TECHNICAL REPORT

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

TR-186-2-07-03
February 2007

Keywords: feature peeling, volume visualization, transfer function, ray analysis

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 profile along the line of sight during rendering and cuts it into layers, according to the peaks and valleys found, which we call transition points. The sensitivity of these transition points is calibrated via two thresholds. The slope threshold influences 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 opacity peeling approach [14], we demonstrate that we can reveal detectable features even in the third and forth layers for both, CT and MRI datasets.

CR Categories: 1.3.7 [Computer Graphics]: Three dimensional graphics and realism—Life Cycle

Keywords: feature peeling, volume visualization, transfer function, 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 maligned 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 is then reused for several patients, this is not always possible.

Especially, Magnetic Resonance Imaging (MRI) datasets are more difficult then and different from Computed Tomography (CT) datasets. Hounsfield numbers give a good indication of the tissue type in CT, independent of the patient. In contrast, the variance of tissue response between different patients in MRI dataset 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 very 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 still uses 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. The algorithm allows the user to browse through the layered features of the dataset for each particular view point (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 has been inspired by the work of Rezk-Salama and Kolb [14] and we detail our 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 to the 1D transfer function. From simple histograms [3] to the contour spectrum [1] to Laplacian-weighted histograms [12] have been suggested. Potts and Möller [13] investigate the usage of a logarithmic scale that eases transfer function specification.

Multi-dimensional transfer functions [6] have been introduced which assign optical properties based on data values but also first and second derivative information thereof. Kindlmann and Durkin [5] atomize 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. [7] 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 [8]. A detailed description of volume graphics techniques and there applicability to medical visualization is given by Hadwiger et al. [4].

While all of these techniques require user intervention, one can argue, that it would be preferable to cut this step from the visualization process in order to provide quick insight into the dataset. This was the basic idea of the opacity peeling approach of Rezk-Salama and Kolb [14]. It allows a layered browsing of the dataset instead. The layers are defined through accumulated opacity and basically all information in the dataset is visible. As the layers are based purely 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 a careful analysis of ray profiles and split the ray not along opacity thresholds, but rather at possible feature transition regions.

3 Feature Peeling

Along a ray in a dataset, we find valleys, peaks and homogeneous regions. Areas of interest are, by default, the regions where the data field is changing. If there are detectable transitions from one tissue to another, then these transitions will be present in the areas where the data field is changing. 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 performs front to back or back to front compositing on 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 [10].

Maximum Intensity Projection (MIP) only displays the highest peak in a ray profile [9] [11] [15]. Similarly, Local Maximum Intensity Projection (LMIP) [16] searches for a peak above a specified threshold along the ray. The first peak with a value above the threshold is displayed and the process is terminated.

Consider figure 1, which shows a ray profile with three features. 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.

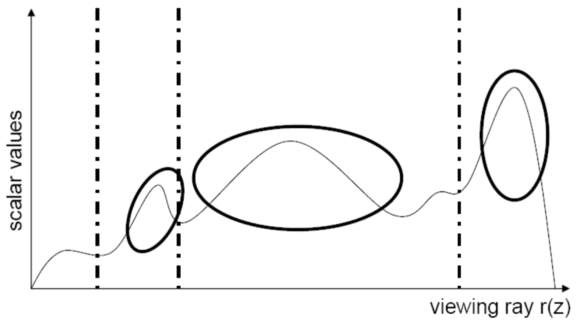


Figure 1: A ray profile showing three features as prominent density peaks. Features are marked with ovals and vertical lines show the transition points. Transition points split a ray profile into different layers.

We present a high level explanation of feature peeling in figure 2 by a 2D illustration. We use three concentric circular layers in figure 2(a) as a dataset. A ray is also shown, which passes through the centre of the dataset. The ray profile and the transition points corresponding to the ray in figure 2(a) are shown in figure 2(b). On the right we show the first three resultant layers from the current view point. The example in figure 2 establishes the importance of finding transition points that are representative of the features in a dataset.

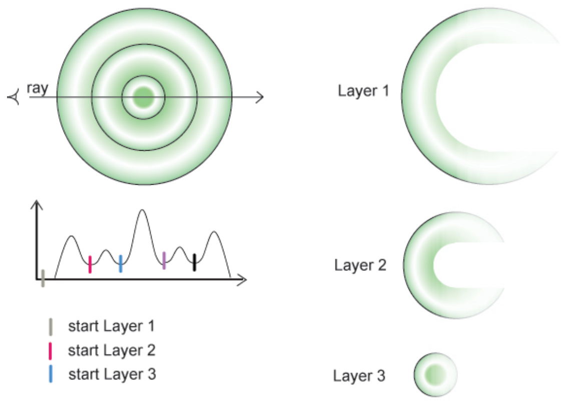


Figure 2: Three concentric layers of a dataset are shown in (a). A ray is depicted in the dataset and the corresponding ray profile is drawn in (b). Small marks on the ray profile are shown in (b) which depict the transition points. On the right we show the first three layers that will be generated from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

3.1 Locating Transition Points

The search for the transition points can be based on the first and second derivatives of the ray profile. While the algorithm proceeds along the ray, it monitors the first and the 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.

These transition points can vary based on the number of features along a ray and the amount of noise. Especially in the case of MRI datasets, there can be a large number of transition points present because of high frequency noise. We therefore need to remove high frequency noise by using a low pass filter. This will smooth the profile, removing all the transition points that exist because of noise and enhancing 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*.

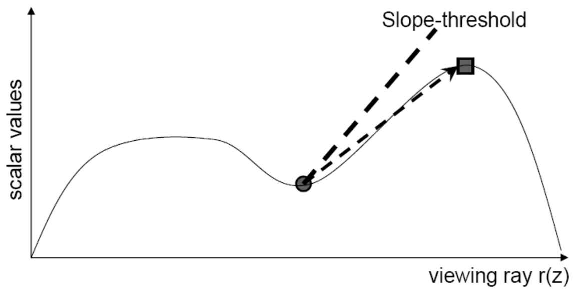


Figure 3: The grey circle shows the local minimum and the grey square shows the first local maximum after this local minimum. The thick dashed line depicts the *slope-threshold*. The slope between the local minimum and the local maximum is shown by a dashed arrow.

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 local minima that are the result of low frequency noise and which are discarded. Figure 4(c) shows a slice from the MRI dataset with a red line showing the path of 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).

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 point of all the three rays. Similarly, the dotted curve connects the second transition point 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 depth in all the rays shown.

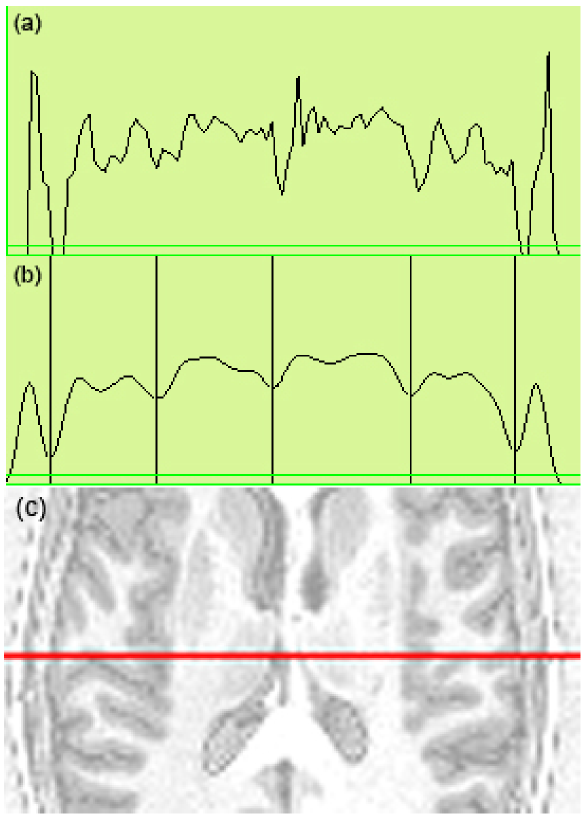


Figure 4: Original ray profile of an MRI dataset in (a) shows a lot of high frequency noise. In (b) the very profile of (a) is filtered with a median low pass filter. Vertical lines depict the transition points and arrows indicate the local minima skipped based on the *slope-threshold*. (c) shows a slice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

That 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 do, however, want to avoid large changes in the transition depths from one ray to its neighbor. 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 us to interactively control the level of peeling, i.e. to specify how many layers will be generated.

We show the calculation of the importance value in two steps. First we calculate a raw importance value, which

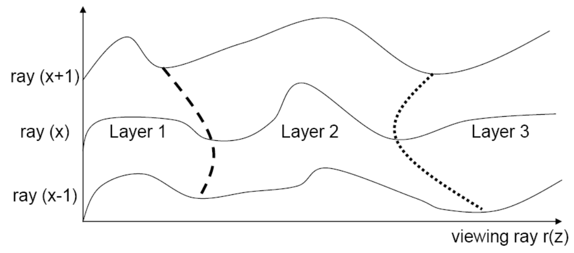


Figure 5: Three rays with position x-1, x and x+1 are shown. A dashed curve is drawn by connecting the first transition point of each of the three rays. It shows an interface between layer 1 and layer 2. A dotted curve shows an interface between layer 2 and layer 3 and is drawn by connecting the second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

is an intermediate value. Then we use this raw importance value to calculate the importance value that always lies in the range from zero to one. The raw importance value RIp_{xy} of a transition point is calculated as 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, where transition depth lies between *zero* and the maximum length of a ray *depthmax*:

$$RIp_{xy} = \left| depth_{x,y} - \sum_{i=-1}^{1} \sum_{j=-1}^{1} \frac{depth_{x+i,y+j}}{9} \right|$$

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

The importance value Ip_{xy} of a transition point is then calculated by subtracting the normalized raw importance value RIp_{xy} from 1.0:

$$Ip_{xy} = 1.0 - [RIp_{xy}/depthmax]$$

The importance value ranges from zero to one with zero as the lowest importance and one as the highest importance for a transition point. The value from the above calculation encodes the importance of 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, which we call *peeling-threshold*. Transition points with importance less then 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 for a given layer is presented. We also show the algorithm for producing layers by opacity peeling in Listing 2. Opacity peeling only takes visibility into consideration and is therefore insensitive to features. Feature peeling on the other hand produces layers where each layer corresponds to a feature inside the dataset.

Listing 1: High level pseudo code for calculating transition points on a single ray is given. It locates a transition point for a given layer on the basis of user specified thresholds.

lowFil() ⇒ filters a location using a user specified low pass filter
sampV() ⇒ returns a scalar value from a 3D volume
calSl() ⇒ returns slope between the parameters
calImp() ⇒ returns an importance value for a transition point

```
LocateTransitionPoint(layer, rayPos)
  counter = 0
  state = 0
  nextVal = lowFil(sampV(rayPos))

  while (NotEndofRay)
    currentVal = nextVal
    nextVal = lowFil(sampV(rayPos + 1))
    sl = calSl(currentVal, nextVal)

    if ((sl>0)&&(state == 0))
      state = 1
      LocalMin = rayPos
    else if ((sl<0)&&(state == 1))
      LocalMax = rayPos
      sl = calSl(sampV(localMin),currentVal)
      if (sl>slope_Threshold)
        IPxy = calImp (localMin)
        if (IPxy>peeling_Threshold)
          if (counter == layer)
            return LocalMin
          else
            counter++
      state = begin
    rayPos = rayPos++
```

Listing 2: The algorithm for generating layers through opacity peeling is shown. Opacity peeling does not have a mechanism to position the layers so that features are not split between layers.

FrontToBackCompositing() ⇒ performs front to back compositing
accOp ⇒ Accumulated Opacity
curOp ⇒ Opacity at current ray position
highTh ⇒ Higher Threshold
lowTh ⇒ Lower Threshold

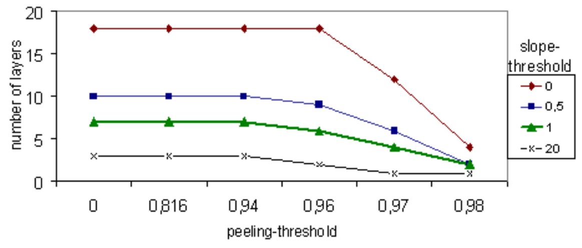
```
LocateLayerPostion (layer, rayPos)
  accOp = 0
  curOp = 0
  counter = 0
  while ((NotEndofRay)
    FrontToBackCompositing()
    if ((accOp>highTh)&&(curOp<lowTh))
      if (counter == layer)
        return accOp
      else
        counter++
        accOp = 0
  rayPos = rayPos++
```

4 Implementation

We implement our system on an AMD Turion, 2.0 GHz CPU and an NVidia GeForce Go7300 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 Direct Volume Rendering. We have used a median low pass filter 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 will be generated as a result of the feature peeling algorithm. Graph 1 shows the number of layers generated from the MRI head dataset by using various combinations of these thresholds. The resolution of the head dataset is 256x256x109.

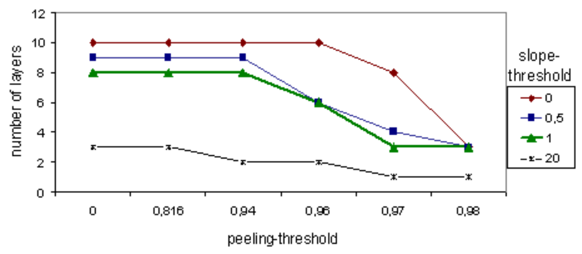
There is no change in the number of layers produced by feature peeling when the *slope-threshold* is zero and the *peeling-threshold* is between 0 and 0.96. There is a sharp decline in the number of layers produced through feature peeling when the *slope-threshold* is set to the lowest value of zero and the *peeling-threshold* is changed from 0.96 to 0.98. On the other hand, there is not significant change in the number of layers produced 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 will 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. Hand 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 then in graph 1 as there are fewer features in the hand dataset.



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

We measure the performance of our system using an MRI head dataset. We also compare our results with an implementation of opacity peeling. Our image resolution is 512x512. Table 1 shows the rendering speed of feature peeling and opacity peeling for four different layers of the head dataset. The *slope-threshold* was set to 1.0 (45 degrees) and the *peeling-threshold* was set at 0.965. This yields four layers.

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

Table 1: This table shows the performance of our algorithm. Column 2 shows the rendering speed in frames per second when 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	7.7fps	7.8fps
Layer2	6.1fps	5.2fps	5.0fps
Layer3	5.6fps	3.8fps	2.0fps
Layer4	5.2fps	3.3fps	1.8fps

5 Results

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

Figure 6 shows an MRI angiography dataset divided into two layers. Figure 6(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 6(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 another vein is removed in figure 6(c), to reveal the hidden vein by showing the next layer. Figure 6(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.

In figures 7(a) to 7(c) and 7(e), we show four layers of an MRI head dataset. The first layer in figure 7(a) shows the outer layer, the second layer in figure 7(b) shows the brain surface, the third layer in figure 7(c) uncovers the eyeball and reveals part of the ventricles and the forth layer in figure 7(e) shows the corpus callosum, inner part of the ventricles and the right eye. Figure 7(d) and 7(f) show the third and forth layers of the same dataset produced using opacity peeling. Ventricles and corpus callosum are for example less recognizable in figure 7(d) and 7(f) compared to figure 7(e).

Figures 8(a) and 8(b) show the second and the third layer of a hand dataset produced by opacity peeling. Figure 8(a) shows bones and some parts of the veins. The third layer in figure 8(b) skips large parts of the veins, making it difficult to visualize them. Figure 8(c) shows zoom-in of the region where veins have been skipped.

Figure 8(d) and 8(e) show the second and the third layer of the hand dataset generated using feature peeling. The second layer in figure 8(d) peels off the upper bone and shows some veins and the lower part of the bone. The third layer in figure 8(e) removes bone to show occluded veins and arteries. Figure 8(f) is a zoom-in of the third layer.

Graph 3(a), 3(b) and 3(c) display the standard deviations over the second, third and forth layers 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 bilateral 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 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.

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 Moeller, 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*, 12:1259–1266, 2006.

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

[5] 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.

[6] 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.

[7] 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.

[8] Andreas Koenig, Helwig Loeffelmann, and Meister Eduard Groeller. Transfer function specification for the visualization of medical data. Technical Report, Vienna University of Technology, March 1998.

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

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

[11] 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.

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

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

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

[15] 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.

[16] 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.

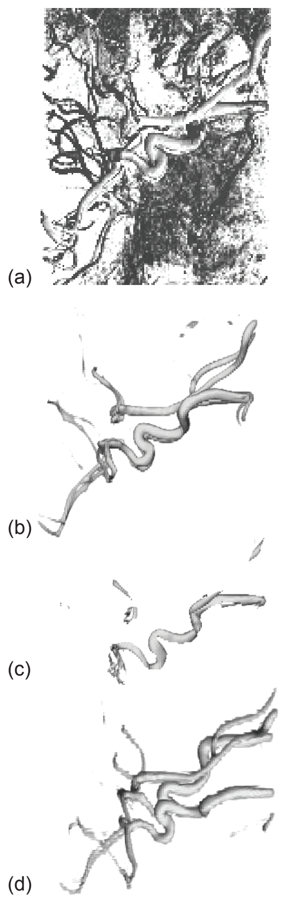
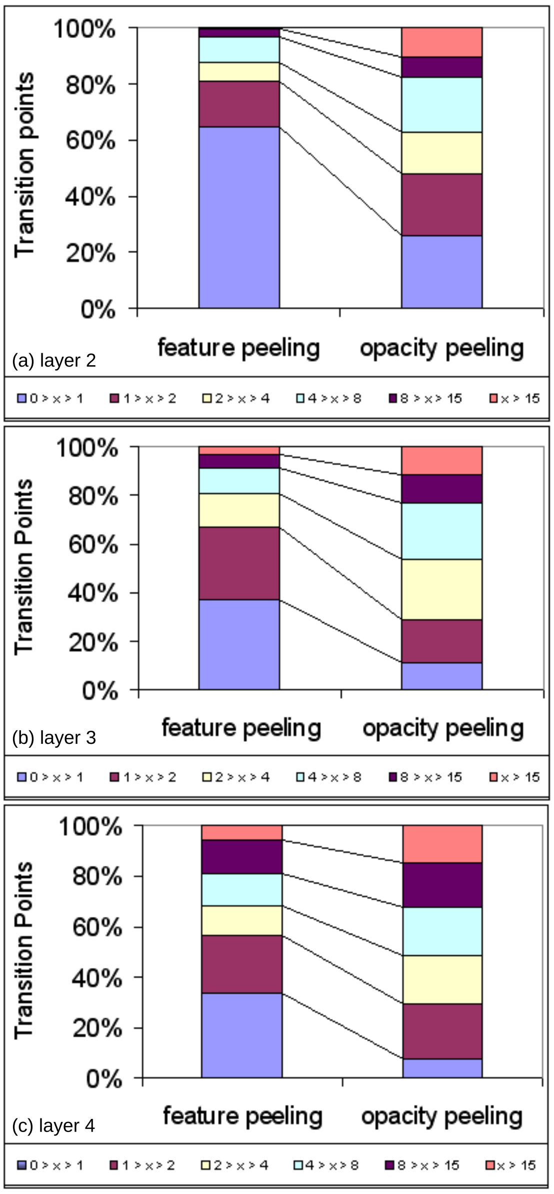


Figure 6: Direct Volume Rendering of an MRI dataset is shown in (a). The first layer of the dataset is given in (b). Two veins with approximately the same shape exist in the region marked with an oval. The vein that occludes the other one is peeled away in layer 2 (c) to show the hidden vein. Both the veins are visible in (d), which is the first layer of the same dataset rendered through feature peeling after slightly rotating 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 second and the third layers respectively. Legend includes ranges of standard deviations in which the transition points were categorized.

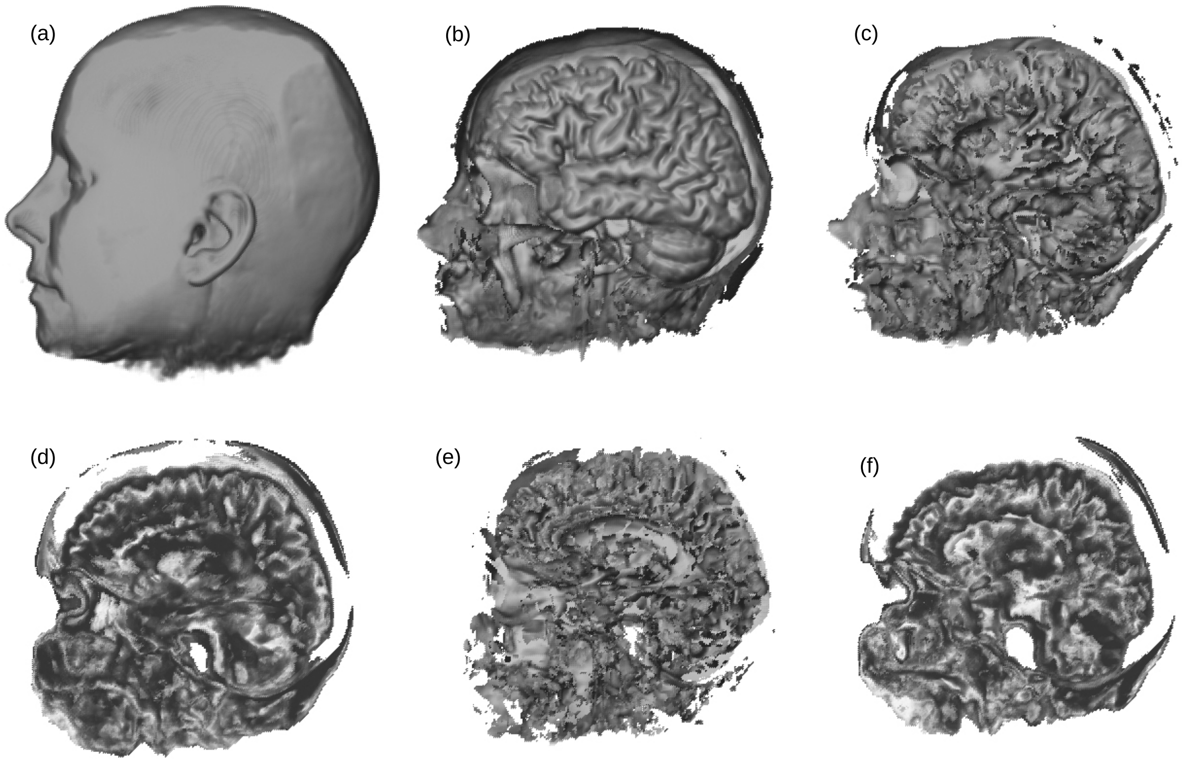


Figure 7: (a), (b), (c) and (e) show the first, the second, the third and the forth layer generated using feature peeling. The third and the forth layer obtained using opacity peeling are shown in (d) and (f). (e) shows ventricles and the right eye as well as a clearly distinguishable corpus callosum. These features are not clearly visible neither in (d) nor in (f). *Slope-threshold* is 1.0 and *peeling-threshold* is 0.965.

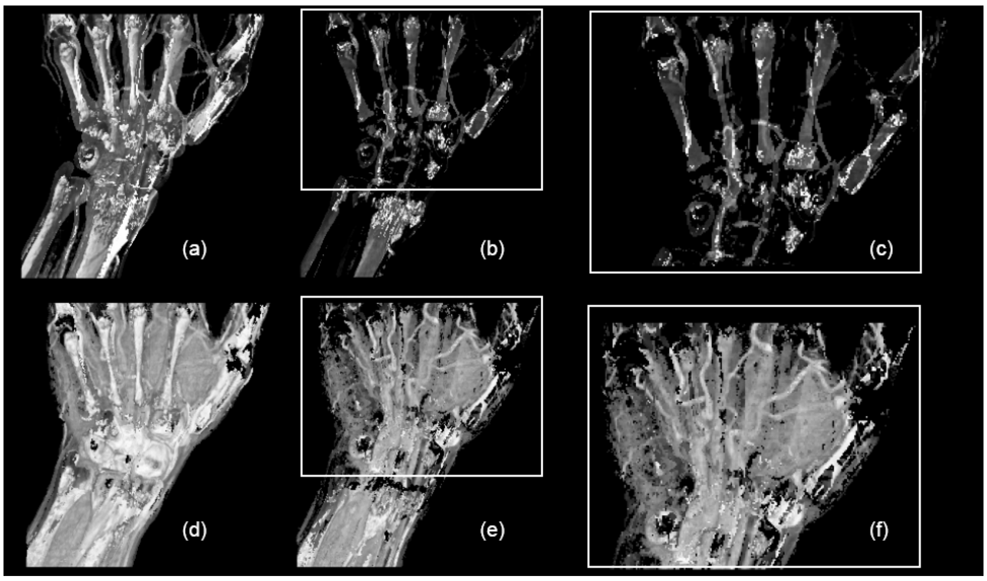


Figure 8: (a) and (b) show the second and the third layer of the hand dataset rendered using opacity peeling. (d) and (e) show the second and the third layer obtained using feature peeling. The veins in (e) are better visible through feature peeling. (c) and (f) show zoom-ins for (b) and (e) respectively. First layer of the hand dataset is not shown as it is not relevant here. *Slope-threshold* is 1.0 and the *peeling-threshold* is 0.97.