Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

ALEXANDER WIEBEL, PHILIPP PREIS, FRANS M. VOS
UND HANS-CHRISTIAN HEGE

Computation and Application of 3D Strokes on Visible Structures in Direct Volume Rendering

ZIB-Report 13-20 (April 2013)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Computation and Application of 3D Strokes on Visible Structures in Direct Volume Rendering

Alexander Wiebel[1,4], Philipp Preis[2], Frans M. Vos[3], and Hans-Christian Hege[1]

[1]Zuse Institute Berlin (ZIB), Berlin, Germany
[2]Freie Universität, Berlin, Germany
[3]TU Delft, Netherlands
[4]Coburg University of Applied Sciences, Germany

April 21, 2013

Abstract

In this paper we describe VisiTrace, a novel technique to draw 3D lines in 3D volume rendered images. It allows to draw strokes in the 2D space of the screen to produce 3D lines that run on top or in the center of structures actually visible in the volume rendering. It can handle structures that only shortly occlude the structure that has been visible at the starting point of the stroke and is able to ignore such structures. For this purpose a shortest path algorithm finding the optimal curve in a specially designed graph data structure is employed. We demonstrate the usefulness of the technique by applying it to MRI data from medicine and engineering, and show how the method can be used to mark or analyze structures in the example data sets, and to automatically obtain good views toward the selected structures.

1 Introduction

Painting-type tools [LSS09] belong to the standard tools in image editing. Strokes and curves are used to directly draw in the image and, even more interestingly, to mark or select locations where certain image processing techniques are intended to be applied. Occasionally, the drawn lines are also used measure area or length of certain image structures. All these and even more applications of the painted strokes or points are also of interest for the direct visualization and analysis of volumetric data: drawing translates to changing the transfer function [GMY11]), selecting can be translated to 3D segmentation [ONI05]), and measurement is eventually translated into obtaining characteristics of a 3D segmentation.

In the 2D setting of image editing, obtaining the strokes and points from user interactions is straight forward: the pixel coordinates belonging to the interactions can be easily transformed into coordinates relative to the image. When dealing with volumetric data the transformation is complicated because of the additional third dimension. If a user draws a stroke in the 2D screen space it is usually not obvious at which depth this line should run through the volume. This problem is further complicated by the application of a transfer function: should the depth of the line be determined by considering the original data or the optical properties assigned by the transfer function?

For single points, solutions for both variants, i.e. using the original data respectively using the visible rendering, have been conceived recently, e.g. [KBKG09] respectively [WVFH12]. For lines or strokes, the previous work has mainly focused on considering the original data, e.g. [ONI05] [DR12].

In this paper, we introduce an effective approach to obtain 3D lines from 2D strokes by considering the render-

ing instead of the original data. In other words, the presented technique allows to draw 3D lines on visible structures in direct volume rendering (DVR). If desired by the user, the technique is optionally able the loosen the visible constraint in order to ignore shortly occluding structures in favor of a smoother resulting 3D curve. We refer to the overall method as VisiTrace because the 3D curves can be used to trace visible structures.

2 Related Work

As our approach is designed to select 3D structures from 2D screen locations it is slightly related to volume picking. However, most volume picking techniques select only single points whereas VisiTrace aims at line structures. Thus we do not discuss the single point volume picking approaches here and refer the interested reader to the references in the papers that inspired our work [WVFH12], [AA09].

Owada et al. [ONI05] present a volume segmentation technique that uses 2D strokes on direct volume rendering images to generate 3D constraints for a segmentation algorithm. The step from the 2D stroke to the constraints is carried out by converting the 2D stroke into a 3D path that fits best to silhouettes in the data behind the 2D stroke. The constraints are then generated as offset points from the 3D path. In a later publication [ONI*08], Owada et al. add 2D preprocessing for improving the expressiveness of the strokes but they stick to the same technique for approximating the 3D path. The main differences of their approach and our work in the present paper is that their 3D path is not tailored to lie on the most visible structures but rather at borders of structures in the data. Furthermore, where the volume rendering alters the opacity using transfer functions, they do not consider the local contribution to the final pixel color but the local opacity which does not necessarily correspond to "visually distinct structures" [ONI*08]. Another technique aiming at segmenting structures in volumetric data by painting brushes in 2D has been introduced by Wan et al. [WOCH12]. Their approach is specifically designed for confocal microscopy data where interesting structures are highlighted by staining. The approach is not universally applicable for other types of data. Chen et al. [CSS08] for interactive volume sculpting using stroke input. However, the paper demonstrates the technique only for volume renderings exhibiting only completely opaque surface structures. The Volume Cutout technique by Yuan et al. [YZNC05] also segments 3D structures using 2D sketches, but the sketches remain completely 2D in their approach. Yu et al. [YEII12] introduce a technique selecting regions in 3D point clouds using sketched lassos.

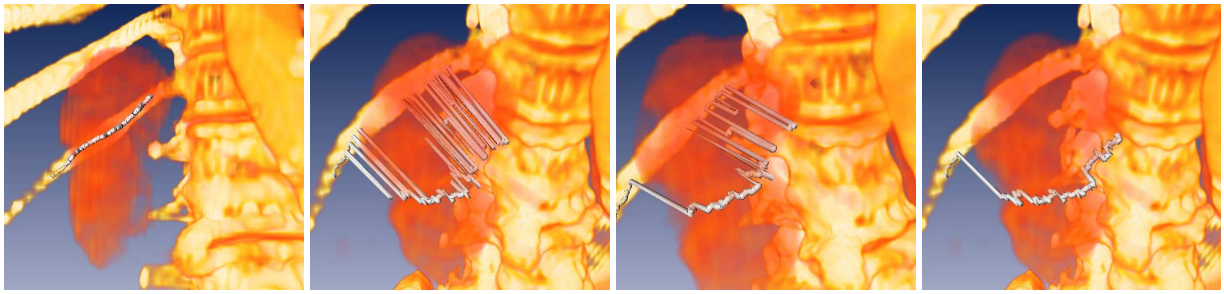
In the context of vessel segmentation, Diepenbrock and Ropinski [DR12] described a technique to trace a centerline in a rendering of a vessel by simply drawing a line on top of the vessel. Parts of their approach are inspired by Owada et al. [ONI05]. They use a path with lowest cost together with an active contour approach to detect the correct intended 3D line of interest. Unfortunately, the description of the weights (or costs) used for the lowest cost path remains vague. Another technique using a lowest cost paths to obtain skeletons of 3D structures has been presented by Abeysinghe and Ju [AJ09]. Their method however is completely data-based (In their case this means intensity-based). They do not consider any volume rendering.

Kohlmann et al. [KBKG09] use their contextual 3D picking technique also to allow the user to draw lines in or on top of volumetric structures. Their technique is not based on visibility consideration but on meta-data and thus can yield completely invisible paths. They cope with shortly occluded objects using the available meta data and following only the type of the structure that has been selected by the contextual picking where the stroke started.

Strokes and other interaction metaphors known from 2D image editors have not only been used to select and/or segment structures in 3D rendering but also to change the transfer function and thus the rendering itself. The most recent approach in this direction, called WYSIWYG Volume rendering, has been conceived by Guo et al. [GMY11].

3 Motivation

The work presented in this paper is motivated by two different types of interaction techniques that we already mentioned in the introduction: picking *visible* structures in images generated by direct volume rendering [WVFH12] and selecting *line* structures in volumetric



(a) (b) (c) (d)

Figure 1: Volume rendering of CT data set. (a) Line, as drawn onto the DVR image in two dimensions. (b)-(d) Other perspective showing the 3D nature of the resulting line. (b) 3D line using WYSIWYP directly. (c) 3D line using WYSIWYP with median filter. (d) 3D line using VisiTrace.

data [ONI05] [DR12]. Both techniques are well justified and good implementations exist. Yet there is no literature describing a combination of both techniques, i.e. a technique for tracing *lines* lying in or on top of or in 3D structures *visible* in a direct volume rendering. The present paper aims at filling this gap by introducing such a technique ("VisiTrace") in Section 4.

One could be tempted to simply apply visibility oriented picking [WVFH12] for each of the positions along a stroke drawn by the user. However, this can result in very jaggy lines (see Figure 1(b)) because structures touched by neighboring pixels might exhibit a very similar visibility. In such a case, the picked position will repeatedly jump between the two (or more) similarly visible structures. Application of a median filter to the 3D curve does not resolve this problem entirely (see Figure 1(c)). To avoid such jumping the presented technique considers more than one visible location for each pixel and tries to find the most plausible path along these locations.

When working with VisiTrace it becomes evident that in some cases, like shown in Figure 6, it is undesirable to strictly stick to the request for visibility. For these cases a modified version relaxing the visibility constraint, while still considering only the rendered image, is introduced in Section 5.

4 VisiTrace

In the following, we first give an overview of the VisiTrace method and describe the underlying algorithmic details afterwards. The overall approach consists of six main phases and one optional step:

1. The user draws a stroke on the DVR image using the mouse (Figures 1(a) and 6 top left). For all pixels hit by this drawing action a ray is cast in viewing direction.

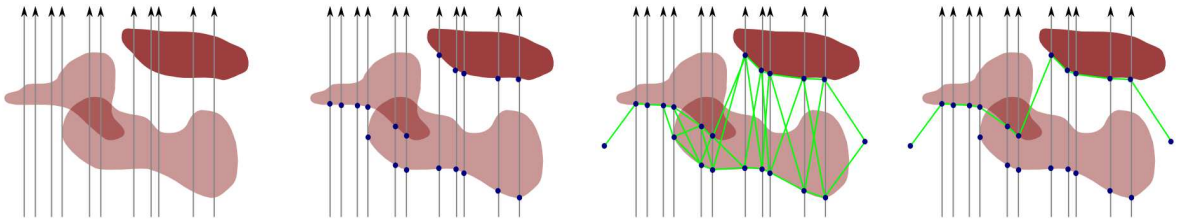
2. For each ray the WYSIWYP algorithm [WVFH12] is executed. The algorithm is modified in two regards:

   (a) Instead of only the highest jump of accumulated opacity, all detected intervals are considered (Figure 2(b)).

   (b) The locations corresponding to all jumps are stored.

3. A graph containing all stored locations is built: the locations are the nodes of the graph and the edges connect all locations belonging to one viewing ray with all locations belonging to the neighboring ray (Figure 2(c)).

4. Weights derived from the locations of the jumps and the corresponding opacity are assigned to the edges.



(a) Step 1 (b) Step 2 (c) Steps 3&4 (d) Steps 5&6

Figure 2: The steps of the VisiTrace algorithm. See Section 4 for a detailed description of the algorithm and its steps.

5. Dijkstra's algorithm [Dij59] is executed on the graph to find the shortest path that consists of exactly one node on each viewing ray.

6. The locations contained in the shortest path are connected to form a three-dimensional line strip (Figure 2(d)).

7. *Optional*: Global filters are applied as post-processing for the line strip.

We dedicate a separate subsection to each step of the algorithm steps.

4.1 Drawing

In this first step the only user interaction, namely the user drawing on top of the direct volume rendering image, is performed and recorded. The stroke generated by the user is recorded as array of 2D pixel coordinates returned by mouse events. Because the events coming of the mouse are not equidistant and continuous in the sense that the pixels form a connected line, we resample the line strip obtained by connecting the positions from the mouse events. The resulting pixel locations are either connected or equidistant (depending on an optional user-defined parameter). These 2D coordinates are transformed to world coordinates (3D). The resulting points $P^i, i \in \{0,\ldots,N-1\}$ are stored for further processing. For each point P^i a viewing ray R^i is traced through the volume (Figure 2(a)).

4.2 Detecting Visible Structures

With the viewing rays R_i at hand we extract the most visible volumetric structures along each ray. For this purpose we employ the visibility oriented picking approach described by Wiebel et al. [WVFH12]. While their algorithm extracts only the single most visible structure along the ray, we aim at having a relatively smooth 3D curve and thus are interested in considering also less visible structures. Hence, the locations $v^i_k, k \in \{1,\ldots,J\}$ of the J highest jumps of accumulated opacity along the ray R^i are stored (dots in Figure 2(b)). The locations are considered candidates for being members of the final 3D curve.

Together, steps 1 and 2 simply collect and store the location information needed for the following steps.

4.3 Graph Generation

Steps 3 to 6 try to find the best 3D line that runs through exactly one location v^i_k of each ray and contains as many "most visible" locations as possible without creating many spikes due to negligible noise in the volume rendering. The optimization is carried out by a shortest path algorithm on the specially defined graph. This graph is created in step 3 of the algorithm.

The graph consist of the v^i_k as nodes and directed edges $e^i_{j,k}$ connecting nodes v^i_j from ray R^i and v^{i+1}_k from R^{i+1} (see Figure 2(c)). The direction of the edges corresponds to the direction of the originally drawn stroke. A path along the directed edges will represent the final 3D curve. To provide an equally probable origin and target for the shortest path we add an artificial start node as well as an artificial end node. The start node is connected to all nodes v^0_k of the first ray, and all nodes v^{N-1}_k of the last ray

have edges connecting them to the artificial end node.

The size of the graph depends on the number of rays N and on the average number of jumps per ray J. Thus the number of nodes is $N \cdot J + 2$ and the number of edges is $2 \cdot J + (N-1) \cdot J^2$. The overall complexity of the graph thus is $\mathcal{O}(NJ^2)$. As the number of steps along the longest ray in the DVR is a sensible constant limit for J, the complexity of the graph for a certain rendering can be considered to be $\mathcal{O}(N)$.

4.4 Edge Weights

To obtain the final 3D curve from the graph described in Section 4.3 we employ a shortest path algorithm which we describe in Section 4.5. A clever choice of the weights assigned to the edges in step 4 is crucial for the shortest path algorithm's result. After thorough investigation we decided for weights that are a combination of the spatial location of the selected nodes and the magnitude of the corresponding jump in accumulated opacity. The actual weight function and the reasoning for its design are described in the following.

Our reasoning is based on a number of requirements for the behavior of the final 3D curve. We derive these requirements from the abstract yet basic application scenarios show in Figures 3(a)-(d).

Requirement 1 *If two path variants, differ in the magnitude of their opacity jumps, but are equivalent otherwise, the one with the higher jumps should be chosen (it is more visible and thus probably intended by the user). (Figure 3(a))*

There is no reason to change the structure early or late apart from the varying opacity jump magnitude.

Requirement 2 *If the opacity jumps mentioned in Requirement 1 are similar, both path variants should be equally likely.*

Requirement 3 *Avoidable changes of the structure the path runs on should only be performed if the opacity jump of the continued structure is much smaller than the one of the newly appearing structure. (Figure 3(b))*

Here we assume that a user usually wants to select continuous structures.

Requirement 4 *Changing the structure should be less likely than staying on it (when opacity values are equal). For this reason, the spatial position of the nodes must be considered.*

Thus, two structures having the same opacity jumps can be weighted differently in order to achieve a "smoother" path.

Requirement 5 *The weight of straight structures should be very low to avoid devaluating the cost of a structure change.*

Requirement 6 *If a structure change is unambiguous, it should not be weighted depending on the depth difference of the two structures. (Figure 3(c))*

It would be desirable that the selected structure in Figure 3(c) only depends on the opacity jumps because it unambiguously specifies what is visible here.

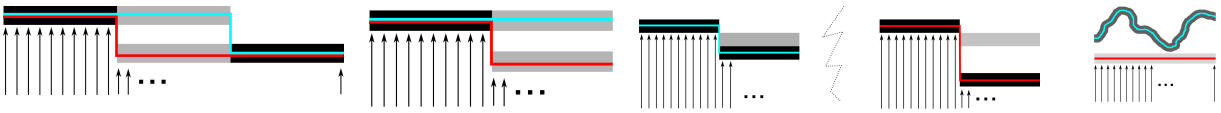
Requirement 7 *The weight of slightly bent structures should not be much higher than that of a straight structure because a followed structure might be oblique in space and might thus cause a longer path without an actual structure change. (Figure 3(d))*

4.4.1 Opacity Weighting

Above, we mentioned that it is desirable to let the magnitude of opacity jumps for candidate locations have an influence on the final weight of an edge (Requirements 1-3). For an increasing opacity jump magnitude, the goal is an increased probability for an edge to be in the shortest path and thus a decreasing edge weight. Therefore, and because the magnitude of opacity jumps is in $[0,1]$, we set

$$w_\alpha(v_j^i, v_k^{i+1}) := 1 - (\text{magnitude of opacity jump } v_k^{i+1}).$$

The opacity of the previous jump v_j^i has already been incorporated in the weight of the preceding edge and is thus neglected here. A further modification of the weight does not seem necessary because the opacity influence of WYSIWYP working in the same way has proven to be effective.



(a) Unavoidable change (b) Avoidable change (c) Equal treatment (d) Curved structure

Figure 3: Basic scenarios for structure selection in case of partly occluding semi-transparent structures. Brightness of structures indicates magnitude of corresponding opacity jump (black $\widehat{=}$ high, light grey $\widehat{=}$ low). Blue and red lines indicate possible variants for resulting curve.

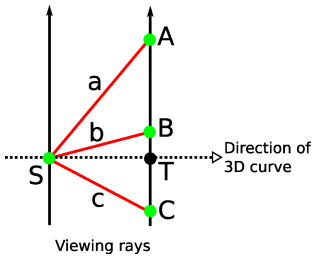


Figure 4: Illustration of distance types for edge weighting.

4.4.2 Distance Weighting

Requirements 4-7 imply that the edge weights should also depend on the relative positions of the the graph nodes. We identified two possible ways to incorporate the relative positions into the weight function. Both can be described best by using Figure 4. The first way is simply using the Euclidean distance between the nodes, i.e. the length of the geometric representation of the edge connecting the nodes. In Figure 4 this corresponds to the lengths of $\overline{SA} \widehat{=} a$, $\overline{SB} \widehat{=} b$ or $\overline{SC} \widehat{=} c$. The second way is to use changes in the distance between the node and the camera. This correspond to the distances $\overline{TA}$, $\overline{TB}$ or $\overline{TC}$ in Figure 4.

We choose the Euclidean distance because its rate of growth (when stepping away from T) is smaller in the beginning (Requirement 7). Furthermore, it assigns sufficiently small weights to very small distances. As the rays are very close because of the resampling, the fact that the Euclidean distance also depends on the distance between the rays does not violate Requirement 5.

To obtain the weights from the Euclidean distances, we normalize all distances to lie in $[0,1]$ by dividing by the overall maximum distance:

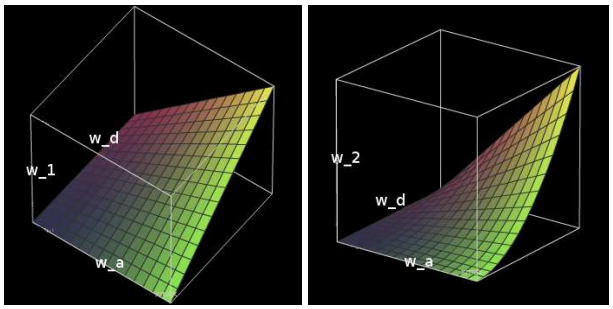
$$w_d := \frac{\|v_j^i, v_k^{i+1}\|}{\max\limits_{r,s,t} \|v_s^r, v_t^{r+1}\|}$$

The weight function w_d has been devised in an effort to fulfill most requirements related to distance. However, in favor of an intuitive weight function we decided to violate Requirement 6. Functions fulfilling Requirement 6 by the introduction of special cases appeared to be unreliable.

4.4.3 Combined Weight

The partial weights incorporating spatial distance and opacity jumps have been described above. They can be combined to form the final weight function $w(v_j^i, v_k^{i+1})$ now. In summary, the combined function should have the following properties:

- Increasing opacity jump magnitude should imply increased probability for the jump location to be in the final path. Thus
  - w is strictly increasing with w_α
  - $w_\alpha \approx 0 \Rightarrow w \approx 0$, so that completely opaque structures are selected with very high probability
- Large distances between jump locations should increase w
  - w is monotonically increasing with w_d
  - $w_d \approx 0 \Rightarrow w \approx 0$, so that close jump locations are chosen with very high probability



(a) $w = w_1$ (b) w_2

Figure 5: Plots of combined weight functions. Small differences in distance (which might result from an oblique structure) produce a smaller weight in w_2.

A straight forward choice fulfilling these criteria would be

$$w_1 := w_\alpha \cdot w_d.$$

This function works well in many cases but leaves room for improvements regarding Requirements 5 and 7. In order to give very small differences in distance (which might result from an oblique structure) an even lower influence on the final path we use

$$w_2 := w_\alpha \cdot {w_d}^2.$$

In our experiments this function yielded the best results. A visual comparison of w_1 and $w = w_2$ is provided in Figure 5.

4.5 Shortest Path

In the fifth step Dijkstra's shortest path algorithm [Dij59] is applied to the graph equipped with edge weights described above. The algorithm searches for a path between the artificial start node and the artificial end node. Because the graph contains $\mathcal{O}(N)$ edges, a time complexity of $\mathcal{O}(N \log N)$ can be achieved using an implementation based on a Fibonacci heap [FT84].

4.6 3D Curve

The shortest path obtained in the previous step is easily translated into the final 3D curve. The artificial start and end nodes are removed from the obtained path as they do not correspond to any spatial location. The spatial locations, that is, the jump locations, of the remaining nodes form the final 3D curve. We simply connect them by straight lines. However, one could also fit a smooth curve through these positions to beautify the curve rendering. The latter should, however, be performed after the optional 7th step.

4.7 Optional Filtering

Step six already establishes the desired 3D curve. If users, however, would like to have a smoother curve than can be obtained by the previous steps, we provide the possibility to filter the 3D curve. We use a nine point median filter moving along the curve. This will result in a smoother curve but can deviate from the actually rendered structures to some extent.

5 VisiTrace Extended

The algorithm as described up to this point achieves the goal of tracing lines on top of visible structures (Figure 6, top). In some cases, however, the user might be interested in tracing a line that stays on the object where it started as long as possible (Figure 6, lower right) and ignore crossing objects. This is definitely not possible when applying WYSIWYP to each ray separately because a crossing object, as in the top right image of Figure 6, might be the most visible object and would thus always be selected by WYSIWYP. VisiTrace, using the shortest path approach, has similar problems when the crossing object is completely opaque. Only one candidate will be found along the ray in this case (see Figure 7 upper right). The desired result, shown in the lower right image of Figure 6 and the lower row of Figure 7, can be achieved by using VisiTrace with down-scaled opacities along the viewing ray. This down-scaling, achieved by dividing the opacity values of all samples by the maximum number of samples, makes all structures transparent to some degree, leading to candidate points also behind originally opaque structures. In other words the down-scaling allows the picking algorithm to look through the originally completely opaque crossing structure (see Figure 8 and Figure 7 lower right). The resulting opacities do not correspond to the actually perceived opacities anymore, but lead to the desired 3D

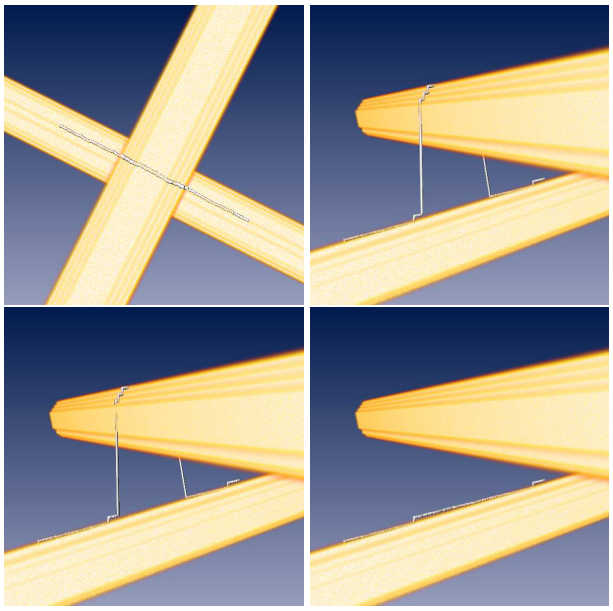


Figure 6: From top left to bottom right: View at 3D line from direction from which 2D line was drawn on screen; other perspectives showing 3D lines resulting from 2D line by direct picking, VisiTrace, and VisiTrace with alpha scaling.

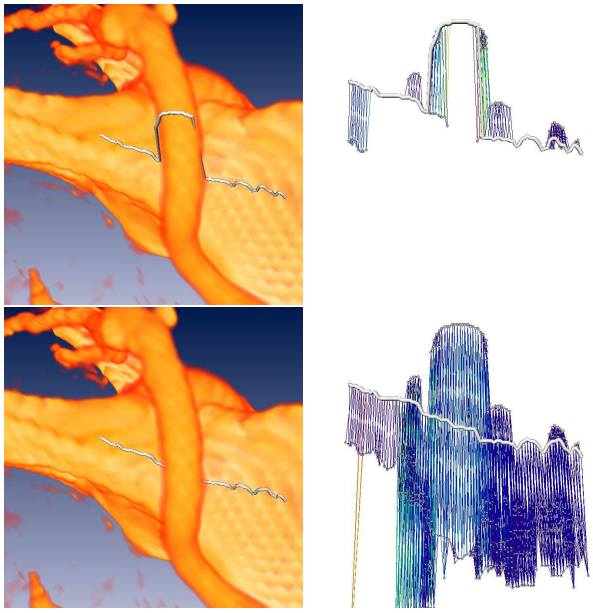


Figure 7: Left column selected curve, right column corresponding graph representation (weights color coded). Top row without opacity scaling, lower row with opacity scaling.

curve which intentionally does not run only on the visible structures. Figure 7 illustrates the generated graphs for VisiTrace with and without opacity scaling. It is obvious that the relatively smooth line in the lower left image is not achievable using the graph for the unscaled opacities in the upper right image.

We expect the user to point intentionally on visible structure first. Thus we deactivate down-scaling for the first position to get the actually visible depth there. In practice, this ensures that we obtain the desired 3D curve.

We note that we also evaluated the opacity peeling technique by Rezk-Salama [RSK06] to handle the shortly occluding structures. While opacity peeling is very useful for making non-visible parts of volume renderings visible, employing it for VisiTrace failed because it was not possible to choose a generally appropriate value for the technique's lower bound `T_low`.

The usual result of VisiTrace is a 3D line running on top of certain structures of the direct volume rendering. If the center of the jump intervals is used instead of the first position, VisiTrace also allows to trace lines in visible volumetric structures. This can be especially interesting for tubular structures.

6 VisiTrace Applications

The 3D curve resulting from the VisiTrace algorithm can be used in various ways. In general, its use only depends on the application context of the direct volume rendering it is applied to. We picked out some applications which can be of interest for a wide range of scientific and medical visualization tasks and describe them in the following.

The simplest and most obvious application is to mark visible structures in the rendering. This can be useful for exploring the data as well as for discussions between users working together. Due to this application's simplicity we will not provide a deeper discussion here. More elaborate applications will be discussed instead.

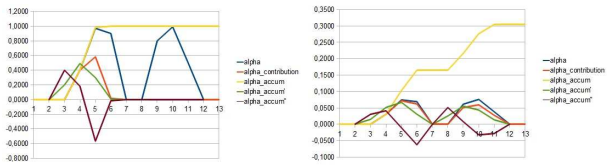


Figure 8: Illustration of the effect of opacity scaling on the accumulated opacity. Original on the left, scaled on the right. In the scaled case both peaks of the local opacity are reflected in the accumulated opacity.

6.1 Automatic Viewpoint Selection Using VisiTrace

An automatic selection of good view points on specific structures in DVRs can aid fast navigation and data exploration. A number of different techniques to determine good views has been presented in the past. Such techniques use different measures to determine the view points. Examples of the used measures are

- a single 3D location together with context information [KBKG07],
- information theoretic measures for viewpoint "goodness" [BS05], [VMN08],
- high-dimensional feature clusters based on gradient variation [ZAM11],
- high-intensity features and their distribution (using principal component analysis, PCA) [KUBS12],
- and topological features of the original scalar field [TFTN05].

In contrast to these methods, we propose to use the 3D curve obtained by VisiTrace to mark the desired features in the volume rendering and then compute a viewpoint providing the best view toward the 3D curve. We design this best view to have two balanced properties: the curve should be occluded by the DVR as little as possible, and the structure of the curve should be perceivable as good as possible.

In order to determine how strongly the curve is occluded if viewed from possible view points, we perform the following procedure for candidate points can_i, $i \in \{0,\ldots,C-1\}$ equally distributed on a sphere [1] surrounding the whole 3D curve. A ray $canR_i$ is cast from the current candidate point can_i to every point on the curve. For each of these rays the opacity is accumulated using the same sampling distance as the direct volume rendering. The accumulated opacities for all these rays is summed up. The sum represents the visibility of the curve.

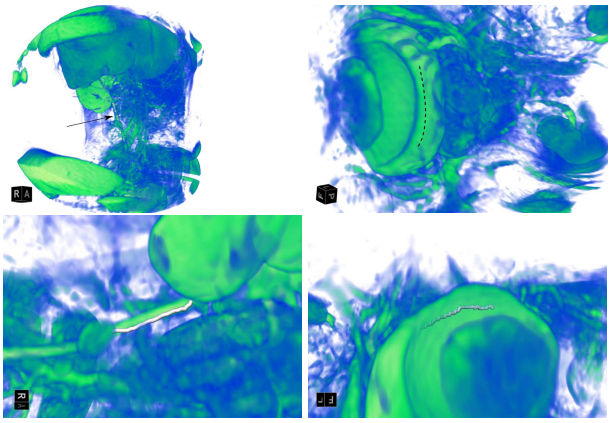


Figure 9: Upper row: Original perspective when stroke was drawn. Lower row: Corresponding reoriented perspective providing best view. In left column a line on top of the structure is selected, in right column a line in the center of the structure is selected. The dashed line shows the approximate shape of 3D curve that is hidden because it is in the center of the structure. In the corresponding reoriented view the curve is visible because it is viewed from the side and thus less occluded.

Choosing the candidate point with the lowest sum of accumulated opacities, satisfying results can be achieved in most cases. However, in some cases the direction yielding least occlusion may not be optimal. Consider a relatively straight curve. If the best viewing direction would be nearly parallel to this curve than the selected view would not provide much insight because the curve would cover only a very small part of the screen space. In order to avoid this, we weight the summed opacities of the view points using the overall shape of the curves. The shape of a curve can be characterized by the eigenvalues

[1] We choose the radius of the sphere to be two times the diagonal of the bounding box of the curve and the center of the sphere to lie in the center of the bounding box.

$e_1 \geq e_2 \geq e_3$ and eigen vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ obtained using a PCA of the positions representing the curve. The relations of the eigenvalues can be used to differentiate between three different shapes [WPG*97]: linear, planar (e.g. a spiralling 3D curve lying on a flat surface) or spherical (e.g. circular *jumpy* 3D curve). Three parameters c_l, c_p and c_s can be obtained. The largest of these parameters indicates the approximate shape of the curve. For each of the three shape types a different weighting is used:

Linear The summed opacities of the candidate directions can_i are weighted by the dot product[2] $can_i \cdot \mathbf{e}_1$. This disqualifies viewing directions (nearly) parallel to the curve.

Planar The summed opacities of candidate directions can_i are weighted by $(1 - (can_i \cdot \mathbf{e}_3))$. This promotes directions pointing toward the "flat" part of the shape.

Spherical All directions are weighted equally.

Note that low weights promote the corresponding direction because the direction with lowest summed opacities is chosen as best viewing direction. For the final view the camera is located at the candidate position on the sphere corresponding to the best viewing direction and is looking toward the center of the sphere.

As an alternative to using the best viewing angle, one can choose the viewing angle (out of the best, e.g., 5 viewing angles) lying closest to the initial viewing direction. This maintains a certain kind of *view coherence*.

6.2 Distance Measurements Using VisiTrace

The standard interaction mode of VisiTrace is free-hand sketching. In this this mode it is easy to trace structures of arbitrary shape. In contrast, drawing straight lines is very tedious in this mode. Nevertheless, straight lines have an important application. They can be used to measure *straight distances* on top of or in visible 3D structures.

In order to allow users to measure such distances we provide a second interaction mode. In this mode the user specifies two points in screen space and the system interpolates samples P^i on a straight line between these points. The samples P^i are then processed by the usual VisiTrace pipeline. The resulting 3D curve is straight in the original perspective but follows the visible 3D object. Summing up the lengths of the line segments making up the 3D curve, or fitting a spline through the 3D positions and computing its length, the desired *straight distance* measurements can be achieved.

[2] Only normalized vectors are considered and the norm of the dot product is used. Thus the weights lie in $[0, 1]$.

6.3 Further Applications

As mentioned in the beginning of this section more applications can be envisioned. Examples of such applications, which we might address in future work, are the selection of cutting planes best fitting the curve or even a curved planar reformation [KFW*02] based on the VisiTrace result.

7 Results

In this section, we summarize the types of interaction and selection in volume renderings that can be achieved by VisiTrace, its extension using opacity scaling, and the proposed automatic view selection.

Using the shortest path search in the VisiTrace graph, one can achieve plausible 3D curves running on top of relatively transparent volumetric regions. This can be shown by applying the algorithm to the human computed tomography data mentioned as being problematic in the Motivation section. The result is depicted in Figure 1(d). Instead of repeatedly jumping between the organ in the foreground and the bone in the background as in Figures 1(b) and (c), the 3D curve stays on top of the organ structure. More examples for this effect are shown during the discussion of the extensions below and in the video accompanying this article.

VisiTrace alone can only achieve the just described effect. If users are interested in ignoring some foreground structures they need to activate opacity scaling. Figures 6 and 7 illustrate how the scaling enables the user to select continuous structures shortly being in the background. In Figure 6 a synthetic test data set is used to highlight the effect. The volume rendering of a real world magnetic resonance data set shown in Figure 7 support the findings of Figure 6.

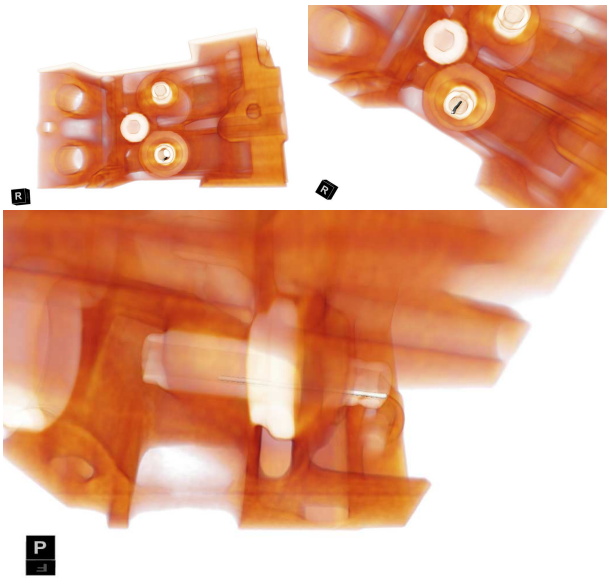


Figure 10: Top: Original perspective and opacity-based reorientation of DVR of engine data set. Bottom: Reorientation based on opacity and direction weighting.

Figure 9 demonstrates the usefulness of the view selection approach. In the left column the front of a visible structure has been selected by a stroke and the view has been reoriented to provide a view toward the structure and the 3D curve on top of it. The reorientation of the view is even more useful when selecting curves *inside*, i.e. in the center, of visible structures. This has been done in the upper image of the right column. The original stroke is illustrated using the dashed line. The 3D curve itself is not visible from this view point because it lies inside the relatively opaque structure. Reorienting the view as in the lower image of the same column, reveals the the location of the 3D curve. From this automatically found view point, the curve is visible because it is seen from the side of the structure where only a very thin layer covers the curve.

The effect of the shape-based weighting is shown in Figure 10. Although the curve is least occluded when looking into the tube like engine part (top right image), our method chooses a different viewing direction (lower image). In the upper right image, where only the opacity is considered, the shape of the curve is hardly perceivable because the viewing direction is nearly parallel to the curve. Using the direction weighting the lower image can be obtained. In this image the shape of the curve as well as the curve itself are visible.

7.1 Timings

In the video accompanying this article, we demonstrate the interactive nature of our approach. If we use VisiTrace without opacity scaling the final 3D curve is always available almost instantly, that is in less than a second. With activated opacity scaling producing more candidate points per ray, long strokes can lead to a comparatively large graph and thus to a longer computation for obtaining the shortest path through the graph. Nevertheless, computing the 3D curve is still faster than carefully drawing the original stroke, that is in the order of a few seconds.

8 Conclusion

In this paper, we have introduced a new way of interacting with direct volume rendering. The presented approach allows to draw lines and obtain corresponding 3D curves based on features visible in the rendering. The 3D curves are extracted in *soft* real-time and can thus be integrated into an every-day work-flow dealing with DVR. We discussed a number of applications of the 3D curves. In this course we described a way to automatically generate good view points for observing the curves and the corresponding structures. As the presented method is focused on *visible* structures it is naturally and intentionally dependent on what is visible, i.e. on the chosen transfer function and the viewing direction when drawing the initial 2D stroke. A short video accompanying this paper demonstrates the method in action.

8.1 Future Work

This work extends an approach by Wiebel et al. [WVFH12] by allowing to draw strokes, i.e. one-dimensional structures, instead of selecting points, i.e. zero-dimensional structures, on visible features of direct volume renderings. We plan to increase the dimension even further and obtain surfaces adapting to the visible 3D structures like a plaster cast. This will probably lead

to a formulation similar to work by Grady [Gra06]. We are currently working on creating such surfaces from closed strokes or user drawn *brushes* in the screen space.

Drawing lines with the mouse can be imprecise and cause fatigue. We plan to extend our implementation to support touch screens and digital table input devices as is done other work [YEII12], [WOCH12]. This extension is straight forward because VisiTrace only takes the pixel positions as input and is thus independent from the actual input device.

Acknowledgments

The basics of the methods presented here have been developed in the course of a master's thesis [Pre12]. The volvis.org data repository has been very useful for evaluating our work.

References

[AA09] ARGELAGUET F., ANDUJAR C.: Efficient 3d pointing selection in cluttered virtual environments. *IEEE Computer Graphics & Applications 29*, 6 (Nov. 2009), 34–43.

[AJ09] ABEYSINGHE S. S., JU T.: Interactive skeletonization of intensity volumes. *Vis. Comput. 25*, 5-7 (Apr. 2009), 627–635.

[BS05] BORDOLOI U. D., SHEN H.-W.: View selection for volume rendering. In *IEEE Visualization Conference* (2005), IEEE Computer Society, pp. 487–494.

[CSS08] CHEN H.-L. J., SAMAVATI F. F., SOUSA M. C.: Gpu-based point radiation for interactive volume sculpting and segmentation. *The Visual Computer 24*, 7-9 (2008), 689–698.

[Dij59] DIJKSTRA E.: A note on two problems in connexion with graphs. *Numerische Mathematik 1* (1959), 269–271.

[DR12] DIEPENBROCK S., ROPINSKI T.: From Imprecise User Input to Precise Vessel Segmentations. In *Eurographics Workshop on Visual Computing for Biology and Medicine* (Norrköping, Sweden, 2012), Eurographics Association, pp. 65–72.

[FT84] FREDMAN M., TARJAN R.: Fibonacci heaps and their uses in improved network optimization algorithms. *Foundations of Computer Science, IEEE Annual Symposium on 0* (1984), 338–346.

[GMY11] GUO H., MAO N., YUAN X.: WYSIWYG (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec. 2011), 2106–2114.

[Gra06] GRADY L.: Computing exact discrete minimal surfaces: Extending and solving the shortest path problem in 3D with application to segmentation. In *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition* (2006), pp. 69–78.

[KBKG07] KOHLMANN P., BRUCKNER S., KANITSAR A., GRÖLLER M. E.: LiveSync: Deformed viewing spheres for knowledge-based navigation. *IEEE Trans. on Visualization and Computer Graphics 13*, 6 (Oct. 2007), 1544–1551.

[KBKG09] KOHLMANN P., BRUCKNER S., KANITSAR A., GRÖLLER M. E.: Contextual picking of volumetric structures. In *Proceedings of the IEEE Pacific Visualization Symposium 2009* (May 2009), Eades P., Ertl T., Shen H.-W., (Eds.), IEEE Computer Society, pp. 185–192.

[KFW*02] KANITSAR A., FLEISCHMANN D., WEGENKITTL R., FELKEL P., GRÖLLER M. E.: CPR: Curved planar reformation. In *Proceedings of the conference on Visualization '02* (Washington, DC, USA, 2002), VIS '02, IEEE Computer Society, pp. 37–44.

[KUBS12] KIM H. S., UNAT D., BADEN S. B., SCHULZE J. P.: Interactive data-centric viewpoint selection. 829405–829405–12.

[LSS09] LIU J., SUN J., SHUM H.-Y.: Paint selection. *ACM Trans. Graph. 28*, 3 (July 2009), 69:1–69:7.

[ONI05] OWADA S., NIELSEN F., IGARASHI T.: Volume catcher. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), I3D '05, ACM, pp. 111–116.

[ONI*08] OWADA S., NIELSEN F., IGARASHI T., HARAGUCHI R., NAKAZAWA K.: Projection plane processing for sketch-based volume segmentation. In *ISBI* (2008), IEEE, pp. 117–120.

[Pre12] PREIS P.: *Tracing and Picking Visible Structures in Direct Volume Rendering Images With and Without Local Illumination*. Master's thesis, Freie Universität Berlin, 2012.

[RSK06] REZK-SALAMA C., KOLB A.: Opacity peeling for direct volume rendering. *Computer Graphics Forum 25*, 3 (2006), 597–606.

[TFTN05] TAKAHASHI S., FUJISHIRO I., TAKESHIMA Y., NISHITA T.: A feature-driven approach to locating optimal viewpoints for volume visualization. In *IEEE Visualization* (2005), IEEE Computer Society, pp. 495–502.

[VMN08] VÁZQUEZ P.-P., MONCLÚS E., NAVAZO I.: Representative views and paths for volume models. In *Proceedings of the 9th international symposium on Smart Graphics* (Berlin, Heidelberg, 2008), SG '08, Springer-Verlag, pp. 106–117.

[WOCH12] WAN Y., OTSUNA H., CHIEN C.-B., HANSEN C.: Interactive extraction of neural structures with user-guided morphological diffusion. In *Proceedings of IEEE Symposium on Biological Data Visualization (BioVis'12)* (2012).

[WPG*97] WESTIN C.-F., PELED S., GUDBJARTSSON H., KIKINIS R., JOLESZ F. A.: Geometrical diffusion measures for MRI from tensor basis analysis. In *ISMRM '97* (Vancouver Canada, April 1997), p. 1742.

[WVFH12] WIEBEL A., VOS F. M., FOERSTER D., HEGE H.-C.: WYSIWYP: What you see is what you pick. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (Dec. 2012).

[YEII12] YU L., EFSTATHIOU K., ISENBERG P., ISENBERG T.: Efficient structure-aware selection techniques for 3D point cloud visualizations with 2DOF input. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Scientific Visualization / Information Visualization 2012) 18*, 12 (Dec. 2012).

[YZNC05] YUAN X., ZHANG N., NGUYEN M. X., CHEN B.: Volume cutout. *The Visual Computer (Special Issue of Pacific Graphics 2005) 21*, 8-10 (2005), 745–754.

[ZAM11] ZHENG Z., AHMED N., MUELLER K.: iView: A feature clustering framework for suggesting informative views in volume visualization. *IEEE Transactions on Visualization and Computer Graphics 17* (2011), 1959–1968.