Feature Peeling

Muhammad Muddassir Malik
Institute of Computer Graphics and Algorithms
Vienna University of Technology
mmm@cg.tuwien.ac.at

Torsten Möller
Graphics, Usability, and Visualization (GrUVi) Lab
Computing Science Department
Simon Fraser University
torsten@cs.sfu.ca

M. Eduard Gröller
Institute of Computer Graphics and Algorithms
Vienna University of Technology
groeller@cg.tuwien.ac.at

ABSTRACT

We present a novel rendering algorithm that analyses the ray profiles along the line of sight. The profiles are subdivided according to encountered peaks and valleys at so called transition points. The sensitivity of these transition points is calibrated via two thresholds. The slope threshold is based on the magnitude of a peak following a valley, while the peeling threshold measures the depth of the transition point relative to the neighboring rays. This technique separates the dataset into a number of feature layers. The user can scroll through the layers inspecting various features from the current view position. While our technique has been inspired by the opacity peeling approach, we demonstrate that we can reveal detectable features even in the third and fourth layers for both CT and MRI datasets.

CR Categories: I.3.7 [Computer Graphics]: Three dimensional graphics and realism

Keywords: feature peeling, volume visualization, ray analysis

1 INTRODUCTION

Transfer functions are used in 3D visualization to assign user defined optical properties to a volumetric dataset based on scalar values. This specification of the optical properties should be able to highlight defective tissue or features that are of interest for a particular medical study. This is a non-trivial task and often requires considerable time and expertise to achieve desired results. While one might be able to set up a system which can be reused for several patients, this is not always possible.

Magnetic Resonance Imaging (MRI) datasets are more difficult to handle than and different from Computed Tomography (CT) datasets. Hounsfield numbers give a good and patient independent indication of the tissue type in CT. In contrast, the variance of tissue response between different patients in MRI datasets is too large to use pre-defined transfer functions for the detection of features. Transfer function specification has to be performed every time a new MRI dataset is generated. This fact makes transfer function specification a difficult and time-consuming task.

Additionally, MRI typically contains a considerable amount of noise that makes it harder and more challenging to produce insightful visualizations. There is high frequency noise that affects the clarity of the images and there is low frequency noise that slowly changes the intensity of a signal. 3D visualization techniques are rare for MRI datasets and often medical personnel use manual exploration of the datasets through slice-based inspection.

We propose a novel rendering technique that identifies interesting features along viewing rays based on a ray profile analysis. For each particular view point, the algorithm allows the user to browse through the layered features of the dataset (see section 3). This technique can be used without specifying any transfer function and henceforth is suitable for medical applications. Further, we include a de-noising step (as explained in section 3.1) to be able to deal with MRI datasets. We successfully apply this technique to a variety of medical and synthetic datasets (section 5).

This work aims at similar goals as the work of Rezk-Salama and Kolb [16]. We intend to show the entire dataset to the user, in a layered manner without putting effort in setting up a complicated transfer function. We detail the differences as well as the relationship to other work in the next section.

2 RELATED WORK

In volume rendering, transfer function specification is the main tool for the user to define optical properties. The transfer function guides the user to detect features in a volumetric dataset. The 1D transfer function is the simplest example which maps scalar values to opacity and color. More effective, but complex transfer functions that require user training and experience have been proposed.

A number of interesting enhancements, meant to give the user insight into the data, have been proposed for the 1D transfer function. Simple histograms [3], the contour spectrum [1] and Laplacian-weighted histograms [14] have been suggested. Potts and Möller [15] investigate the usage of a logarithmic scale that eases transfer function specification.

Multi-dimensional transfer functions [9] have been introduced which assign optical properties based on data values but also first and second derivative information thereof. Kindlmann and Durkin [8] automate the generation of a transfer function. They use the relationship between scalar values and their first and second order derivative to highlight boundaries in the dataset.

Kniss et al. [10] describe how probing and clipping can enhance the understanding of the features that exist in a dataset. Bergner et al. [2] use a spectral representation of colors instead of the RGB model to enhance details and allow effective exploration of datasets. Transfer function specification is also tailored specifically to the visualization of medical datasets by making use of metadata [4].

Gao and Shen [6] efficiently extract isosurfaces for a given viewpoint. They divide the dataset into spherical partitions and store these in a binary tree designed for fast access. They propose a method for extracting a single isosurface and thus their technique is not generally applicable to CT and MRI based medical datasets.

Höhne et al. [7] extend the Marr-Hildreth segmentation operator to 3D. They apply their technique in a view-independent manner. They require correction of errors through user input and produce a segmentation of the dataset as output. In our approach, we do not intend to perform segmentation but instead want to reveal features in a view-dependent manner. As we neither require human intervention nor any pre-processing of the entire dataset, our technique is interactive and we can control the level of peeling in real-time. A

detailed description of volume graphics techniques and their applicability to medical visualization is given by Engel et al. [5].

While most of these techniques require user intervention, it would be preferable to cut this step from the visualization process to provide a quick insight into the dataset. This is the idea of the opacity peeling approach of Rezk-Salama and Kolb [16]. It allows a layered browsing of the dataset. The layers are defined through accumulated opacity and basically all the information in the dataset is visible. As the layers are based entirely on visibility (as opposed to features), objects of interest might be split and distributed among several layers. Instead of peeling different layers of opacity, we propose to do an analysis of ray profiles and split the rays not according to opacity thresholds, but rather at possible feature transition points.

3 Feature Peeling

Along a ray in a dataset, we find valleys, peaks and homogeneous regions. Areas of interest are, by default, regions where the data field is changing. If there are detectable transitions from one tissue to another, then these transitions will be present in regions with changing data values. These transitions can be very useful and the transition points are the places where interesting features inside the dataset start or end. Whether we wish to look at objects occluded by others or want to search for any disorder inside a single organ, such transitions will help us explore and locate information fast and easily.

A ray cast through a medical dataset produces a ray profile based on the scalar values encountered. This ray profile is used by various techniques in a specific way to generate images. For example, Direct Volume Rendering (DVR) performs front to back or back to front compositing along the ray profile. Considering a linear ramp as a transfer function, peaks in the ray will contribute most to the volume rendering integral. Averaging on the other hand calculates an average of all the scalar values encountered by a ray, as is common in X-Ray rendering or Fourier volume rendering [12].

Maximum Intensity Projection (MIP) only displays the highest peak in a ray profile [11] [13] [17]. Similarly, Local Maximum Intensity Projection (LMIP) [18] searches for the first peak above a specified threshold along the ray.

In figure 1, a ray profile with three features is shown. These features are prominent density peaks in the ray profile as compared to the rest of the profile. Feature peeling is separating these features into different layers by locating transition points between them.

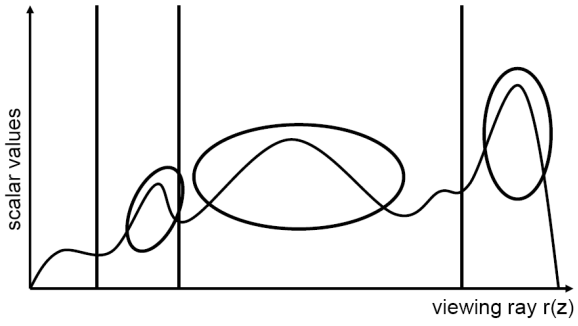


Figure 1: A ray profile showing three features as prominent density peaks. Features are marked with ovals and vertical lines show the transition points. Transition points split a ray profile into different layers.

We present a high level explanation of feature peeling in figure 2 by a 2D illustration. We use three concentric circular layers shown at the top left of the figure 2 as dataset. A ray is also shown, which passes through the center of the dataset. The ray profile and the transition points corresponding to the ray are drawn below the dataset. The first three transition points of the ray are depicted with a square, a rectangle and a circle respectively. These transition points mark the start of features. On the right we show the first three resulting layers from the current view point. The example in figure 2 shall illustrate the importance of finding transition points that are representative of the features in a dataset.

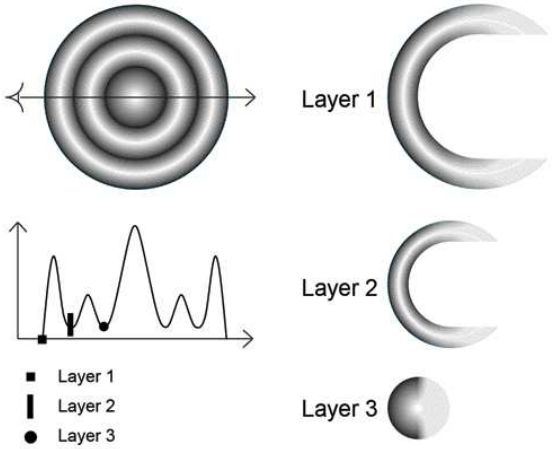


Figure 2: Three concentric layers of a dataset are shown at the top left of the figure. A ray is depicted in the dataset and the corresponding ray profile is drawn below. Small marks on the ray profile are shown which depict the transition points. On the right we show the first three layers that will be generated from this dataset as a result of feature peeling.

3.1 Locating Transition Points

The search for the transition points is based on the first and second derivatives of the ray profile. Local minima, i.e., locations along the ray where the first order derivative is zero and the second order derivative is positive are considered as the transition points.

The number of transition points can vary from ray to ray based on the number of features along a ray and the amount of noise present. In the case of MRI datasets, a large number of transition points might be present due to high frequency noise. We remove high frequency noise by using a low pass filter. This will smooth the profile, remove the transition points that exist because of noise, and enhance the transition points that are representative of the dataset.

Low frequency noise in the MRI datasets may also generate false transition points. These transition points are removed by calculating the slope between the transition point and the local maximum of the peak immediately following the transition point. If the slope is less than some user specified threshold, it is discarded and the search for another transition point is carried on from the local maximum onward. We call this threshold *slope-threshold* henceforth in this paper.

Figure 3 illustrates the usage of the *slope-threshold*. A local minimum and a local maximum are depicted as a circle and a square respectively. The dashed line shows the *slope-threshold* specified by the user. A dashed arrow depicts the calculated slope. This local minimum will not be considered a transition point as the slope between the local minimum and the local maximum is below the *slope-threshold*.

Figure 4(a) shows an original ray profile from an MRI head dataset. High frequency noise is visually recognizable. Transition points for this ray profile are shown in figure 4(b) as vertical lines, calculated after removing high frequency noise. Arrows point at the

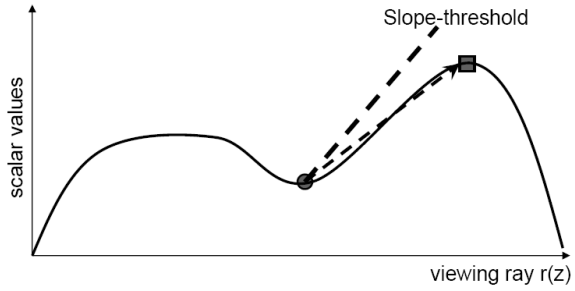


Figure 3: The grey circle shows the local minimum and the grey square shows the first local maximum after this local minimum. The dashed line depicts the *slope-threshold*. The slope between the local minimum and the local maximum is shown by a dashed arrow.

local minima that are the result of low frequency noise and which are discarded using the *slope-threshold*. Figure 4(c) shows a slice from the MRI dataset with a line indicating the ray whose profile is in figure 4(a) and 4(b). The *slope-threshold* was set to 1.0 (45 degrees) for generating figure 4(b).

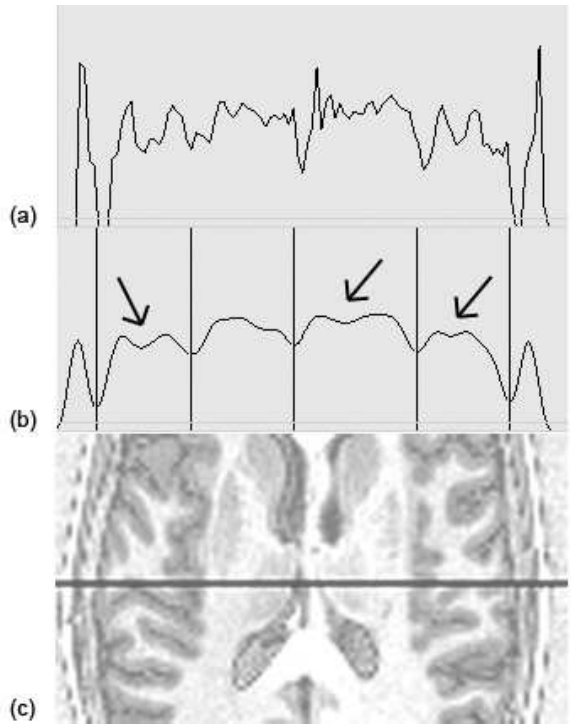


Figure 4: Original ray profile of an MRI dataset in (a) shows a lot of high frequency noise. In (b) the profile of (a) is filtered with a median low pass filter. Vertical lines depict the transition points and arrows indicate the local minima skipped based on the *slope-threshold*. (c) shows a slice from the MRI dataset and the horizontal line indicates the path of the ray.

3.2 Relevance Across Neighboring Rays

The transition points are generated with no input from neighboring ray profiles. This may subdivide the volume data into non-smooth feature layers because of the difference in depths of the transition points of neighboring rays. Figure 5 shows three neighboring rays and their first and second transition points. A dashed curve is drawn by connecting the first transition points of all the three rays. Similarly, the dotted curve connects the second transition points of the rays. The distance from the start of a ray to the location of the transition point along the ray is called transition depth. The transition points have variable transition depths in all the rays shown.

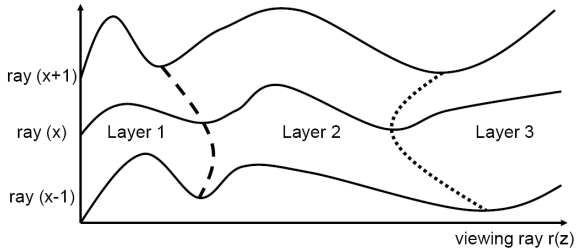


Figure 5: Three rays with position x-1, x and x+1 are shown. A dashed curve is drawn by connecting the first transition point of each of the three rays. It shows an interface between layer 1 and layer 2. A dotted curve shows an interface between layer 2 and layer 3 and is drawn by connecting the second transition points of all the rays.

The part of the dataset that lies between its boundary and the dashed curve in figure 5 is called layer 1. Similarly, data between the dashed curve and the dotted curve is called layer 2 and so forth. We cannot expect constant transition depths across neighboring rays as the main point of feature peeling is to generate images from the individual irregular feature layers. In order to accommodate inaccuracies of our de-noising procedure, we however, want to avoid large changes in the transition depths from one ray to its neighbors. This is achieved by assigning each transition point an importance or a relevance value that is based on its similarity to corresponding transition points in the neighboring rays. This will help to interactively control the level of peeling, i.e. to specify how many layers will be generated.

According to equation (1) we calculate the importance value Ip_{xy} of a transition point by taking the absolute difference of the transition depth $depth_{x,y}$ at the current ray profile x,y and the average of the transition depths in the $3x3$ neighborhood of the ray x,y. Finally we subtract the normalized absolute difference from 1.0. The normalized absolute difference is computed by dividing the absolute difference by $depthmax$. The maximum transition depth $depthmax$ can be set individually for each view-point or it may also be set globally, i.e., equivalent to the diagonal of the bounding box of the dataset.

$$Ip_{xy} = 1.0 - [\left| depth_{x,y} - \sum_{i=-1}^{1}\sum_{j=-1}^{1} \frac{depth_{x+i,y+j}}{9} \right| / depthmax] \quad (1)$$

$$\text{where } 0 \leq depth \leq depthmax$$

The importance value ranges from close to zero to one with zero being the lowest importance and one being the highest importance for a transition point. Transition points with a low importance are potential jumps in a layer and should be discarded. Similarly, the user can also control the level of peeling (i.e., number of layers) by means of a user specified threshold, called *peeling-threshold*. Transition points with importance less than the *peeling-threshold* are ignored by the algorithm.

Listing 1 shows the high level pseudo code for finding transition points using feature peeling. A function, which locates a transition point on a ray for a given layer is presented. We also show the pseudo code for finding a given layer by opacity peeling in Listing 2. Opacity peeling only takes visibility into consideration and is

therefore insensitive to features. It performs front to back compositing until the accumulated opacity is above a certain threshold called *T_high* and the opacity for the current ray position has dropped below a threshold called *T_low*. Feature peeling on the other hand produces layers where each layer corresponds to a feature inside the dataset.

Listing 1: Pseudo code of feature peeling for calculating a transition point on a single ray. A transition point for a given layer is calculated on the basis of user specified thresholds.

```
lowPassFilter() /* filters a location using a user specified filter */
sampVol() /* returns a scalar value from a 3D dataset */
calSl() /* returns slope between the parameters */
calImp() /* returns an importance value for a transition point */

LocateTransitionPoint(layer, rayPos)
  counter = 0 /* counts number of feature layers */
  LocalMinFd = false /* local minimum found or not*/
  nextVal = lowPassFilter(sampVol(rayPos))

  while (NotEndofRay)
    curVal = nextVal
    nextVal = lowPassFilter(sampVol(rayPos + 1))
    slope = calSl(curVal, nextVal)

    if ((slope>0)&&(LocalMinFd == false))
      LocalMinFd = true
      LocalMin = rayPos
    else if ((slope<0)&&(LocalMinFd == true))
      LocalMax = rayPos
      slope = calSl(sampVol(localMin),curVal)
      if (slope>slope_Threshold)
        IPxy = calImp (localMin)
        if (IPxy>peeling_Threshold)
          if (counter == layer)
            return LocalMin
          else
            counter++
      LocalMinFd = false
    rayPos = rayPos++
```

Listing 2: Pseudo code of opacity peeling. Features might be split among opacity layers.

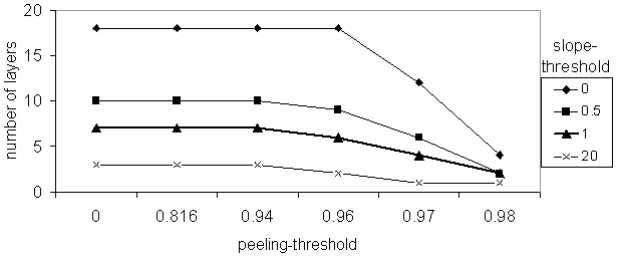
```
FrontToBackComp() /* performs front to back compositing */
T_high /* threshold for the accumulated opacity */
T_low /* threshold for the opacity at current ray position*/


LocateLayerPostion (layer, rayPos)
  accOp = 0 /* accumulated opacity */
  curOp = 0 /* opacity at current ray position */
  counter = 0 /* counts number of opacity layers */
  while (NotEndofRay)
    FrontToBackComp()
    if ((accOp>T_high)&&(curOp<T_low))
      if (counter == layer)
        return accOp
      else
        counter++
        accOp = 0
  rayPos = rayPos++
```

4 IMPLEMENTATION

We implemented feature peeling on an AMD Athlon, 2.1 GHz CPU and an NVidia GeForce 6600 graphics board. Feature peeling is generic with respect to rendering. The separate layers can be rendered for example with DVR, MIP, LMIP. The images that we show in this paper are generated by using DVR. We have used a median low pass filter of width five for de-noising in all of our test cases.

The result images are computed by controlling just two sliders. One slider specifies the *slope-threshold* and the second slider controls the *peeling-threshold*. Both of these thresholds affect the number of layers that are generated by the feature peeling algorithm. Graph 1 shows the number of layers generated from an MRI head dataset by using various combinations of these thresholds. The resolution of the head dataset is 256x256x109.

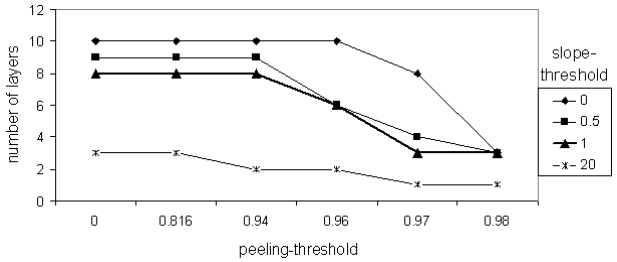
There is no change in the number of layers produced by feature peeling when the *slope-threshold* is zero and the *peeling-threshold* is between 0 and 0.96. There is a sharp decline in the number of layers produced through feature peeling when the *slope-threshold* is set to the lowest value of zero and the *peeling-threshold* is changed from 0.96 to 0.98. On the other hand, there is not a significant change in the number of layers when the *slope-threshold* is set to a high value of 20 and the *peeling-threshold* is changed from 0 to 0.98.



Graph 1: This graph shows the number of layers produced from an MRI head dataset for different combinations of the *peeling-threshold* and the *slope-threshold*. Dataset resolution is 256x256x109.

This shows that by assigning a low value to the *slope-threshold* the number of layers does not vary uniformly with changing the *peeling-threshold*. For a high *slope-threshold* the number of layers that can be produced by manipulating the *peeling-threshold* is limited.

Graph 2 shows the number of layers generated by different combinations of the *slope-threshold* and the *peeling-threshold* for the CT hand dataset. The dataset has a resolution of 244x124x257. The variance in the number of layers with respect to the thresholds is similar as witnessed in graph 1. However, the variance in the number of layers is less than in graph 1 as there are fewer features in the hand dataset.



Graph 2: This graph shows the number of layers produced from the CT hand dataset for different combinations of the *peeling-threshold* and the *slope-threshold*. Dataset resolution is 244x124x257.

The *slope-threshold* can also affect how features show up in an image. The user can interactively change the value of the *slope-threshold* for each layer according to the requirements. Figure 6(a) shows the first layer of the carp dataset. Figure 6(b) and (c) both show the second layer of the carp with different *slope-thresholds*. In 6(b) we observe that noise and parts of the spinal cord are partially occluding the swim bladder of the carp. An increase in the *slope-threshold* can fade away the spinal cord and noise and display a clear view of the swim bladder as in 6(c). The swim bladder is visually highlighted by a rectangle in 6(c).

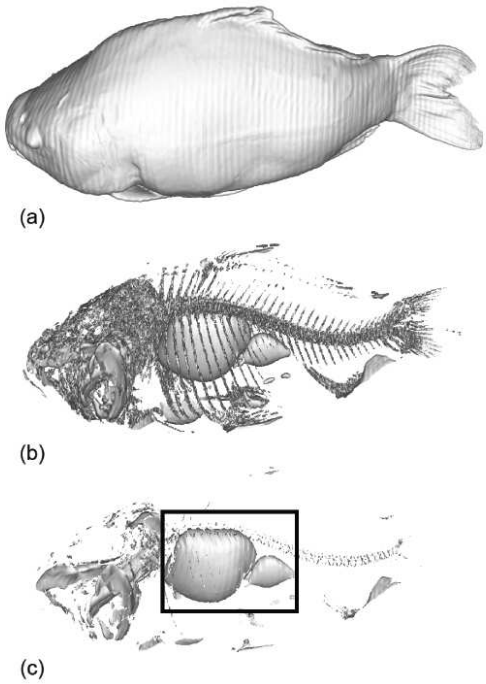


Figure 6: The first layer of the carp is shown in (a). (b) and (c) display the second layer of the carp with different values for the *slope-threshold*. The *slope-threshold* is 1.3 and 7.0 for (b) and (c) respectively.

We measure the performance of our system using an MRI head dataset. We also compare our results with an implementation of opacity peeling. Table 1 shows the rendering speed of feature peeling and opacity peeling for four different layers of the head dataset. Opacity peeling produces layers through Direct Volume Rendering. In order to extract a certain layer, all the layers on top of that layer have to be rendered as well. Feature peeling defines layers based on the ray profile analysis and requires less computation to identify a feature layer. The *slope-threshold* was set to 1.0 (45 degrees) and the *peeling-threshold* was set to 0.965. The image resolution is 512x512.

The performance decreases with an increasing number of layers. The frame rate is dependent on the depth of the transition points and also on the number of local minima being skipped. Therefore, the rendering time must increase to visualize layers of larger depth.

5 RESULTS

We show results by using two MRI datasets (head angiography and head) and two CT datasets (present and hand). We also show the variance in the transition depth across the layers produced by feature peeling and compare the results with opacity peeling.

Table 1: This table shows the performance of feature peeling. Column 2 shows the rendering speed in frames per second if only the *slope-threshold* is used to decide if a local minimum will be considered a transition point. Column 3 shows the rendering speed for different layers when both thresholds are calculated. The last column shows the performance of opacity peeling.

	Slope-threshold only	Slope-threshold and peeling-threshold	Opacity peeling
Layer1	8.2fps	7.8fps	4.3fps
Layer2	6.1fps	5.9fps	3.6fps
Layer3	5.6fps	5.2fps	2.3fps
Layer4	5.2fps	4.8fps	2.0fps

Figure 7 shows an MRI angiography dataset divided into two layers. Figure 7(a) shows the Direct Volume Rendering of the dataset without feature peeling. A lot of high frequency noise is present and thus the veins are not clearly visible. Figure 7(b) shows the first layer of the MRI dataset, generated through feature peeling. All the high frequency noise is filtered out and we have a clear view to the veins. However, two veins of approximately the same shape are present in the area marked with an oval. The vein that is covering the other vein is removed in figure 7(c), to reveal the hidden vein by showing the next layer. Figure 7(d) again shows the first layer rendered through feature peeling after slightly rotating the dataset. Both the veins can now be seen to have almost the same shape.

Figure 8 shows six layers of the present dataset: the complete present as it looks from the outside 8(a), the inner box 8(b), the snow globle 8(c), the St. Stephan's Cathedral 8(d), the platform and the hidden towers of the St. Stephan's Cathedral 8(e) and the mouse 8(f) are all distinctively visible in seperate layers.

In figures 9(a) to 9(c) and 9(e), we show four layers of an MRI head dataset. The first layer in figure 9(a) shows the outer layer, the second layer in figure 9(b) shows the brain surface, the third layer in figure 9(c) uncovers the eyeball and reveals parts of the ventricles and the fourth layer in figure 9(e) shows the corpus callosum, inner parts of the ventricles and the right eye. Figure 9(d) and 9(f) show the third and fourth layers of the same dataset produced using opacity peeling. Ventricles and corpus callosum are for example less recognizable in figure 9(d) and 9(f) compared to figure 9(e).

Figures 10(a) and 10(b) show the second and the third layer of a hand dataset produced by opacity peeling. Figure 10(a) shows bones and some parts of the veins. The third layer in figure 10(b) skips large parts of the veins, making it difficult to visualize them. Figure 10(c) shows zoom-in of the region where veins have been skipped.

Figure 10(d) and 10(e) show the second and the third layer of the hand dataset generated using feature peeling. The second layer in figure 10(d) peels off the upper bone and shows some veins and the lower part of the bone. The third layer in figure 10(e) removes bone to show occluded veins and arteries. Figure 10(f) is a zoom-in of the third layer.

Graph 3(a), 3(b) and 3(c) display the standard deviations for the second, third and fourth layer of the head dataset using both feature peeling and opacity peeling. Feature peeling consistently produces lower variance on the layer boundaries as compared to opacity peeling. Opacity peeling is concerned with visibility irrespective of feature boundaries, while feature peeling separates the volume data along smooth feature interfaces. We have used a *slope-threshold* of value 1.0 to generate these graphs.

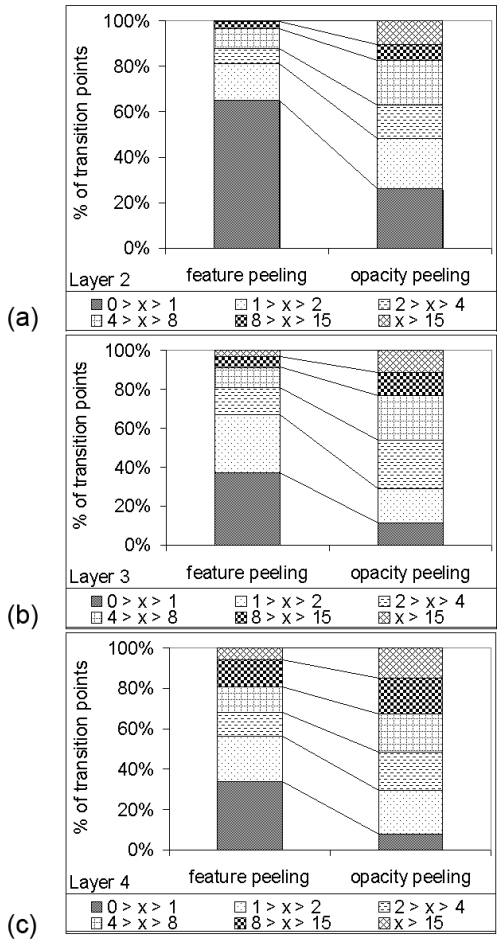
6 Conclusion and Future Work

This paper introduces feature peeling, a browsing of volumetric data in feature layers for a selected view-point. Feature peeling successfully works for medical datasets. It has shown promising results for MRI datasets, which are hard to visualize using traditional 3D visualization techniques.

While feature peeling requires the specification of two thresholds (a *slope-threshold* as well as a *peeling-threshold*), we believe that these thresholds can remain constant over a large amount of patient studies. However, this needs further investigation.

Further, a more thorough test needs to be done on the influence of the de-noising method. Currently we are simply using a low pass median filter. It is possible that a bi-lateral filtering in combination with Gaussian smoothing or similar approaches might improve the coherency of transition points.

It will be interesting to investigate if we can use feature peeling to dynamically select view positions. We would like to detect the regions inside the dataset where the data field is changing most rapidly. The algorithm can then calculate an optimal viewing position for these regions and perform feature peeling. This could provide a separate view for almost every feature of the dataset.



Graph 3: The variance in the transition depth of a layer for feature peeling as well as for opacity peeling is shown for three layers. (a) shows variation of depth in the second layer of the MRI head dataset. (b) and (c) show results for the third and the fourth layers respectively. The legend includes ranges of standard deviations into which the transition points were categorized.

7 Acknowledgments

The carp, head angiography, head and hand datasets are courtesy of Michael Scheuring from the University of Erlangen, Özlem Gürvit from the Institute for Neuroradiology, Siemens Medical Systems and Tiani Medgraph respectively. The present dataset was scanned with an industrial CT by Christoph Heinzl, Wels College of Engineering, Austria. This project has been funded in part with grant from Higher Education Commission of Pakistan.

References

[1] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. The contour spectrum. In *IEEE Visualization '97*, pages 167–175, 1997.

[2] Steven Bergner, Torsten Möller, Mark S. Drew, and Graham D. Finlayson. Interactive spectral volume rendering. In *IEEE Visualization '02*, pages 101–108, 2002.

[3] Duffy Brian, Denby Brian, and Hamish Carr. On histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2007)*, 12(5):1259–1266, September-October 2006.

[4] Silvia Castro, Andreas König, Helwig Löffelmann, and Meister Eduard Gröller. Transfer function specification for the visualization of medical data. Technical Report, Vienna University of Technology, March 1998.

[5] Klaus Engel, Markus Hadwiger, Joe M. Kniss, Christof Rezk-Salama, and Daniel Weiskopf. *Real-time Volume Graphics*. A. K. Peters, 2006.

[6] Jinzhu Gao and Han-Wei Shen. Hardware-assisted view-dependent isosurface extraction using spherical partition. In *Joint Eurographics-IEEE TCVG Symposium on Visualization '03*, pages 267–276, 2003.

[7] Karl Heinz Höhne, Michael Bomans, Andreas Pommert, Martin Riemer, Carsten Schiers, Ulf Tiede, and Gunnar Wiebecke. 3D visualization of tomographic volume data using the generalized voxel model. *The Visual Computer*, 6:28–36, 1990.

[8] Gordon Kindlmann and James W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Symposium on Volume Visualization '98*, pages 79–86, 1998.

[9] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *IEEE Visualization '01*, pages 255–262, 2001.

[10] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, 2002.

[11] GA. Laub and WA. Kaiser. MR angiography with gradient motion refocusing. *Journal of Computer Assisted Tomography*, 12(3):377–382, 1988.

[12] Tom Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12:233–250, 1993.

[13] S. Naple, M.P. Marks, R.D. Rubin, R.B. Jeffrey, M.D Dake, D.R. Enzmann, and McDonnell. CT angiography using spiral CT and maximum intensity projections. In *Radiology '92*, pages 607–610, 1992.

[14] Vladimir Pekar, Rafael Wiemker, and Daniel Hempel. Fast detection of meaningful isosurfaces for volume data visualization. In *IEEE Visualization '01*, pages 223–230, 2001.

[15] Simeon Potts and Torsten Möller. Transfer function on a logarithmic scale for volume rendering. In *Graphics Interface '04*, pages 57–63, 2004.

[16] Christof Rezk-Salama and Andreas Kolb. Opacity peeling for direct volume rendering. *Computer Graphics Forum*, 25:596–606, 2006.

[17] S. Rossnick, D. Kennedy, G. Laub, G. Braeckle, R. Bachus, D. Kennedy, A. Nelson, S. Dzik, and P Starewicz. Three dimensional display of blood vessels in MRI. In *IEEE Computers in Cardiology '86*, pages 183–196, 1986.

[18] Yoshinobu Sato, Nobuyuki Shiraga, Shin Nakajima, Shinichi Tamura, and Ron Kikinis. Local maximum intensity projection (LMIP): A new rendering method for vascular visualization. *Journal of Computer Assisted Tomography*, 22(6):912–917, 1998.

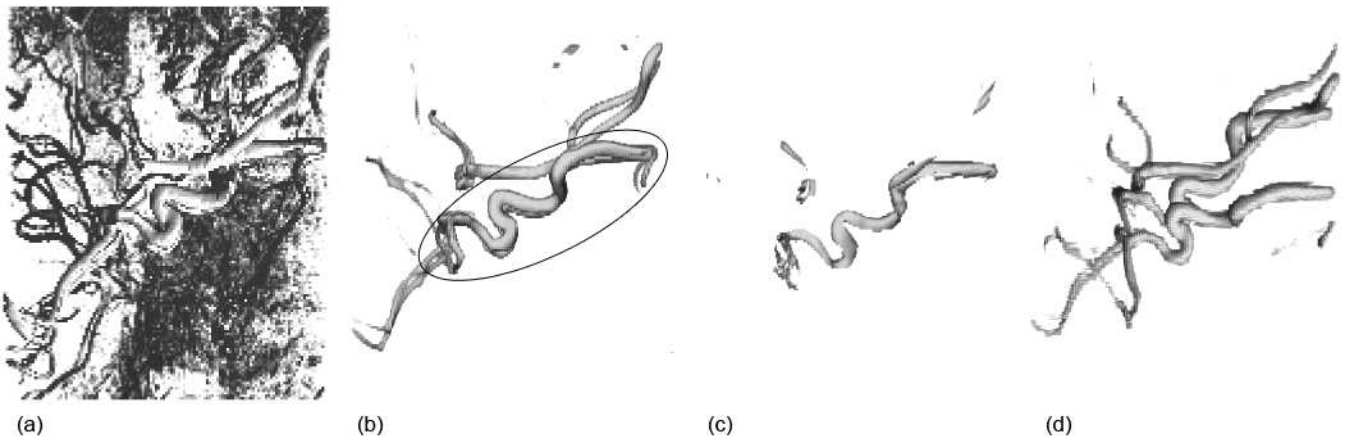


Figure 7: Direct Volume Rendering of an MRI dataset (head angiography) is shown in (a). The first layer of the dataset is given in (b). Two veins with approximately the same shape exist in the region marked with an oval. The vein that occludes the other one is peeled away in layer 2 (c) to show the hidden vein. Both veins are visible in (d), which is the first layer of the same dataset rendered through feature peeling after slightly rotating the dataset. The *slope-threshold* is set to 3.0 and the *peeling-threshold* is 0.4.

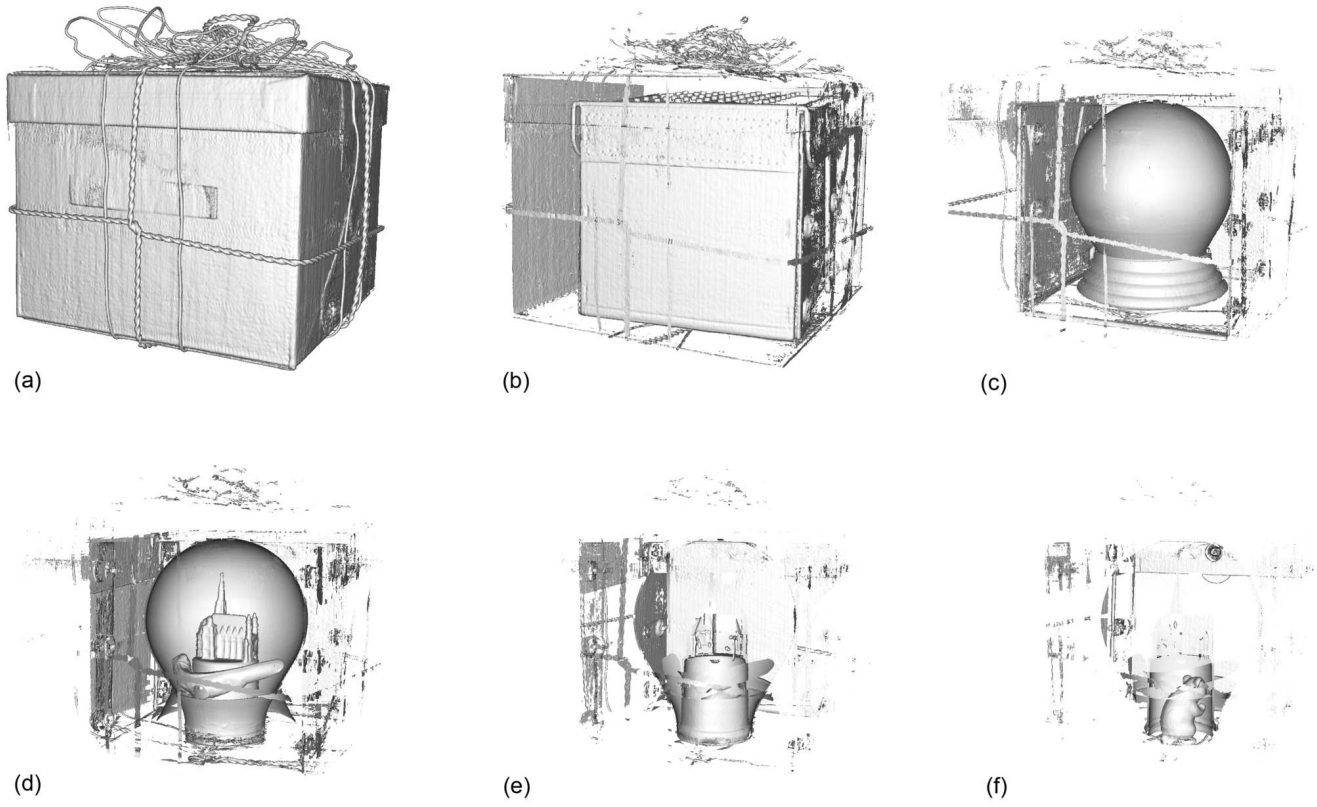


Figure 8: From (a) to (f), six layers of the present dataset are shown in order. Present is a CT dataset with a resolution of 492x492x442. The *slope-threshold* is 2.0 and the *peeling-threshold* is 0.9.

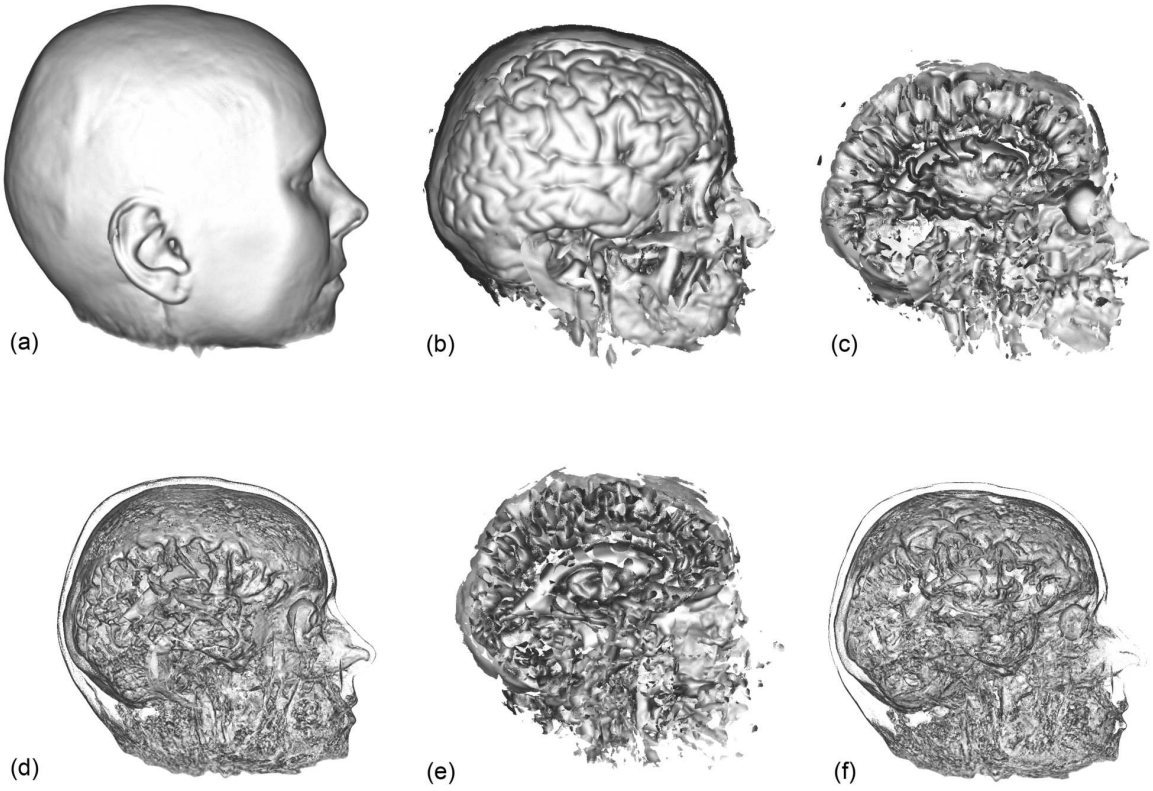


Figure 9: (a), (b), (c) and (e) show the first, the second, the third and the fourth layer generated using feature peeling. The third and the fourth layer obtained using opacity peeling are shown in (d) and (f). (e) shows ventricles and the right eye as well as a clearly distinguishable corpus callosum. These features are not clearly visible neither in (d) nor in (f). The *Slope-threshold* is 1.0 and the *peeling-threshold* is 0.965.

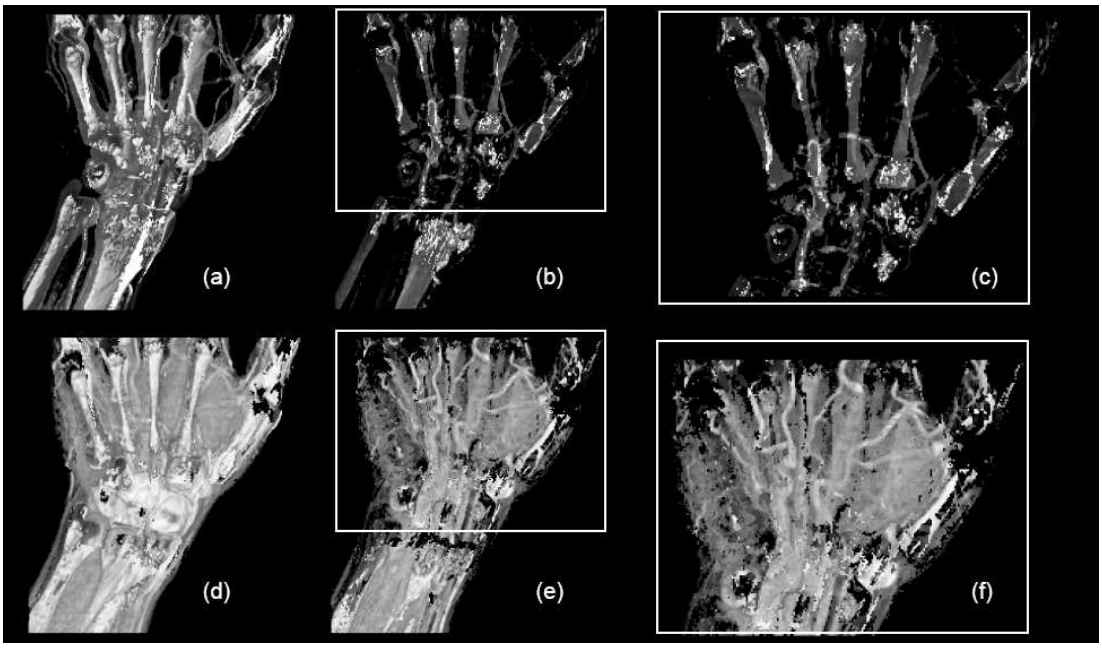


Figure 10: (a) and (b) show the second and the third layer of the hand dataset rendered using opacity peeling. (d) and (e) show the second and the third layer obtained using feature peeling. The veins in (e) are better visible through feature peeling. (c) and (f) show zoom-ins for (b) and (e) respectively. The first layer of the hand dataset is not shown as it is not relevant here.The *Slope-threshold* is 1.0 and the *peeling-threshold* is 0.97.