Diffusion-Based Applications for Interactive Medical Image Segmentation

Laura Fritz[*]

VRVis Research Center
Vienna, Austria

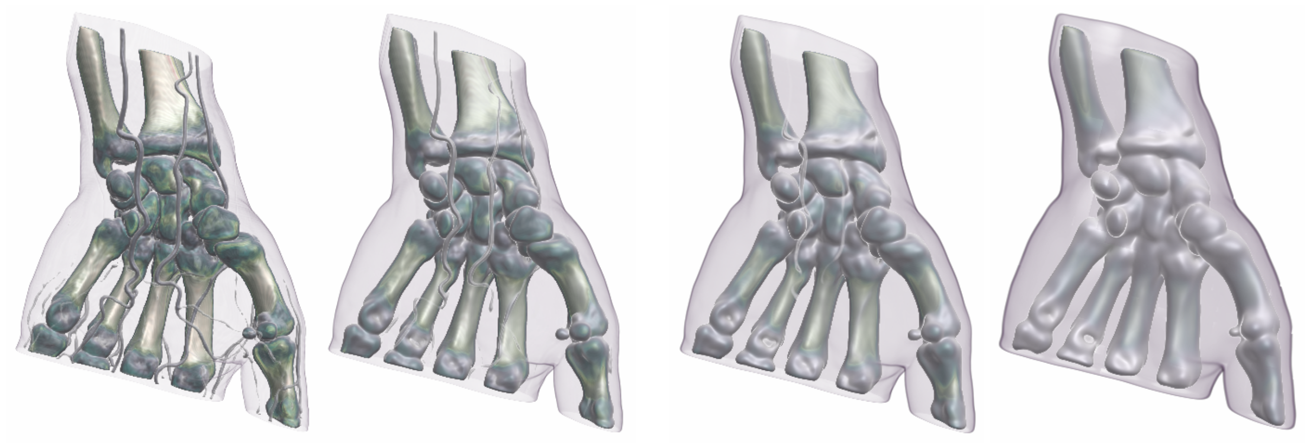


Figure 1: Nonlinear isotropic diffusion after 4, 10 and 25 iterations compared with the original data volume (left).

Abstract

Volume segmentation is an important part of medical diagnostics, and fast solutions which nevertheless can handle large datasets are required. This paper presents a diffusion-based method that can be used for volume smoothing and segmentation purposes, and has been implemented on the GPU in order to achieve interactive performance. We describe the nonlinear isotropic and anisotropic diffusion methods, which – in combination with seeded region growing – are also used for segmentation purposes. The procedures are embedded in a high-quality hardware based volume renderer and provide interactive data handling and parameter editing.

Keywords: volume smoothing, segmentation, GPU, isotropic diffusion, anisotropic diffusion, region growing

1 Introduction

Medical image segmentation i.e., the identification of individual objects in images and volumetric datasets, is an important topic in clinical diagnostics. Medical imaging allows scientists and surgeons to glean important information by peering noninvasively into the human body and separating out anatomical structures (e.g. tumors). There exists a huge amount of solutions, most semiautomatic, requiring highly trained staff and parameter settings. Therefore it is very important to achieve interactivity while executing a lot of computations on large datasets. The main topic of this work is diffusion based image smoothing and segmentation, implemented on the GPU (graphics processing unit), to ensure the highest possible interactivity for working with large datasets. We use nonlinear isotropic and anisotropic diffusion models in combination with different nonlinear diffusivities, to obtain good results for different datasets. In this approach, the power of graphics hardware is used to obtain interactivity during the smoothing and segmentation process. The GPU is, nowadays, a fully programmable parallel processor and therefore well suited for a high throughput of data that is essential in medical volume handling. This encourages interactivity because the computation of the volume preprocessing (e.g. image smoothing) as well as the segmentation and rendering of the volume are executed in parallel to observe the progress of the algorithm. In our approach, image smoothing is mainly used for preprocessing of the datasets before the segmentation is executed. In the next section, we will first introduce the basic idea of diffusion, especially the nonlinear isotropic and anisotropic algorithms are explained. We also show the discretization of these algorithms used in our implementation. In Section 3, we show how to use our approach for medical image segmentation. Especially the processing steps that are essential to obtain good segmentation results are spec-

[*]laura@vrvis.at

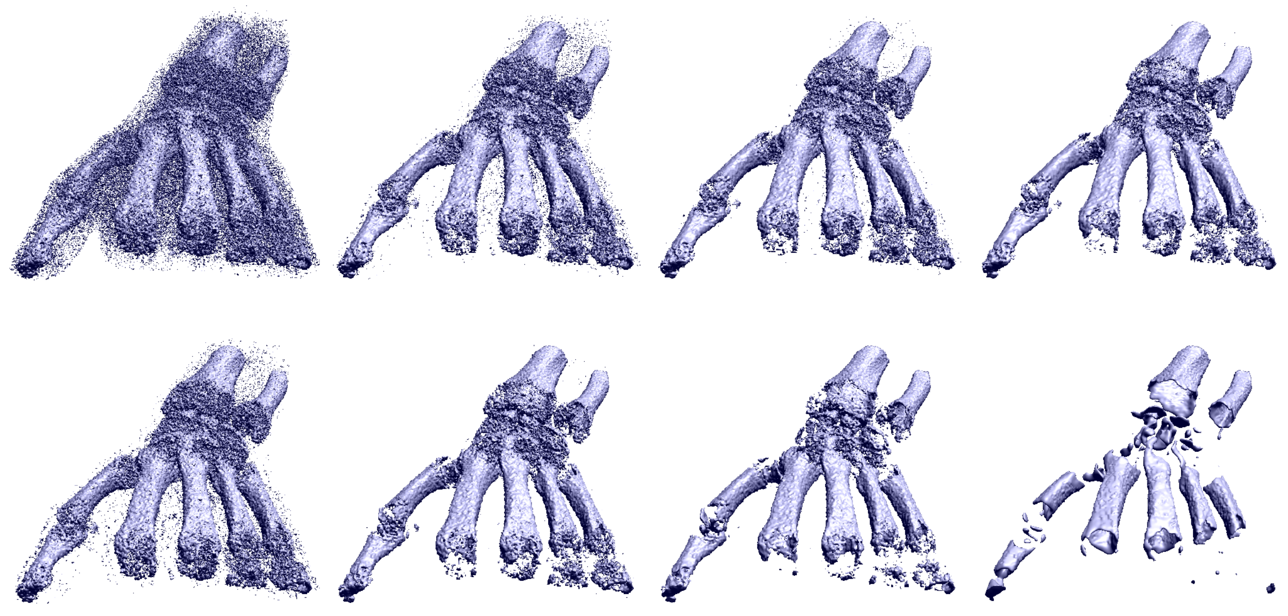


Figure 2: Uniform white noise is applied to the original dataset (first image in the top row) and isosurface rendered with GPU raycasting. The smoothing is shown with anisotropic diffusion after 8 (about 4.08 sec), 14 (about 7.14 sec) and 24 (about 12.24 sec) iterations (top row) and with isotropic diffusion after 2 (about 0.116 sec), 5 (about 0.29 sec), 7 (about 0.406 sec) and 15 (about 0.87 sec) iterations (bottom row). It is obvious that the anisotropic diffusion preserves the structure while removing the noise, whereas the isotropic diffusion destroys the structure using less iterations.

ified. Section 4 explains the implementation details of our hardware-based algorithm where also an overview of the shading language Cg is given. In Section 5, we demonstrate our approach considering an example, where we illustrate a stepwise segmentation of a head dataset. Section 6 shows some results we have achieved.

Related Work
The image smoothing and segmentation techniques used in this paper, are based on nonlinear diffusion algorithms described in [12] and [13]. There exists various applications which uses modifications of one of the first nonlinear diffusion formulations from [5] like [1, 2]. Because the GPU is optimized for a high throughput of data, it is well suited for implementations on medical image processing as shown e.g. in [6, 7, 8]. The segmentation method using diffusion-based seeded region growing, used in this work, was first introduced by [10]. The segmentation tool for the isotropic and anisotropic nonlinear diffusion is integrated in a high-quality, hardware based volume renderer [4] by Markus Hadwiger.

2 Diffusion Background

In our work, diffusion is used both for image smoothing, as shown in Figure 2 and segmentation purposes, also described in Section 3.

Continuous Diffusion Equation

A common intuitive notion of diffusion is a physical process that equilibrates concentration differences without creating or destroying mass as described in [13]. A composition of *Fick's law* and the *continuity equation* results in the mathematical formulation of the *diffusion equation*:

$$\boxed{\partial u_t = \mathbf{div}(D \cdot \nabla u).} \tag{1}$$

The *diffusion tensor D* is a positive definite, symmetric matrix and describes the relation between ∇u (the *concentration gradient* at position $u(x,y,z)$) and the flux at each timestep t. *div* is the divergence operator $div\,\vec{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$. ∂u_t determines the modification of the image dependent on the time t. In the anisotropic diffusion equation, the diffusion tensor D is a matrix which controls the orientation and intensity of the flux. In the isotropic case a diffusivity function $g()$ is used instead, which controls the intensity of the flux.

Nonlinear Isotropic Diffusion

To reduce artifacts due to linear diffusion filtering (like blurring of important features or the dislocation of edges), in the nonlinear isotropic diffusion filtering a feedback system is established by adapting the diffusivity g to the gradient of the evolving image:

$$\partial_t u = div\,(g(\|\nabla u\|)\,\nabla u). \tag{2}$$

It depends strongly on the diffusivity $g(\|\nabla u\|)$ how efficient the blurring at edges is reduced and the localization is increased. $g(\|\nabla u\|)$ scales the gradient in the original image to stop the flux along edges, shown in Figure 3(right). Mainly we use the Perona–Malik model [5] (they call it anisotropic) defined in Equation 3

$$g(s) = e^{-(s^2/\lambda^2)}, \tag{3}$$

and the Tukey/Biweight model [1] defined in Equation 4. They turned out to be the most efficient models for the diffusivity $g(\|\nabla u\|)$.

$$g(s) = \begin{cases} \frac{1}{2}[1-(\frac{s}{\lambda})^2]^2 & , |s| \leq \lambda \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The parameter λ is the most critical value concerning the success of image smoothing and segmentation. Therefore we implemented a method to automatically estimate λ depending on the underlying density values described in Section 3.2.

Edge-Enhancing Anisotropic Diffusion

Instead of the scalar diffusivity $g()$, edge-enhancing anisotropic diffusion [12] uses a tensor D as shown in Equation 1. This tensor rotates and scales the flux in order to preserve interesting features (e.g. parallel to edges instead of smoothing across them preserves edges). Now, the flux is no longer parallel to ∇u which maintains the location and strength of edges.
In the edge-enhancing anisotropic diffusion equation the orthogonal system of eigenvectors v_1, v_2 and v_3 of the diffusion tensor D for a three dimensional volume can be written as follows:

$$D = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} v_1^\top \\ v_2^\top \\ v_3^\top \end{bmatrix} \tag{5}$$

where $v_1 \| \nabla u_\sigma$, $v_2 \perp \nabla u_\sigma$ and $v_3 \perp \nabla u_\sigma$. ∇u_σ is the gradient defined by using a Gaussian kernel with width σ. To obtain smoothing along the edges, the corresponding eigenvalues can be selected as $\lambda_2 = \lambda_3 = 1$. λ_1 is defined using one of the diffusivities in Equation 3 or 4. If $\lambda_1 = \lambda_2 = \lambda_3 = g()$ it results in the isotropic diffusion.

2.1 Discretization

In this section the discretization of the diffusion equations is explained.

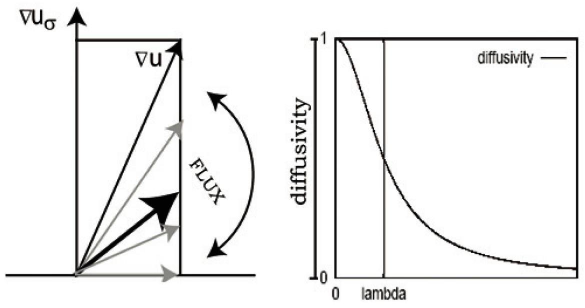


Figure 3: The diffusivity g() (in [0,1]; right) determines how much the flux direction deviates from the gradient ∇u. Near edges $g()$ approaches zero, and the flux direction becomes orthogonal to ∇u_σ.

Nonlinear Isotropic Diffusion

The discretization of the diffusion equation 2 is introduced in [5] and can be written as follows:

$$\begin{aligned} u^{t+1}_{x,y,z} &= u^t_{x,y,z} \\ &+ h \cdot \\ &\cdot \Big[g\left(\frac{u_{x-1,y,z}-u_{x,y,z}}{\Delta x}\right) \cdot \left(u^t_{x-1,y,z} - u^t_{x,y,z}\right) + \\ &+ g\left(\frac{u_{x+1,y,z}-u_{x,y,z}}{\Delta x}\right) \cdot \left(u^t_{x+1,y,z} - u^t_{x,y,z}\right) + \\ &+ g\left(\frac{u_{x,y-1,z}-u_{x,y,z}}{\Delta y}\right) \cdot \left(u^t_{x,y-1,z} - u^t_{x,y,z}\right) + \\ &+ g\left(\frac{u_{x,y+1,z}-u_{x,y,z}}{\Delta y}\right) \cdot \left(u^t_{x,y+1,z} - u^t_{x,y,z}\right) + \\ &+ g\left(\frac{u_{x,y,z-1}-u_{x,y,z}}{\Delta z}\right) \cdot \left(u^t_{x,y,z-1} - u^t_{x,y,z}\right) + \\ &+ g\left(\frac{u_{x,y,z+1}-u_{x,y,z}}{\Delta z}\right) \cdot \left(u^t_{x,y,z+1} - u^t_{x,y,z}\right) \Big], \end{aligned} \tag{6}$$

where $u_{x,y,z}$ denotes the density value of the volume at the location (x,y,z). In this discretization method, the divergence at time step t is calculated for each position (x,y,z) by applying the diffusivity $g()$ on the forward and the backward differences in each direction and then again calculating the forward differences of both values. The parameter h depends on the neighbourhood used for diffusion. This discretization scheme uses a neighbourhood of six voxels, but we also extend this equation for eighteen– and twenty-six neighbours by additionally fetching the respective neighbour voxels. Generally, when h is increased, evolution of the diffusion process speeds up. The discretization of the nonlinear isotropic diffusion is also often called anisotropic [5], because the diffusivity function $g()$ is applied separately on the different gradient directions. Figure 1 illustrates an example where the isotropic diffusion is used to smooth a hand dataset rendered DVR (direct volume rendering) shaded with raycasting on the GPU.

Edge-Enhancing Anisotropic Diffusion

In 3D, we denote the components of the diffusion tensor D as

$$D = \begin{pmatrix} a & d & e \\ d & b & f \\ e & f & c \end{pmatrix} \qquad (7)$$

and substitution in Equation 1 yields:

$$\partial u_t = \mathbf{div}(D \cdot \nabla u) = \mathbf{div} \begin{pmatrix} a\partial_x u + d\partial_y u + e\partial_z u \\ d\partial_x u + b\partial_y u + f\partial_z u \\ e\partial_x u + f\partial_y u + c\partial_z u \end{pmatrix}. \quad (8)$$

For discretization the inner derivatives are usually approximated by central differences. It is crucial to use smoothed gradients u_σ for the stencil calculation. Otherwise, the diffusion stencil would degenerate to the isotropic case because no rotation of the gradient takes place. Therefore, the gradients have to be calculated (with a Sobel or Gaussian kernel for example), in a preprocessing step. This gradient volume, has high values at locations where the intensity contrast is high (at edges) and low values at homogeneous regions. As shown in Figure 3 the precalculated gradient ∇u_σ at a specific location, has another orientation than the gradient at the same location from the original density volume ∇u obtained by central differences. The gradient ∇u is now scaled (because of D), depending on the diffusivity progression shown in Figure 3 (right side). If the region is homogeneous, the gradient is small and the flux is high, which means that the diffusivity is near one. Then the flux proceeds along the gradient direction ∇u. The more the diffusivity decreases to zero (at edges), the more the flux direction differs from ∇u. The gradient ∇u increases and the flow decreases and rotates towards the tangent-plane. This causes the rotation of the flux. The more different the precalculated gradients ∇u_σ are from ∇u, the more edge stopping is the function. So, the intensity and the direction of the flux, can be controlled by the smoothing kernel.

Considering the isotropic case, all λ parameters in Equation 5 are the same. Therefore, the gradient ∇u is scaled uniformly, which means that no flux rotation takes place. The propagation of the seeds depends only on the gradient ∇u and the diffusivity.

The semidiscretization of the convolution kernel can be written as a stencil. The stencil weights for the standard approximation of Equation 8 are in Tables 1 to 3. Here h_1, h_2 and h_3 denotes the stepsize in the x, y and z direction respectively, and a, b, c, d, e and f are the elements of the diffusion tensor D at their specified location (x,y,z).

In our implementation, we have to divide the calculations into two render passes, in order to cache the result of on-the-fly computation of smoothed gradients. In the first pass we calculate the gradients for one slice respectively. Therefore we use either a Gaussian or a Sobel kernel. In the second pass, the stencils shown in Tables 1, 2 and 3 are used together with the gradient volume to calculate the divergence. In the last step, the stencil is applied on the original density, added up and normalized. In Figure 2 the two algorithms are applied on a hand dataset and compared.

	$\frac{-f_{x,y,z-1}-f_{x,y+1,z}}{4h_2h_3}$	
$\frac{e_{x,y,z-1}+e_{x-1,y,z}}{4h_1h_3}$	$\frac{c_{x,y,z-1}+c_{x,y,z}}{2{h_3}^2}$	$\frac{-e_{x,y,z-1}-e_{x+1,y,z}}{4h_1h_3}$
	$\frac{f_{x,y,z-1}+f_{x,y-1,z}}{4h_2h_3}$	

Table 1: The anisotropic diffusion stencil for the $z = -1$ position.

$\frac{-d_{x-1,y,z}-d_{x,y+1,z}}{4h_1h_2}$	$\frac{b_{x,y+1,z}+b_{x,y,z}}{2{h_2}^2}$	$\frac{d_{x+1,y,z}+d_{x,y+1,z}}{4h_1h_2}$
$\frac{a_{x-1,y,z}+a_{x,y,z}}{2{h_1}^2}$	$-\frac{a_{x-1,y,z}+2a_{x,y,z}+a_{x+1,y,z}}{2{h_1}^2}$ $-\frac{b_{x,y-1,z}+2b_{x,y,z}+b_{x,y+1,z}}{2{h_2}^2}$ $-\frac{c_{x,y,z-1}+2c_{x,y,z}+c_{x,y,z+1}}{2{h_3}^2}$	$\frac{a_{x+1,y,z}+a_{x,y,z}}{2{h_1}^2}$
$\frac{d_{x-1,y,z}+d_{x,y-1,z}}{4h_1h_2}$	$\frac{b_{x,y-1,z}+b_{x,y,z}}{2{h_2}^2}$	$\frac{-d_{x+1,y,z}-d_{x,y-1,z}}{4h_1h_2}$

Table 2: The anisotropic diffusion stencil for the $z = 0$ position.

	$\frac{f_{x,y,z+1}+f_{x,y+1,z}}{4h_2h_3}$	
$\frac{-e_{x,y,z+1}-e_{x-1,y,z}}{4h_1h_3}$	$\frac{c_{x,y,z+1}+c_{x,y,z}}{2{h_3}^2}$	$\frac{e_{x,y,z+1}+e_{x+1,y,z}}{4h_1h_3}$
	$\frac{-f_{x,y,z+1}-f_{x,y-1,z}}{4h_2h_3}$	

Table 3: The anisotropic diffusion stencil for the $z = +1$ position.

3 Application to Medical Segmentation

The previous explained smoothing equations, are used in a so called *hybrid method* together with region growing for segmentation purposes.

3.1 Diffusion-Based Region Growing

In general, region-based techniques assume that neighbouring voxels within the same region are homogeneous

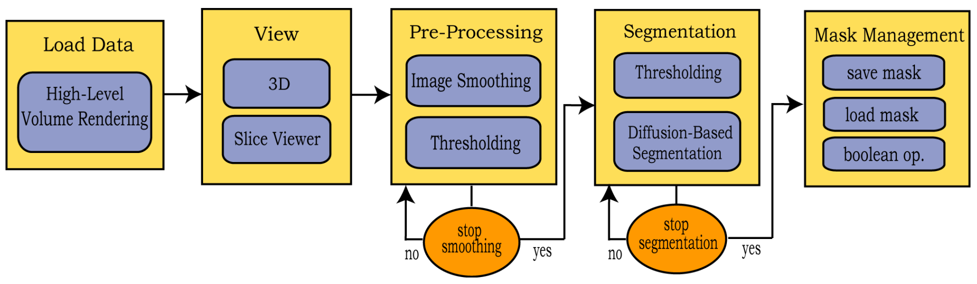


Figure 4: The **segmentation pipeline** demonstrates the whole segmentation process.

with respect to a defined, prespecified criterion (e.g. similar intensity values). This homogeneity criterion is the minimum difference between the voxel's value (the initial *seed*) and the average value of the region where the next pixel should be appended. Starting from one (or multiple) initial points (i.e. voxels), either user defined or automatically estimated, the seeds evolve in each iteration of the algorithm as long as this homogeneity criterion is satisfied. This yields closed regions in the final segmentation. Therefore, the results strongly depend on the initial choice of the seeds.

The advantage of using a hybrid of seeded region growing and diffusion filtering is, that region properties are not tracked explicitly, but diffusion of seeds can be controlled by partial differential equations (PDEs). This technique operates locally by solving a large amount of PDEs and is therefore well-suited to be implemented on GPUs. Usually diffusion filters are applied directly to an intensity image, but in case of diffusion-based segmentation approaches the actual diffusion equation is applied to the seed image, incorporating the intensity image as additional term into the equation.

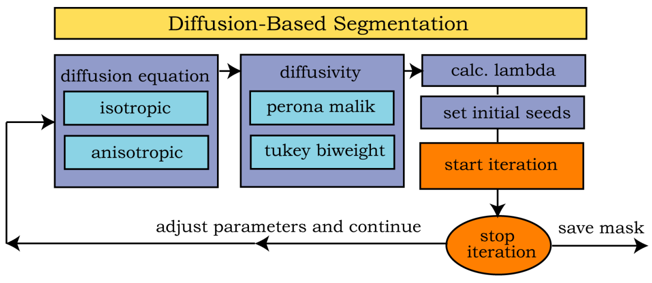


Figure 5: The selection possibilities for the diffusion-based segmentation process.

3.2 Segmentation Pipeline

The main advantage of this segmentation process is the interactivity provided by the fast, hardware-based implementation. So the user can observe and influence the smoothing and segmentation process anytime the segmentation does not develop as desired. The user can stop the process, change the parameter settings, paint new seeds or delete wrong ones and continue the process as often as necessary to obtain a satisfying solution. The possible steps of the whole segmentation process are demonstrated in Figure 4 and described below. The possible algorithm-settings especially for the segmentation are demonstrated in Figure 5.

Parameter Setting

The parameter λ is the most critical value concerning the success of image smoothing and segmentation. In [2] the filtering process is integrated in a closed-loop system, to determine λ and the number of iterations automatically. Therefore the results of the filtering are analysed in order to adjust the parameters before the next iteration. In our approach, the number of iterations is controlled interactively. To estimate λ, we use the fact that for the flow in image smoothing, the proportion between $\|\nabla u\|$ and λ stays the same in all scales. The flow of the Perona–Malik diffusivity (Equation 3) reaches its maximum at $\|\nabla u\| = \frac{1}{\sqrt{2}}\lambda \approx 0.71\lambda$ and of the Tukey/Biweight diffusivity (Equation 4) at $\|\nabla u\| = \frac{1}{\sqrt{5}}\lambda \approx 0.45\lambda$. ∇u is calculated by building the difference of two density values from each side of the edge. The user can estimate them interactively by clicking on them in the slice viewer.

Volume Smoothing

Nonlinear diffusion filtering can be applied to images or volumes (e.g. medical data), to remove noise and enhance edges. We mainly use smoothing as a preprocessing step for segmentation, but it can also be used in the ray-caster [7] for different visualization purposes.

Segmentation

Thresholding approaches expect one or more intensity values, the *thresholds* and segment a scalar image by creating a binary partition of the image intensities. This is achieved by grouping all pixels with intensity greater than the threshold into one class, and all other pixels into the second class. Thresholding is a good basic method to gain a fast segmentation in images where different structures have contrasting intensities, or used as an initial step in a sequence of imaging processing [11, 9].

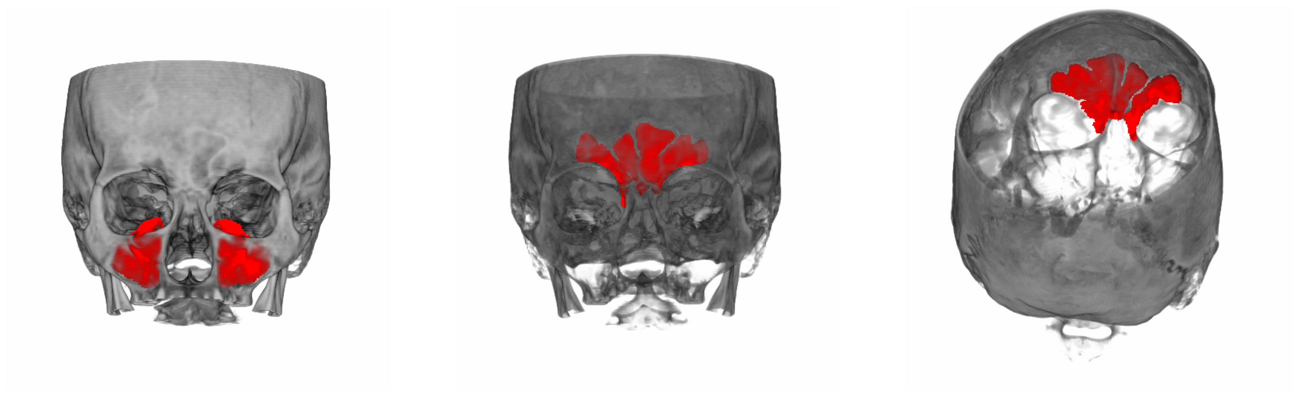

(a) Segmentation of the paranasal sinuses. (b) Segmentation of the frontal sinus. (c) Back view of the segmented frontal sinus.

Figure 6: The segmented parts using diffusion-based seeded region growing are displayed in red (dark gray). These parts are saved as separate masks.

For **diffusion-based volume segmentation** the initial user input (seeds) is saved in a seed mask s, a second volume (in addition to the density volume) where each voxel has a corresponding seed point. The merging criteria for the neighbouring seeds is based on diffusion filtering, where the diffusion flow still depends on the underlying image but is multiplied with the gradient magnitude of the seed mask. Equation 1 can be modified for the seed flow:

$$\partial_t s = div\,(g(\|\nabla u\|)\,\nabla s), \tag{9}$$

where u indicates the image and s the seed mask. The seeds progress along the flux until their expansion is stopped at the edges, where the flow is slowed down. The seeds expand fast if the seedflow in Equation 9 is large. For the discretization of Equation 9, Equation 6 has to be modified for the seedmask. Instead of the density volume, $s^t_{x,y,z}$, the number of seeds in the seed mask at time step t has to be calculated. Just for the calculation of the diffusivities, the original density values u are taken. The stencil for the anisotropic diffusion shown in Tables 1 to 3, can also be modified for segmentation purposes when applied on the seedmask s.

Mask Operations

If the user is satisfied with a partial segmentation result, he can save the segmented part as a mask. In already segmented volumes the data values are assigned to different objects. To indicate the membership of each voxel, an object ID volume is created that contains the object ID for each voxel, as described in [4]. These objects are saved in binary masks on the CPU and can be arbitrary assembled together anytime. To achieve a quick update of the object mask during the segmentation process, the most significant bit of the object ID volume is used, to indicate whether the voxel is part of the currently segmented volume or not.
Boolean operators can also be applied on combinations of already existing masks to generate new segmented parts.

4 Implementation

The GPU is optimized for high throughput of data and therefore well suited for solving diffusion equations which are based on partial differential equations. These equations are implemented in *fragment shaders*, which are user-written programs that are executed once for each fragment on the GPU and responsible for the final colour and opacity of each fragment. Our fragment shaders for the diffusion filtering and volume segmentation, are written in Cg a high level shading language developed by NVIDIA. What makes Cg and other shading languages different from conventional program languages is that they are all based on a dataflow computational model, where computation occurs in response to data that flows through a sequence of processing steps as described in [3]. Cg removes the need to program with hardware assembly language and provides a complete programming platform that is easy to use and similar to C. The reason why we preferred Cg over HLSL or the OpenGL Shading Language, is because it provides the use of interfaces. Using interfaces makes it possible to write different implementations of various algorithms in one shader and choosing the desired ones at compile time. This allows, to exclude as much as possible from the Cg code that is not used in the specific – user defined – situation, from being compiled by the Cg compiler, which leads to reduced execution time. The compilation into GPU assembly code can take place either in advance or on demand at run time. This is very important when using interfaces, because the actual interface implementations can be connected during run-time.

Performance

Table 4 shows the time in seconds used for one iteration for the different algorithms respectively. One iteration updates and renders the whole volume. Especially the performance of the anisotropic case is very high considering the amount of calculations performed on the whole volume. In

dataset size	isotropic	anisotropic
256x128x256	0.058	0.51
512x512x128	0.179	1.873

Table 4: This table shows the time in seconds used for one smoothing iteration. Both algorithms are used with the Perona–Malik diffusivity (Equation 3).

one iteration eighteen smoothed gradients of an 3x3 kernel in a 3x3 neighbourhood have to be calculate. Then 18 different tensors are calculated for the 3x3x3 convolution kernel and finally applied on the whole density volume. Each iteration updates and renders the whole volume.

5 Segmentation Example

This section illustrates a possible segmentation process of a 512x512x128 CT-head dataset, downsampled to 256x256x128.
After the dataset is loaded in the volume renderer [4], it can be observed and modified in the *3D view* and in the *slice viewer*. A main advantage is, that the development occurred during different operations applied on the dataset, can be immediately tracked in the 3D view as well as in the slice viewer. So the user has full control over the evolution of the volume smoothing and segmentation, can stop the process and change the parameters whenever necessary (Figure 5). The following steps are applied on the dataset:

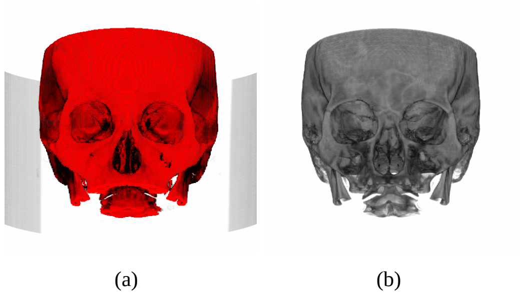

(a) (b)

Figure 7: The segmented part using thresholding is displayed in Figure 7(a) in red. This part can be saved and reloaded as a mask, Figure 7(b).

1. **Smoothing** is used as pre-processing to reduce noise which often occurs in medical datasets. Before starting the smoothing process, we have to choose one of the diffusivity Equations 3 or 4. This is also important for the parameter λ, because the calculation depends on the conditions described in Section 3.2. Therefore, the gradient is determined by estimating the different density values along an edge to be preserved/enhanced during smoothing. We use the Tukey/Biweight diffusivity (Equation 4) together with the isotropic diffusion 2 and for $\lambda = 0.245529$. After about 30 iterations we get the result smoothed enough to continue with segmentation. All in all this takes about 1.74 seconds.

2. **Thresholding** can be used to get a fast segmentation when different structures have contrasting intensities. Therefore, we use thresholding to segment the bones of the skull, as shown in Figure 7(a). Here we use one threshold from 191 to 3000 HU (Hounsfield Units).

3. **Save mask** is now used to save the segmented part from Figure 7(a) as a separate mask shown in Figure 7(b). A main advantage of using masks is, that a separate transfer function and render mode can be applied to every single mask, as shown in Figure 8. The objects can be arbitrarily enabled an disabled for better visualization purposes, demonstrated in Figure 8(a) where only the segmented paranasal sinuses and the frontal sinus are visible.

4. We use **diffusion-based segmentation** to segment the paranasal sinuses. Therefore, we take the same parameters as already used for smoothing purposes. This is only reasonable if the original dataset was not too noisy. Otherwise the density values changes too much after smoothing– then a new lambda value has to be estimated. The parameters turned out to be a good choice, because after setting the initial seeds and starting the region growing process, the dispersion of the seeds stops automatically after about 200 iterations which needs about 11.6 seconds. The segmentation result is displayed in red in Figure 6(a).

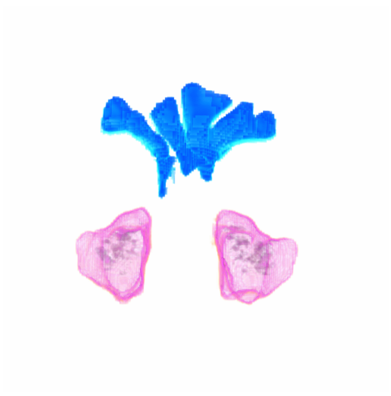
5. **Save mask** is again used to save the segmented paranasal sinuses.

6. Applying **transfer functions** before the segmentation process can sometimes lead to better results. The differences in density can be enhanced by windowing the intensity input, which causes in better edge detection of the desired regions. For segmenting the frontal sinus, we first apply a transfer function which enhances the bone compared to the soft tissue. This leads to the darker appearance of the bone structure shown in Figure 6(b) and Figure 6(c).

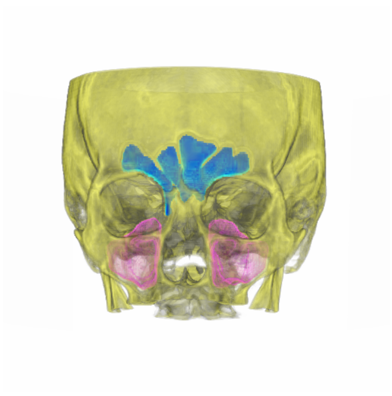
7. The **diffusion-based segmentation** of the frontal sinus proceeds as before, with the difference that we have to stop the segmentation process after about 300 iterations by hand. Otherwise the dispersion of the seeds would continue into the nasal cavity. The results of this segmentation step are shown in Figures 6(b) and 6(c).

8. **Save Mask** to store the segmented frontal sinus.

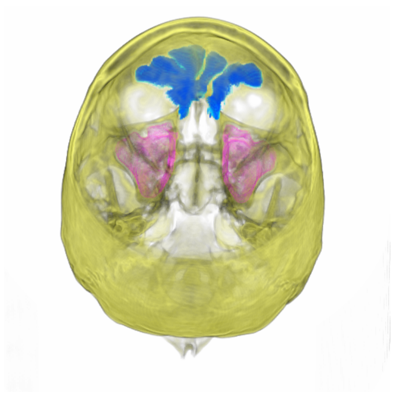
9. For the final **data processing**, we load the three masks obtained in the segmentation steps described before. Then we update the object IDs, as already described in Section 3.2, to obtain one volume with



(a) Different transfer functions applied on the masks of the paranasal sinuses and the frontal sinus.



(b) Different transfer functions applied on all segmented object masks.



(c) The back view of the object from Figure 8(b).

Figure 8: For better visualization purposes, different transfer functions are applied on the different object masks.

the segmented masks. For visualization purposes, we apply a separate transfer function on each mask. The final segmentation result can be view in Figures 8(b) and 8(c).

6 Conclusions

This paper demonstrates that diffusion-based methods are well suited for the implementation on GPUs. Nevertheless, the results show that diffusion is easier applicable for image smoothing than for segmentation purposes. If the density values have a high difference in contrast satisfying results are easily achieved. Then the intensity is high at the edges, the parameters are easy to estimate and the segmentation succeeds. But when the edges are weak, it is often challenging to estimate fitting parameters. When the dispersion of seeds does not stop, the seeds run out and flood also other regions. Therefore this segmentation method is well suited for combining with other methods like windowing (as described above). In combination, satisfying results can also be achieved with objects with weak edges. Image smoothing is useful as preprocessing step for segmentation but also for many other applications. Generally, the anisotropic diffusion turned out to be more edge enhancing than the isotropic diffusion (Figure 2). But the parameter λ and the number of iterations used to reach a good smoothing result, differ completely depending on the diffusion equation an the underlying data. So it is difficult to give a meaningful comparison.

Acknowledgements

I want to thank Katja Bühler for the possibility to accomplish this internship at VRVis. Special thanks go to my advisor Markus Hadwiger for his permanent encouragement helping me with my work and for everything I learned during the time at VRVis.

References

[1] M. J. Black, G. Sapiro, D. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Trans. on Image Processing*, 7(3):421–432, 1998.

[2] J. Castellanos, K. Rohr, T. Tolxdorff, and G. Wagenknecht. De-noising MRI Data – An Iterative Method for Filter Parameter Optimization. In *Proc. Workshop Bildverarbeitung für die Medizin 2005*, pages 40–44, 2005.

[3] Radima Fernando and Mark J. Kligard. *The Cg Tutorial*. Addison-Wesley, 3rd edition, 2003.

[4] M.Hadwiger, C.Berger, and H.Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proceedings of IEEE Visualization 2003*, pages 301–308, 2003.

[5] P. Perona and J. Malik. Scale-Space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7), 1990.

[6] Martin Rumpf and Robert Strzodka. Nonlinear Diffusion in Graphics Hardware. pages 75–84.

[7] Henning Scharsach. Advanced GPU Raycasting. In *Proceedings of CESCG 2005*, pages 69–76, 2005.

[8] Stefan Schenke, Burkhard C. Wuensche, and Joachim Denzler. GPU-Based Volume Segmentation. In *Proceedings of IVCNZ 2005, Dunedin, New Zealand*, pages 171–176, 2005.

[9] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13:146–165, January 2004.

[10] Anthony Sherbondy, Michael Houston, and Sandy Napel. Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In *IEEE Visualization*, pages 171–176, 2003.

[11] K. Udupa and G. Herman. *3D Imaging in Medicine*. CRC Press, 1999.

[12] J. Weickert. *Anisotropic Diffusion in Image Processing*. PhD thesis, University Kaiserslautern, 1996.

[13] J. Weickert. A review of nonlinear diffusion filtering. *Scale Space Theories in Computer Vision*, pages 3–28, 1997.