphs* (i.e., lines or arrows) pointing from this surface to where the other surface would be. Additionally, the researchers frequently use color mapping to visualize measures on a single surface.

5.2 Evaluation method

The evaluation consisted of two meetings, during which we discussed our visualizations with the researchers. The first meeting was an informal introduction of our visualization, consisting of a demonstration of our technique and a discussion of shape model visualization in general. We used the information gathered at this meeting to improve our prototype implementation, which was used to formally evaluate our technique in the second meeting. Also, we gathered information on currently used visualization techniques, and used the feedback to formulate a set of three shape research scenarios in which our visualizations could be applied.

1. Exploring shape differences between two static 3D surfaces. This scenario has been studied in detail by Weigle et al. [1].
2. Exploring variation of a single shape model. This includes both comparing new individuals to an existing model and examining variation within the model with respect to the mean shape. For this scenario as well as the next, the user can interactively control the shape model parameters to explore different shapes.
3. Validation (error-checking) of a shape model. A shape model is created by moving the vertices of a mesh to match different individuals in a population. If these vertices move along the surface of the model to match two individuals, the model may indicate a difference where in fact there is none. The models are validated by checking for such errors.

The comparison of two different shape models is another important topic for shape model visualization. However, this requires statistical information to be included in the visualization and is therefore considered future work.

We defined eight visualizations (Fig. 11) for exploring shape differences. The combinations were selected to allow for the evaluation of the strengths of each technique separately while still creating meaningful visualizations. To achieve this, we included both single surface and intersecting surfaces visualizations, intersecting surfaces with fogging or with shadow-casting glyphs, and combinations with or without relevance thresholding. To compare the proposed enhancements to currently accepted visualization techniques, we also included basic color mapping, point-correspondence glyphs and Weigle's original intersecting surfaces visualization (without our enhancements) in the evaluation. During the second meeting, the researchers were asked to grade these visualizations for each scenario.

We used a Likert scale [38] to grade agreement with the statement "Visualization X is suitable for use in scenario Y," where X and Y formed all combinations of the visualizations in Fig. 11 and the three scenarios mentioned above. Secondly, the researchers were asked to rank the visualizations in order of preference for each scenario. As a final

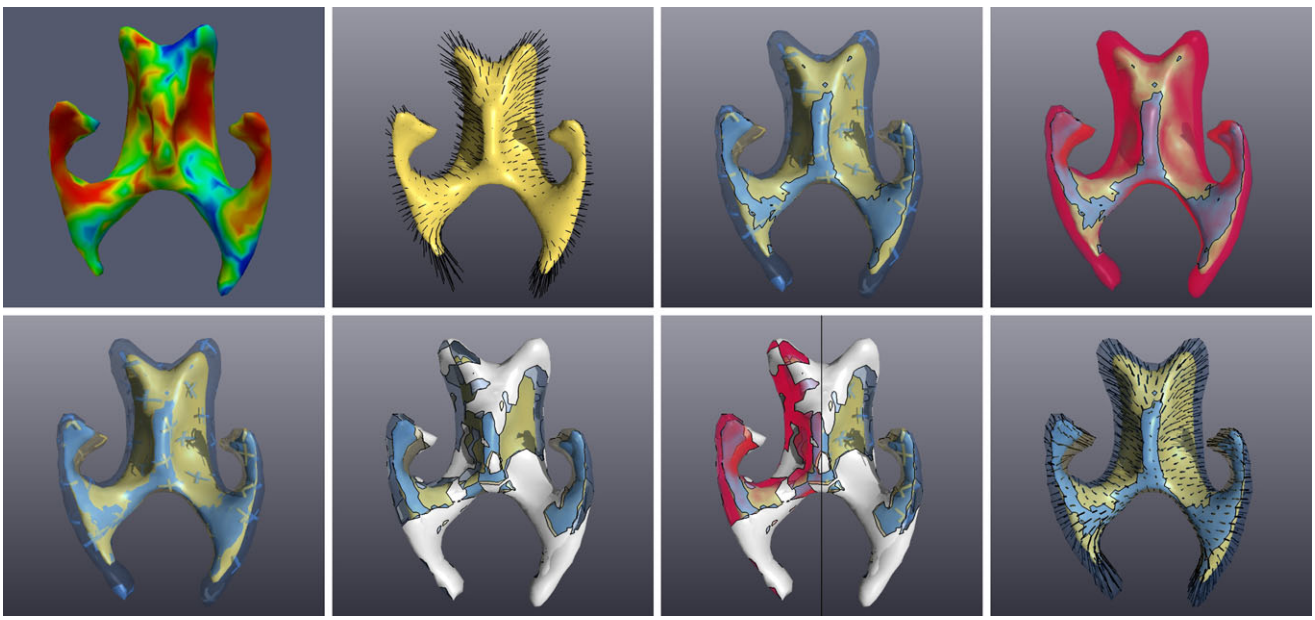


Fig. 11 The eight visualizations used in our evaluation

question, we asked the researchers to create their preferred visualization from the techniques in the prototype application.

In order to avoid distortion of the results, the researchers were explicitly warned about possible biases in using a Likert scale and told to evaluate each visualization independently of the others for the first question, leaving comparison for the ranking question. We limited the evaluation to the interactive use of each visualization, as such use is the primary aim of our contribution.

5.3 Software setup

We performed the case study using a specially prepared version of our software, which could be toggled between each of the predefined combinations of techniques. For the purposes of this study, we added an implementation of point-correspondence glyphs, similar to the glyphs currently in use by the researchers (shown in Fig. 12) and those used by Weigle [23].

In analyzing shape models, the researchers are most frequently interested in the statistical properties of the model. However, there are use cases in which direct shape comparisons can be useful. The interactivity provided by our technique enables the intersecting surfaces technique to be used on dynamically deforming surfaces. We adapted our software to allow for interactive deformation of the shape model mean surfaces along each of the modes of variation, in order to explore the variability of the statistical model. The researchers were allowed free use of this functionality for the evaluation of each technique in the latter two scenarios.

Based on feedback received from the researchers at the initial meeting, we enhanced the user interface of the prototype application for the formal evaluation by adding quantitative feedback about visualization parameters. We also added linked views showing the separate shapes involved in the comparison, with linked cursors between all views. While the researchers were asked to only use the intersecting surfaces visualization for comparison, they indicated the simple views were helpful in learning how to interpret the intersecting surfaces visualizations.

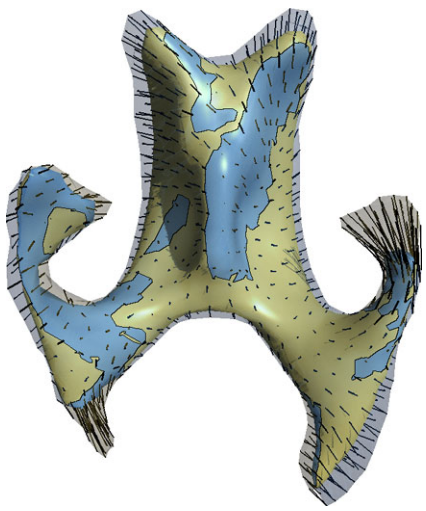


Fig. 12 Point-correspondence glyphs connect corresponding points on the mean and deformed shape model surfaces. Here they are used to show the third principal mode of variation in the control group shape model

5.4 Evaluation results

After evaluating each visualization, the experts were asked to explain their grades. The Likert scale grades and rankings served to provide important structure to this discussion. This section summarizes the feedback gathered from this process.

In all scenarios, the researchers preferred the intersecting surfaces visualizations over the single-surface visualizations. In a single-surface visualization, it is hard to determine the shape of the second surface. The researchers did note that for certain specific questions, such as examining statistical relevance of deformations, a color-mapped surface provides enough information.

Fogging and shadow-casting surface glyphs Both fogging and glyphs are preferred over plain intersecting surfaces, as without them it is very hard to see the 3D structure of and separation between both surfaces. However, the researchers had some concerns regarding the fog, causing them to prefer the glyphs. First, fogging adds visual complexity; opacity is not commonly used in current visualization techniques to indicate distance, and the fog color adds a third color to the visualization. Secondly, glyphs also served to emphasize the shape of the outer surface. Their removal, coupled with the fact that the fog obscures the inner surface, makes the shape of both surfaces harder to perceive. The fog color can be useful, however, to quickly direct attention to areas with large difference. Other techniques can then be used to explore surface shapes in detail.

Intersection contours When using the shadow-casting surface glyphs, the researchers noted that intersection contours are important to help distinguish between glyphs and intersections. The presence of these contours also helped the researchers to better perceive the intersecting surface visualizations in general.

Relevance The researchers see the benefit of suppressing small differences using a threshold. However, they indicated several points where the current visualization should be improved. First, the visual complexity of the current implementation is too high. Thresholding being performed on both surfaces separately clutters rather than suppresses these areas in the visualization. Because of this clutter, visualizations which included the thresholding technique were

ranked very low in the evaluation. More work should be done to reduce the visual complexity of the masked areas, e.g., using illustrative techniques. The presence of a third color (white) is acceptable in these visualizations, as the color has a well-defined meaning (areas with no relevant differences). Second, comparing thresholding to color mapping, the researchers indicated a need for better visualization of the values in non-masked areas. Additionally, a color scale should be used to highlight particularly large or significant differences in the unmasked areas.

Point-correspondence information While not normally available for general surface-to-surface comparison, correspondence information is essential for shape model validation (scenario 3), and should therefore always be included in these visualizations. The point-correspondence glyphs fulfill this requirement, but can be enhanced in several ways. Color mapping the glyphs by their length can help show the distance between corresponding points. Similarly to surface thresholding by distance, an option should also be added to filter glyphs that are too short to be relevant by thresholding, and the user should be able to toggle their display interactively.

One thing to note is that for single-surface visualizations, point-correspondence glyphs may point inside the surface. This makes such areas impossible to distinguish from areas of no change, unless the surface is made transparent (thereby causing other perceptual issues). The intersecting surfaces visualizations always show all point-correspondence glyphs. It was mainly for this reason that this was a preferred combination for the researchers.

5.5 Lessons learned

A recurring concern during the evaluation was the high visual complexity of the intersecting surfaces visualizations, especially when deforming surfaces are involved. Despite this, intersecting surfaces were ranked higher than single-surface visualizations for all scenarios. Overall, the researchers feel that the visualizations can be easily interpreted after some use. However, the initial learning curve is rather steep, and the additional complexity introduced by our fogging and thresholding techniques added to this. For exploration of the variation in a shape model, the researchers considered information about exact differences less important than for the other scenarios. This means techniques like thresholding, color mapping and the point-correspondence glyphs are best omitted in this scenario, in order to reduce visual complexity.

Scenario 3 furthermore showed the importance of tuning visualizations to application-specific requirements. The preferred visualization chosen by both researchers consisted of a basic combination of intersecting surfaces with contours and shadow-casting surface glyphs to aid perception of the shape of the transparent surfaces. However, due to the nature of statistical shape models, this visualization should also include point-corresponding glyphs, which can be colored by distance, filtered by relevance and toggled on or off by the user.

5.6 Summary and outlook

In this case study, we applied our technique to real-world data and problems to gauge their effectiveness. Our expert evaluation agrees with most of our own findings as well as those from Weigle's studies. However, it also indicates that more work should be done on reducing the visual complexity of the depth cuing (fogging) and relevance masking (thresholding) techniques, as well as of the intersecting surfaces technique in general. Given this, there is no absolute winner among the visualizations considered in this study. Due to this visual complexity, we need to tune the combination of techniques to answer specific questions rather than simply including every bit of information. Interactive performance in changing the visualization and its parameters certainly helps in making this process easy for the user.

We intend to continue the collaboration in future work to apply our technique to shape model comparison and population studies. Additionally, user studies should be performed at a larger scale to more thoroughly evaluate the techniques.

6 Conclusions and future work

We have presented technique for the interactive visualization and exploration of surface differences using intersecting surfaces. Our main points of contribution are:

- A new image-based formulation of the proven intersecting surfaces visualization, based on our new layered rendering pipeline. Our formulation provides interactive performance, even when the objects being compared are deformed or moved relative to each other. This interactivity enables intersecting surface techniques to be applied in new domains such as shape model visualization.
- Enhancements to the original intersecting surfaces technique. These include object-intersection contours, as well as the integration of local difference measures in the form of view-dependent depth cues and view-independent distance fields. Furthermore, we extend the intersecting surfaces visualization to a three-way segmentation of surface differences, where the visual impact of insignificant differences is reduced.
- A case study, exploring the usefulness of intersecting surfaces techniques for the visualization and comparison of statistical shape models.

Our layered rendering formulation offers a flexible framework for various rendering techniques which are otherwise not straightforward to implement. The approach has strong parallels with ray casting, in that information is propagated from the viewer into and through the geometric scene. This enables image-based implementation of various ray-casting-like algorithms for layered surfaces, such as CSG operations and fogging inside objects. However, as the method is still based on rasterization, performance is much higher than that of a ray-casting technique and the technique can take full advantage of hardware acceleration. Furthermore, due to the formulations sharing a common framework, techniques can be combined in a straightforward way.

Based on the layered rendering formulation, our technique creates an interactive visualization for the comparison of surfaces. As indicated by Weigle [23], this interactivity is an important aid for understanding the shapes of objects being compared. Interactivity during object manipulations such as relative movement and/or deformation also enables new applications for the intersecting surfaces visualization, such as the visualization of statistical shape models explored in our case study.

Our visualization addresses the requirements defined in Sect. 1: The visualization of differences is local and explicit, and we presented methods for the suppression of irrelevant differences in the visualization. The distance cues and relevance filtering techniques can be used with various distance metrics, as long as it is possible to create a distance field for such a metric.

Furthermore, our enhanced intersecting surfaces technique addresses most limitations of the original technique: Contour lines remove ambiguity between intersections, occlusions and surface glyphs, distance cues add additional inter-surface distance information at any point on the surfaces, and relevance filtering hides irrelevant information. Regarding the visualization of highly different objects or dealing with occlusion, we provide both suggestions and (through our framework) opportunities for solving these in future work.

The resulting visualization can provide an overview of differences for any given pair of surfaces. This can assist a researcher in selecting areas of interest and/or choosing further techniques to apply to quantitatively analyze these differences. Our evaluation has shown that intersecting surfaces can also be a useful technique for comparing dynamic objects, provided that: (1) The visualization is properly adapted to the problem domain, e.g., by adding correspondence feedback in the case of our shape models; (2) The visual complexity of the visualizations is kept as low as possible. For example, rather than combining the various techniques and enhancements, a better option is to allow switching between them one-by-one, based on user interests.

6.1 Future work

Further improvements can be made in the use of texture patterns to show local surface shape as well as differences. In particular, if a surface is deformed, comparison of the deformed pattern to the original may yield more insight into the nature of the deformation. Texture-based techniques, possibly shown on cross sections of the data, could help to illustrate these changes.

The use of (linked) cross-sectional views could also help to reduce the perceived visual complexity of our visualizations. On-demand slicing has the potential to show differences which are hard to interpret in 3D clearly in a 2D plane. Another option not yet explored in this work is the effect of adding legends and similar tools to aid interpretation of the visualizations. For instance, fogging could be easier to interpret if some simple geometry was used as a "legend": this legend could consist of two planes intersecting in the middle along a scale of distances, visualized using the same settings as the current visualization.

In our case study we applied our technique to the visualization of statistical shape models. The statistical aspects of these models are currently missing from the visualizations. In work [39] performed subsequently to that presented in this paper, we presented an application for the visualization of high-dimensional shape spaces. We plan to integrate the intersecting surfaces techniques into our software in the future, as well as to extend these visualizations to include feedback on the statistical properties of the model.

Given two shape models for different populations, statistically significant differences can be determined between the models themselves [37]. By including such statistical information, our visualizations could help to link such statistical differences to physical differences in shape. This would aid both in understanding the causes and in validating the statistical models.

Finally, there is a need for intuitive methods for the interactive exploration of differences within the data. Our use of layered rendering already enables objects to be manipulated interactively. We are looking for ways to guide this interaction in order to select and eliminate certain differences which are not relevant to the application domain. Additionally, we plan to integrate linked cross-sectional views and probing tools to provide quantitative statements about differences. Combined, these techniques will enable better exploration and visualization of relevant differences.

Acknowledgements This research is supported by the Netherlands Organization for Scientific Research (NWO), project number 643.100.503 "Multi-Field Medical Visualization".

References

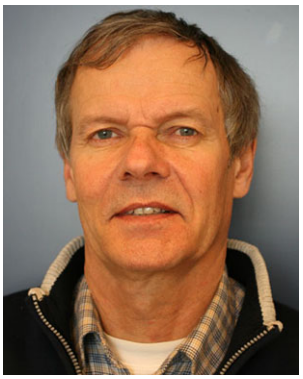
1. Weigle, C., Taylor, R.M. II: Visualizing intersecting surfaces with nested-surface techniques. In: IEEE Visualization, Proceedings, pp. 503–510 (2005). doi:10.1109/VISUAL.2005.1532835
2. Cootes, T.F., Taylor, C.J., Cooper, D.H., Graham, J.: Active shape models—their training and application. Comput. Vis. Image Underst. **61**, 38–59 (1995). doi:10.1006/cviu.1995.1004
3. Pagendarm, H.G., Post, F.H.: Comparative visualization—approaches and examples. In: Visualization in Scientific Computing, pp. 95–108. Springer, Berlin (1995)
4. Rey, D., Subsol, G., Delingette, H., Ayache, N.: Automatic detection and segmentation of evolving processes in 3D medical images: application to multiple sclerosis. Med. Image Anal. **6**, 163–179 (2002). doi:10.1016/S1361-8415(02)00056-7
5. Busking, S., Botha, C.P., Post, F.H.: Direct visualization of deformation in volumes. In: Hege, H.C., Hotz, I., Munzner, T. (eds.) Eurographics/IEEE-VGTC Symposium on Visualization, vol. 28, pp. 799–806 (2009). doi:10.1111/j.1467-8659.2009.01471.x
6. Subsol, G., Roberts, N., Doran, M., Thirion, J.P., Whitehouse, G.H.: Automatic analysis of cerebral atrophy. Magn. Reson. Imaging **15**, 917–927 (1997). doi:10.1016/S0730-725X(97)00002-7
7. Wilson, D.L., Baddeley, A.J., Owens, R.A.: A new metric for grey-scale image comparison. Int. J. Comput. Vis. **24**, 5–17 (1997). doi:10.1023/A:1007978107063
8. di Gesú, V., Starovoitov, V.: Distance-based functions for image comparison. Pattern Recogn. Lett. **20**, 207–214 (1999). doi:10.1016/S0167-8655(98)00115-9
9. Miranda, P.A.V., da Torres, S.R., Falcao, A.X.: TSD: a shape descriptor based on a distribution of tensor scale local orientation. In: SIBGRAPI, Proceedings, pp. 139–146 (2005). doi:10.1109/SIBGRAPI.2005.51
10. Veltkamp, R.C.: Shape matching: similarity measures and algorithms. In: IEEE Shape Modeling and Applications, Proceedings, pp. 188–197 (2001). doi:10.1109/SMA.2001.923389
11. Li, X., He, Y., Gu, X., Qin, H.: Curves-on-surface: a general shape comparison framework. In: IEEE Shape Modeling and Applications, Proceedings, pp. 38–43 (2006). doi:10.1109/SMI.2006.8
12. Masuda, T., Imazu, S., Auethavekiat, S., Furuya, T., Kawakami, K., Ikeuchi, K.: Shape difference visualization for ancient bronze mirrors through 3D range images. J. Vis. Comput. Animat. **14**, 183–196 (2003). doi:10.1002/vis.316
13. Gatzke, T., Grimm, C., Garland, M., Zelinka, S.: Curvature maps for local shape comparison. In: IEEE Shape Modeling and Applications, Proceedings, pp. 244–253 (2005). doi:10.1109/SMI.2005.13
14. Lim, I.S., Sarni, S., Thalmann, D.: Colored visualization of shape differences between bones. In: IEEE Computer Based Medical Systems, Proceedings, pp. 26–27 (2003)
15. Pichon, E., Nain, D., Niethammer, M.: A Laplace equation approach for shape comparison. In: SPIE Medical Imaging, Proceedings, vol. 6141, pp. 373–382 (2006)
16. Tory, M., Möller, T., Atkins, M.S.: Visualization of time-varying MRI data for MS lesion analysis. In: SPIE Medical Imaging, Proceedings, vol. 4319, pp. 590–598 (2001)
17. Johnson, C.R., Sanderson, A.R.: A next step: visualizing errors and uncertainty. IEEE Comput. Graph. Appl. **23**, 6–10 (2003). doi:10.1109/MCG.2003.1231171
18. Rheingans, P.: Opacity-modulating triangular textures for irregular surfaces. In: IEEE Visualization, Proceedings, pp. 219–225 (1996)
19. Interrante, V., Fuchs, H., Pizer, S.: Conveying the 3D shape of smoothly curving transparent surfaces via texture. In: IEEE Transactions on Visualization and Computer Graphics, pp. 98–117 (1997)
20. Bair, A., House, D.: A grid with a view: optimal texturing for perception of layered surface shape. IEEE Trans. Vis. Comput. Graph. **13**, 1656–1663 (2007). doi:10.1109/TVCG.2007.70559
21. Bruckner, S., Grimm, S., Kanitsar, A., Gröller, M.E.: Illustrative context-preserving volume rendering. In: Eurographics/IEEE-VGTC Symposium on Visualization, vol. 1, pp. 69–76 (2005)
22. Bruckner, S., Grimm, S., Kanitsar, A., Gröller, M.E.: Illustrative context-preserving exploration of volume data. IEEE Trans. Vis. Comput. Graph. **12**(6), 1559–1569 (2006). doi:10.1109/TVCG.2006.96. http://www.ncbi.nlm.nih.gov/pubmed/17073377
23. Weigle, C.: Displays for exploration and comparison of nested or intersecting surfaces. Ph.D. thesis (2006)
24. Williams, L.: Casting curved shadows on curved surfaces. In: Computer Graphics and Interactive Techniques, pp. 270–274 (1978). doi:10.1145/800248.807402
25. Goldfeather, J., Molnar, S., Turk, G., Fuchs, H.: Near real-time CSG rendering using tree normalization and geometric pruning. IEEE Comput. Graph. Appl. **9**, 20–28 (1989). doi:10.1109/38.28107
26. Wiegand, T.F.: Interactive rendering of CSG models. Comput. Graph. Forum **15**, 249–261 (1996)
27. Mammen, A.: Transparency and antialiasing algorithms implemented with the virtual pixel maps technique. IEEE Comput. Graph. Appl. **9**, 43–55 (1989). doi:10.1109/38.31463
28. Diefenbach, P.: Pipeline rendering: interaction and realism through hardware-based multi-pass rendering. Ph.D. thesis (1996)
29. Everitt, C.: Interactive order-independent transparency. Tech. rep., NVIDIA (2001). URL http://developer.nvidia.com/attach/6545
30. Deering, M., Winner, S., Schediwy, B., Duffy, C., Hunt, N.: The triangle processor and normal vector shader: a VLSI system for high performance graphics. In: ACM SIGGRAPH, Proceedings, vol. 22, pp. 21–30 (1988)
31. Saito, T., Takahashi, T.: Comprehensible rendering of 3-D shapes. In: ACM SIGGRAPH, Proceedings, pp. 197–206 (1990). http://doi.acm.org/10.1145/97879.97901
32. Guennebaud, G., Barthe, L., Paulin, M.: Splat/mesh blending, perspective rasterization and transparency for point-based rendering. In: IEEE/Eurographics/ACM Symposium on Point-Based Graphics, pp. 49–58 (2006)
33. Nienhaus, M., Kirsch, F., Döllner, J.: Illustrating design and spatial assembly of interactive CSG. In: Computer Graphics, Virtual Reality, Visualization and Interaction in {Africa}, Proceedings, pp. 91–98 (2006). doi:10.1145/1108590.1108605
34. Mauch, S.: A fast algorithm for computing the closest point and distance transform. Tech. rep., CalTech (2000)
35. Peikert, R., Sigg, C.: Optimized Bounding Polyhedra for GPU-Based Distance Transform. Springer, Berlin Heidelberg (2006), pp. 65–77. doi:10.1007/3-540-30790-7_5
36. Bavoil, L., Callahan, S.P., Lefohn, A., Comba, J.L.D., Silva, C.T.: Multi-fragment effects on the GPU using the K-buffer. In: ACM i3D, Proceedings, pp. 97–104 (2007). http://doi.acm.org/10.1145/1230100.1230117
37. Ferrarini, L., Palm, W.M., Olofsen, H., van Buchem, M.A., Reiber, J.H.C., Admiraal-Behloul, F.: Shape differences of the brain ventricles in Alzheimer's disease. Neuroimage **32**, 1060–1069 (2006). doi:10.1016/j.neuroimage.2006.05.048
38. Likert, R.: A technique for the measurement of attitudes. Arch. Psychol. **22**(140), 1–55 (1932)
39. Busking, S., Botha, C.P., Post, F.H.: Dynamic multi-view exploration of shape spaces. In: Melançon, G., Munzner, T., Weiskopf, D. (eds.) Eurographics/IEEE-VGTC Symposium on Visualization, vol. 29, pp. 973–982 (2010). doi:10.1111/j.1467-8659.2009.01684.x

Stef Busking received both B.Sc. (2005) and M.Sc. degrees (2006) in Computer Science from the Technische Universiteit Eindhoven. He is currently a Ph.D. candidate at Delft University of Technology. His research interests include illustrative rendering, real-time interaction, comparative visualization and visual exploration of data sets.

Charl P. Botha is Assistant Professor in the Visualization Group of the Delft University of Technology, the Netherlands, where he leads the Medical Visualization research activities. He is also a visiting scientist at the LKEB research group, Department of Radiology, Leiden University Medical Center. Dr. Botha holds an M.Sc. in Electronic Engineering (1999) and a Ph.D. in Computer Science (2005).

Luca Ferrarini works as a researcher in the field of neuroimaging/neuroscience at the Division of Image Processing (LKEB) of the Leiden University Medical Center (LUMC), in Leiden, the Netherlands. He received his Ph.D. cum laude in neuroimaging from Leiden University in 2008, and his M.Sc. cum laude in Information Engineering from the University of Modena and Reggio Emilia (Italy) in 2003. His research interests include statistical analysis of morphological changes in the brain due to neurodegenerative diseases, as depicted by structural MRI, and the characterization of functional connectivity networks in the resting brain using graph theory and resting state fMRI.

Julien Milles is Assistant Professor in the Department of Radiology of the Leiden University Medical Center, the Netherlands, where he works since 2004. He previously worked as a research associate at the Cardiovascular Research Institute of Maastricht (2003) and the CREATIS laboratory (Lyon, France) between 1999 and 2003. He received his M.Sc. degree in Electrical Engineering from the Polytechnic Institute of Grenoble and Ph.D. from the Applied Science Institute of Lyon in 1999 and 2002 respectively. His research interests include cardiovascular and neurological MR imaging, image enhancement, motion estimation, and quantitative analysis.



Frits H. Post is Associate Professor of Computer Science at TU Delft, where he leads a research group in data visualization. He received an M.Sc. degree in Industrial Design Engineering from TU Delft in 1979. His research interests include flow visualization, medical visualization, virtual reality and 3D interaction. He has (co-)authored more than 100 publications in many areas of data visualization. He is the chairman of the Eurographics Steering Committee on Data Visualization and a co-founder of the annual joint Eurographics-IEEE EuroVis Symposium. He is a Fellow of the Eurographics Association.