Computer Vision Winter Workshop 2008, Janez Perš (ed.)
Moravske Toplice, Slovenia, February 4–6
Slovenian Pattern Recognition Society, Ljubljana, Slovenia

Continuous Globally Optimal Image Segmentation with Local Constraints

Markus Unger, Thomas Pock, and Horst Bischof

Institute for Computer Graphics and Vision,
Graz University of Technology, Austria
{unger, pock, bischof}@icg.tugraz.at

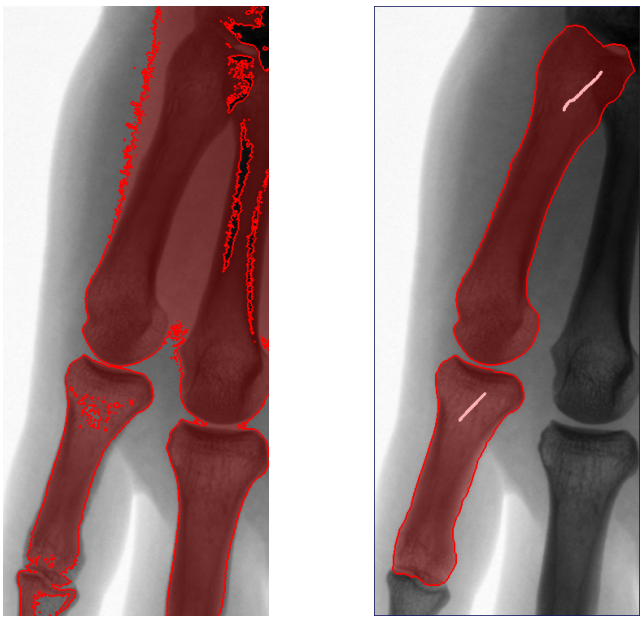
Abstract *The Geodesic Active contour model is a very flexible model for variational image segmentation. Unfortunately the Geodesic Active Contour model exhibits local minima making segmentation results strongly dependent on its initialization. We propose a flexible, interactive segmentation method in two and three dimensions that yields the globally optimal solution with respect to local constraints introduced by the user. A fast numerical scheme is used to minimize the proposed energy which is based on a weighted Total Variation energy functional. With our GPU-based implementation, real-time performance is achieved for both 2D and 3D segmentation problems. We show experimental results on various medical datasets, and discuss the properties of the segmentation framework.*

1 Introduction

Image segmentation is one of the most fundamental problems in computer vision. It aims at partitioning an image into a set of non-overlapping (disjoint) regions. Segmented images are further used as input for various applications such as classification, recognition and measurement. More specifically, in the case of medical applications image segmentation is an important step towards the study of anatomical structures, diagnosis and the planning of surgeries or other forms of treatment [19].

Many approaches have been developed to tackle the problem of image segmentation. One of the most basic approaches is to rely on homogeneity criteria inside the object of interest (e.g. uniform intensity distribution). The most simple algorithm which relies on such an assumption is segmentation by thresholding [18]. As a matter of fact this method often fails, if the desired object can not be characterized solely by intensity distributions. Unfortunately, medical images often suffer from such problems. Fig. 1 depicts a typical medical image, where a user selected intensity range is insufficient to separate the bones from the background. In contrast, the method proposed in this paper is solely based on edge information. By providing a little high-level information (seed regions), our method can easily segment the bones. Moreover, it is quite obvious that the problem of image segmentation is highly ambiguous. In many cases it would therefore be necessary to incorporate some information from the user into the segmentation process.

Our aim is therefore not to develop a fully automatic segmentation algorithm, but to provide a flexible framework to the user allowing to incorporate high-level knowledge. The major advantage of our approach is that our algorithm is guaranteed to yield a globally optimal solution with respect to the constraints provided by the user. Our algorithm is based on continuous optimization techniques. In contrast to the very popular graph cut methods our method does not suffer from any metrication errors. Moreover, our algorithm can effectively be accelerated by implementation on parallel architectures such as graphics hardware leading to real-time performance. In summary, we propose an interactive general purpose segmentation tool for both 2D and 3D problems.



(a) Thesholding (b) Proposed method

Figure 1: Segmentation of a typical medical image.

The remainder of the paper is organized as follows: In Section 2 we review the current state-of-the-art in image segmentation. In Section 3 we describe our variational segmentation model and will develop an efficient algorithm which allows to compute the global minimizer of it. Moreover we will show that our segmentation model is a generalization of the continuous maximal flow algorithm of Appleton and Talbot [1]. In Section 4 we give details about implementation issues. In fact, we show that the algorithm can be

effectively accelerated by implementing it on the graphics processing unit (GPU). This leads to real-time performance for both 2D and 3D segmentation problems. In Section 5 we present various segmentation examples showing the effectiveness of our method. In the last Section we give some conclusions and discuss possible directions for future investigations.

2 Related Work

In the following previous work is discussed, and its relation to the presented segmentation approach is shown.

2.1 The Active Contour/Snake Model

One of the major advances in image segmentation was the introduction of the Active Contour/Snake model by Kass, Witkin and Terzopoulos in [14]. Here, a contour is deformed according to internal and external forces. A standard approach to implement the curve evolution is via the level set approach of Osher and Sethian [16]. However, the curve evolution approach solely acts locally and is therefore prone to get stuck into a local minimum. To overcome this one has to put the initial contour very close to the final segmentation.

2.2 Geodesic Active Contours

Geodesic active contours (GAC) in 2D and minimal surfaces in 3D were introduced by Caselles et al. in [6, 7], as an enhanced version of the snake model of Kass et al. [14]. The GAC model is defined as the variational problem:

$$\min_C \left\{ E_{GAC}(C) = \int_0^{|C|} g\left(|\nabla I(C(s))|\right) dl \right\} , \quad (1)$$

where $|C|$ is the Euclidean length of the the curve C and dl is the Euclidean element of length. The function $g \in (0, 1]$ is an edge detection function which is close to 0 at strong edges in the image I. A quite common choice is

$$g\left(|\nabla I|\right) = \exp\left(-\alpha \left|\nabla I\right|^{\beta}\right) , \quad (2)$$

for some suitable parameters α and β. The GAC model therefore integrates up the Euclidean element of length dl weighted by a term depending on the boundary information in the image. Moreover, Caselles et al. showed that minimizing (1) is equal to finding a geodesic curve in a Riemannian space. Note that the trivial solution $C = \emptyset$ is always a global minimizer of this energy. Therefore the Geodesic Active Contour model is only meaningful in combination with certain constraints (e.g predefined foreground and background seed regions).

In order to minimize E_{GAC}, the standard approach is to apply the gradient descend method to the Euler-Lagrange equation of the GAC model:

$$\frac{\partial C(t)}{\partial t} = g(I)\kappa\mathcal{N} - \langle \nabla g, \mathcal{N} \rangle \mathcal{N} , \quad (3)$$

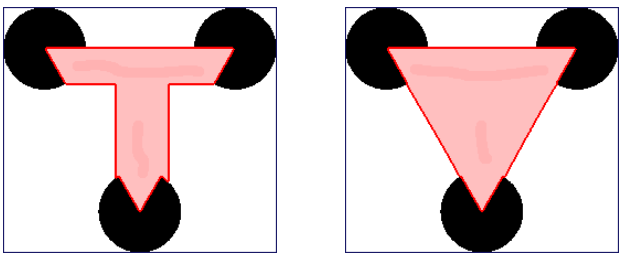
where κ is the curvature of C and $\mathcal{N}$ is the unit normal to C. Similar to the Active Contour/Snake model, the level set method can be used to implement the evolution process, however it also does not yield a global optimum.

2.3 Graph Based Approaches

Graph based approaches have also been used to find a solution for the GAC model. Graph based methods rely on the partitioning of a graph that is build to match the corresponding image. An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ contains a set of nodes $\mathcal{V}$ corresponding to pixels (in 2D) or voxels (in 3D), and a set of undirected edges $\mathcal{E}$. The connectivity of the graph is based on the chosen neighborhood system. Based on the image data, a non-negative weight w_e is assigned to each edge. These weights correspond to the weighted length of the GAC model. Additionally there are other nodes, called terminals, representing the background and foreground seed regions. To find a solution of the graph based approach, a minimum cut $C \subset \mathcal{E}$ is computed [3] which separates the graph around the terminals. The energy of this cut is obtained by summing up all edges which are intersected by the cut:

$$|C|_{\mathcal{G}} = \sum_{e \in C} w_e . \quad (4)$$

It is well-known that the quality of this approximation, depends highly on the order of connectivity of the underlying graph structure. In fact, Boykov et al. showed that for successively finer grids, the discrete approximation of the contour length $|C|_{\mathcal{G}}$ approaches the Euclidean length of the contour. Note that in order to obtain a reasonably good approximation, a high degree of connectivity is needed. Graph based methods suffer from a metrication error induced by the neighbourhood system. Fig. 2 shows the effects of a typically used 4-neighbourhood.



(a) 4-neighbourhood (b) Our approach

Figure 2: Effects of metrication error.

Another graph based approach was taken by Grady. The random walker algorithm [13] starts with a random walker at each unlabeled pixel, and then finds the probability that it first reaches one of the foreground-background terminals. In order to obtain the final segmentation, each pixel is assigned the most probable seed destination. The probability of a random walker is obtained by the solution of a weighted Dirichlet problem [13].

$$D[u] = \frac{1}{2} \int_{\Omega} g(x) \left|\nabla u\right|^2 . \quad (5)$$

Note that the Dirichlet problem is quadratic and therefore easy to solve, but on the other hand does not correspond to the weighted length of the contour. Grady showed how to set up the graph correctly and derived a sparse linear equation

system that has to be solved. This system can be solved in closed form by a LU decomposition or with iterative methods like the multigrid method or the method of conjugate gradients.

2.4 Continuous Maximal Flows

In [1], Appleton and Talbot presented an approach to compute minimal surfaces using continuous maximal flows. The continuous maximal flow system is defined as following:

$$\frac{\partial P}{\partial t} = -\nabla \cdot \mathbf{F} \,, \tag{6}$$

$$\frac{\partial \mathbf{F}}{\partial t} = -\nabla P \,, \tag{7}$$

subject to

$$|\mathbf{F}| \leq g \,. \tag{8}$$

P is a scalar potential field and $\mathbf{F}$ is a vector flow field, that is evolving over time. The foreground can be defined as source by $P(x) = 1$ and the background as a sink by $P(x) = 0$. With (6) the potential P is updated according to the flow field $\mathbf{F}$. (7) makes the flow dependent on the gradients in P. Together they form a system of wave equations. The constraint (8) on the magnitude of $\mathbf{F}$ regulates the diffusion process according to the Riemmanian metric g. At strong borders the propagation is slowed down, while in flat regions evolution is almost unopposed. The algorithm is implemented as an iterative scheme with an artificial timestep $\Delta t < \frac{1}{\sqrt{D}}$ for D-dimensional images. The final segmentation is obtained as the 0.5 levelset of P.

2.5 Weighted Total Variation

In [4, 5], Bresson et al. introduced the g-weighted Total Variation Norm

$$TV_g(u) = \int_\Omega g(x)|\nabla u| d\Omega \,. \tag{9}$$

He showed that if u is a characteristic function 1_C, (9) is equivalent to E_{GAC} in (1). Note that the characteristic function 1_C is a closed set in the image domain Ω and C stands for its boundary. The promise of this formulation is that if u is allowed to vary smoothly between $[0, 1]$, (9) becomes a convex functional, meaning that one can compute the global minimizer of it. The final segmentation can in turn be extracted from u by selecting a level set in $[0, 1]$.

As mentioned above, the trivial solution $u = const$ is always a global minimizer of (9). To account for this, one has to restrict the space of possible solutions by incorporating some constraints. In [4], Bresson et al. coupled the minimization of (9) with the piecewise Mumford Shah functional [10]. This effectively prevents the GAC model from yielding a trivial solution but on the other hand reduces the flexibility of the model. In [15], Leung and Osher unified denoising, segmentation and inpainting. The idea is to use (9) together with a spatially varying L^1 data fidelity term.

Instead of coupling the minimization of the GAC model with some unsupervised data driven constraints we propose to include local constraints provided by the user into the functional (9). We will describe this in the next Section.

3 Method

In this Section we will first formulate our variational image segmentation model. Then we will present a fast algorithm to compute the minimizer of our image segmentation model. Furthermore, we will show that our algorithm can actually be seen as a generalization of the continuous max-flow algorithm of Appelton and Talbot [1].

3.1 Proposed Segmentation Model

We propose to minimize the following variational image segmentation model:

$$\min_{u \in [0,1]} \left\{ \int_\Omega g(x)|\nabla u| d\Omega + \frac{1}{2} \int_\Omega \lambda(x) \left(u - f\right)^2 d\Omega \right\} \,. \tag{10}$$

The first term of the energy is exactly the g-weighted Total Variation of u (see also (9)). Note that u is no longer a characteristic function but can vary continuously between $[0, 1]$. The function $f \in [0, 1]$ is provided by the user and contains information about foreground ($f = 1$) and background ($f = 0$) seed regions. The spatially varying parameter $\lambda(x)$ is used to give a certain weight to the information contained in f. Note that for ($\lambda = \infty$) the information in f is said to be a hard constraint and for ($\lambda = 0$) the information in f is not used. In addition we allow that the edge detection function $g(x)$ can also be altered by the user, i.e. edges can be deleted or added in an interactive manner. Finally, we mention that the constraint $u \in [0, 1]$ can be omitted since $f \in [0, 1]$.

Our segmentation functional has a close connection to the Total Variation denoising model of Rudin, Osher and Fatemi (ROF) [17]. However, there are some significant differences between (10) and the original ROF model. First, our functional relies on the g-weighted Total Variation, second the parameter λ is spatially varying.

3.2 Computing the Solution

The standard approach to minimize our model is to solve the Euler-Lagrange equation associated with (10)

$$-\nabla \cdot \left(g(x) \frac{\nabla u}{|\nabla u|} \right) + \lambda(x) \left(u - f\right) = 0 \,. \tag{11}$$

When looking at this equation, one can make two observations: First, due to the $\frac{1}{|\nabla u|}$ term, the equation is highly non-linear. Therefore, we can not expect to find a closed-form solution to it. Second, the equation is not defined for $\nabla u = \mathbf{0}$. To overcome this limitation, a simple and commonly used approach is to replace $|\nabla u|$ by a regularized version $|\nabla u|_\varepsilon = \sqrt{|\nabla u|^2 + \varepsilon}$. However, for small ε the equation is still nearly degenerated and for larger ε the properties of the model are lost.

In [8, 9], Chambolle proposed a simple fixed point algorithm in order to solve the original ROF model. The advantage of Chambolle's approach is that no ε-regularization of the Total Variation term is needed. Unfortunately, Chambolle's approach can not be used with spatially varying parameter $\lambda(x)$. To account for this, we introduce an auxiliary variable v and propose to minimize the following approxi-

mation of (10):

$$\min_{u,v} \left\{ \int_\Omega g(x)|\nabla u| d\Omega + \frac{1}{2\theta} \int_\Omega (u-v)^2 \, d\Omega + \frac{1}{2} \int_\Omega \lambda(x) \, (v-f)^2 \, d\Omega \right\} . \quad (12)$$

We first note that as $\theta \to 0$, (12) approaches (10). Moreover we note that (12) is still convex. This means that we can compute the global minimizer of it. However, unlike (10), (12) is now an optimization problem in two variables, u and v. Therefore we have to perform an alternating minimization with respect to u and v. The outline of the alternating minimization procedure is as follows:

1. For fixed v, solve (12) for u.

$$\min_u \left\{ \int_\Omega g(x)|\nabla u| d\Omega + \frac{1}{2\theta} \int_\Omega (u-v)^2 \, d\Omega \right\} . \quad (13)$$

We can see that this optimization problem is exactly the ROF model, where θ is now a spatially constant regularization parameter. The only difference to the original ROF model lies in the g-weighting of the Total Variation norm, which does not induce any additional complexity. We can therefore use an adapted version of the projected gradient descend algorithm presented in [9] in order to solve this sub-optimization-problem exactly.

$$\begin{aligned} \tilde{\mathbf{p}}^{n+1} &= \mathbf{p}^n + \frac{\tau}{\theta} \nabla u^n \\ \mathbf{p}^{n+1} &= \frac{\tilde{\mathbf{p}}^{n+1}}{\max\left\{1, \frac{|\tilde{\mathbf{p}}^{n+1}|}{g}\right\}} \\ u^{n+1} &= v^n + \theta \nabla \cdot \mathbf{p}^{n+1} , \end{aligned} \quad (14)$$

where $\mathbf{p}$ is the so-called dual variable and τ is the time step which ensures that the scheme remains stable. In [9] it has been shown that for D-dimensional problems $\tau \leq \frac{1}{2D}$. We do not need to exactly solve this sub-optimization-problem. One iteration of this scheme is sufficient to make the entire algorithm converge.

2. For fixed u, solve (12) for v.

$$\min_v \left\{ \frac{1}{2\theta} \int_\Omega (u-v)^2 \, d\Omega + \frac{1}{2} \int_\Omega \lambda(x) \, (v-f)^2 \, d\Omega \right\} \quad (15)$$

is a point-wise convex minimization problem and therefore easy to solve. The Euler-Lagrange equation for (15) is given by:

$$v - u + \lambda(x)\theta(v-f) = 0 . \quad (16)$$

which can be solved in closed form via

$$v^{n+1} = \frac{u^{n+1} + \lambda(x)\theta f}{1 + \lambda(x)\theta} . \quad (17)$$

3. Goto 1. until convergence.

The algorithm presented above is valid for an arbitrary $\lambda(x)$. However, only two values of λ are meaningful. For a fixed $\theta > 0$ we can identify the following to cases:

1. $\lambda = \infty$ accounts for fixed foreground or background seeds provided by the user. Therefore the second step of our algorithm is given by $v = \lim_{\lambda\to\infty} \frac{u+\lambda\theta f}{1+\lambda\theta} = f$.
2. $\lambda = 0$ enables the algortihm to minimize the Geodesic Active contour energy and hence $v = \lim_{\lambda\to 0} \frac{u+\lambda\theta f}{1+\lambda\theta} = u$.

Fig. 3 shows the evolution process of the variable u during the minimization of the proposed segmentation model. Before running the minimization algorithm the variable u was initialized to $u = 0.5$ and the dual variable $\mathbf{p}$ was initialized to $\mathbf{p} = \mathbf{0}$. The approximation parameter θ was set to 0.5. The edge detection function was chosen as in (2). Fig. 3(a) shows the constraints f as provided by the user. The border was set to be background, the foreground regions were drawn by the user. Specified regions in f correspond to $\lambda = \infty$, unspecified (grey) regions correspond to $\lambda = 0$. Fig. 3(b) and Fig. 3(c) show intermediate steps of the iterative minimization algorithm, Fig. 3(d) shows the final result of u.

3.3 Connection to Continuous Maximal Flows

We now show that in case of $\lambda = 0$ our algorithm is equivalent to the continuous maximal flow algorithm of Appleton and Talbot [1].

Let us first write down our algorithm for. For $\lambda = 0$, the second step of our algorithm given by $v = u$. We can therefore simplify our algorithm as

$$\begin{aligned} \tilde{\mathbf{p}}^{n+1} &= \mathbf{p}^n + \frac{\tau}{\theta} \nabla u^n \\ \mathbf{p}^{n+1} &= \frac{\tilde{\mathbf{p}}^{n+1}}{\max\left\{1, \frac{|\tilde{\mathbf{p}}^{n+1}|}{g}\right\}} \\ u^{n+1} &= u^n + \theta \nabla \cdot \mathbf{p}^{n+1} . \end{aligned} \quad (18)$$

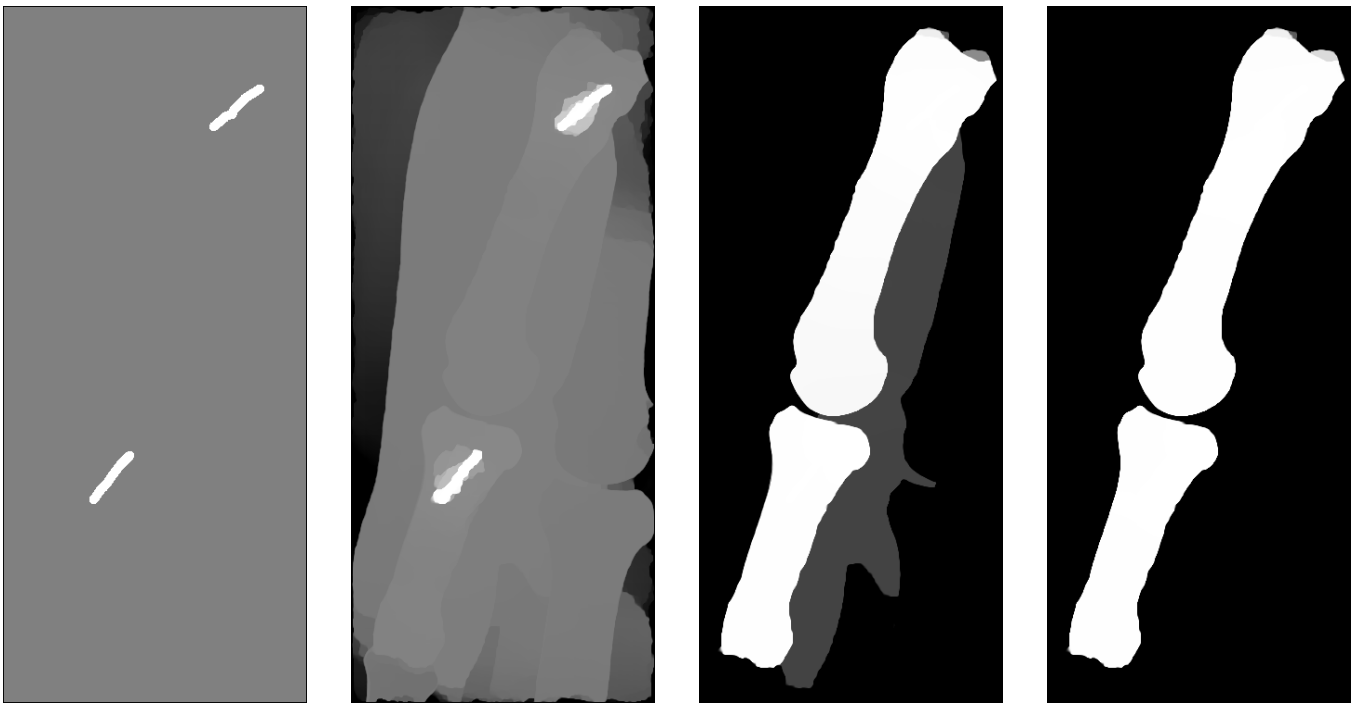
Based on (6), (7) and (8) let us now write down the iterative algorithm of Appleton and Talbot:

$$\begin{aligned} P^{n+1} &= P^n + \Delta t \nabla \cdot \mathbf{F}^n \\ \mathbf{F}^{n+1} &= \mathbf{F}^n + \Delta t \nabla P^{n+1} \\ |\mathbf{F}^{n+1}| &\leq g . \end{aligned} \quad (19)$$

It is now easy to see that for $\Delta t \equiv \theta \equiv \frac{\tau}{\theta}$ and up to a ordering of the updates both schemes are equivalent. Therefore, the continuous maximal flow algorithm of Appleton and Talbot essentially computes the minimizer of the weighted Total Variation functional. In other words, the projected gradient descend algorithm [9] is in principle equivalent to maximal flow algorithm of Appleton and Talbot.

4 Implementation

In this Section we discuss the implementation on the graphics device. We show the benefits of parallel hardware, and address some performance issues. Further the importance of the graphical user interface is discussed.



(a) Initialization (b) Intermediate Step 1 (c) Intermediate Step 2 (d) Final Result

Figure 3: Evolution process of the continuous variable u during minimization of the proposed image segementation model.

4.1 Graphics Hardware

Graphics hardware has to deal with a huge amount of data to render realistic 3D scenes. In the early years, graphics hardware offered a fixed rendering pipeline that could not be altered. In order to compute more complex illumination models programmable shader units were introduced at the vertex and pixel levels. They offered the possibility to apply a small program to each pixel. In turn, the massive parallelisation potential of rendering tasks led graphic processing units (GPUs) towards more and more parallelisation on the hardware side. The latest generation combines the vertex and pixel shader units in the so-called unified shader architecture. State-of-the-art graphics hardware offers the impressive performance of up to 576 GFlops and a memory bandwidth of up to 103.7GB/s.

With the introduction of the GeForce 8-series, NVidia also introduced the CUDA (Compute Unified Device Architecture) framework [11]. CUDA provides a standard C language interface for programming on the GPU [12]. It can handle a massive number of parallel threads that are sheduled to the processor. Moreover, CUDA provides the user with a programming interface that handles sheduling and execution on the GPU. As variational methods are perfectly suited for parallelisation we chose to implement our algorithms using CUDA.

When one takes a close look to the developement of graphics hardware, one can see that the computation performance increases more rapidly than the memory bandwidth. On the GeForce 8800 already 24 floating point operations can be performed while a single operand is fetched from the global GPU memory. Consequently it is sometimes faster to make computations several times, than to store the result in global memory. This also means that algorithms with high arithmetic intensity are best suited for the GPU.

4.2 Implementation on the GPU

The processing units of the GPU are arranged into groups of so-called multiprocessors. One multiprocessor, can execute several independent threads having access to the same shared memory. While reading data from the global GPU memory is still fast (about $75 - 100$ GB/s), reading from the shared memory is even 75 times faster. We exploited this feature by loading a local block (e.g. image patch) into the shared memory and ran the algorithms for several iterations before writing the results back into global memory. High speedups can be gained with this scheme, but the number of such internal iterations should also be limited. The information at block borders can not be exchanged during computation leading to a slower convergence of the entire algorithm. Experiments showed that a number of 5-10 internal iterations provides the best overall performance.

Due to the high costs of accessing global memory and the fact that it is not cached, it is of great importance to obey a continuous access pattern to obtain maximum memory bandwidth. CUDA can load 128-bit words from global memory in a single instruction. Aquisition of memory has to be organized in a way that the simultaneously requested data can be coalesced into as few as possible memory accesses. As we work on higher dimensional blocks, the data has to be appropriately padded to meet the alignment requirements.

In CUDA the warp size indicates the number of threads that are physically executed at the same time on a multiprocessor. Therefore it is important to chose the number of threads per block as a multiple of the warp size. Further-

more if conditionals take different branches in a warp they are serialized. If possible one should account for this to gain optimal speed.

The execution of the algorithm on the GPU has only marginal overhead that has to be executed on the CPU. The CPU can therefore be used for other tasks.

4.3 Graphical User Interface

The graphical user interface is designed to provide the user with simple and fast methods to simplify the segmentation process. We therefore implemented brushes of different functionality:

- Foreground / Background: This hard constraints force the pixels to be of a certain class. At least one foreground and one background seedpoint has to be set.
- Erase Edges: By modifying the edge image with the erase tool, strong edges that are near the desired segmentation border can be deleted.
- Draw Edges (freehand or lines): If certain edges are too weak or missing one can draw additional edges to the image.

For 3D segmentations the user can choose between different views, namely in axial, sagittal or coronal direction. In turn the brushes can be applied to the selected view. Clearly, a user interface working fully in 3D would be of great benefit. The work in [2] suggests some possible directions.

5 Experimental Results

In this Section we demonstrate the effectiveness of our approach by applying our segmentation algorithm to different 2D and 3D medical segmentation tasks. We emphasize that the presented segmentation results should only demonstrate the potential of our method. A rigorous clinical evaluation of our method is an issue of future work. Since our segmentation algorithm solely relies on edge information it is often necessary to apply a smoothing filter to the image before computing the edges.

5.1 2D Results

Fig. 4(a) shows an example where the selection of foreground seeds does not provide enough information to yield the desired segmentation of the liver. By using the erase brush, the edge causing the wrong segmentation is deleted effecting the algorithm to snap to the next stronger edge. Similar, the drawing tool is used to introduce an additional edge that effectively attracts the segmentation border (see Fig. 4(b)). Since our algorithm is guaranteed to yield a global optimum, the algorithm immediately converges against the new optimum after drawing onto the image. Fig. 4(c) depicts the result of the new global optimum after taking into account the local constraints provided by the user.

Fig. 5 shows an segmentation example of a heart MRI data set. Note that any approach that relies on the homogeneity of the intensity distribution would fail in this case. On the other hand only a few inputs from the user are needed to obtain a meaningful segmentation result. The foreground seed regions are shown in light red, and the erased edges are shown in green.

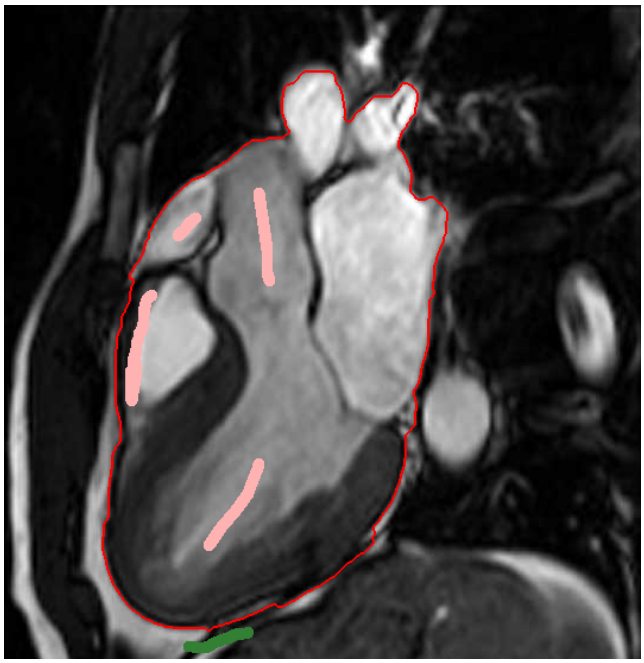


Figure 5: Segmentation of a heart MRI data set.

5.2 3D Results

Due to the large number of voxels in 3D the algorithm usually needs more time to converge to the global minimizer (approximately a few seconds). On the other hand the user also needs more time to draw the constraints to the slices. The interactivity of the segmentation process is therefore not affected at any time. Limited by the GPU memory of 768 MB the largest data set we were currently able to load has a size of $512 \times 512 \times 128$ voxels. Note that this is actually a large data set which can not be processed by most graph cut based approaches.

In the following, we will present three datasets that were segmented in 3D. The first data set is a $512 \times 416 \times 96$ *Liver* MRI data set. The voxel size of this data set is $0.72 \times 0.72 \times 2$mm. Fig. 6 depicts the segmentation result superimposed to a ortho-slice view of the original data set. Note the typical varying contrast frequently found in

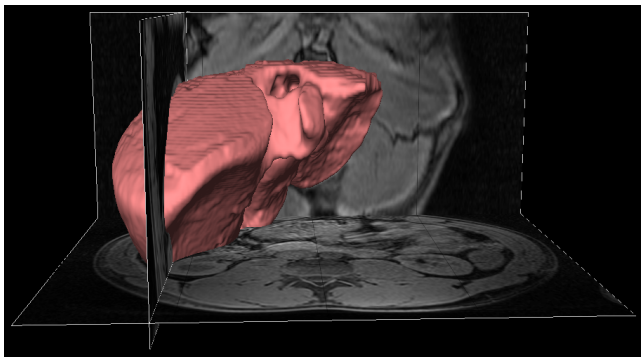
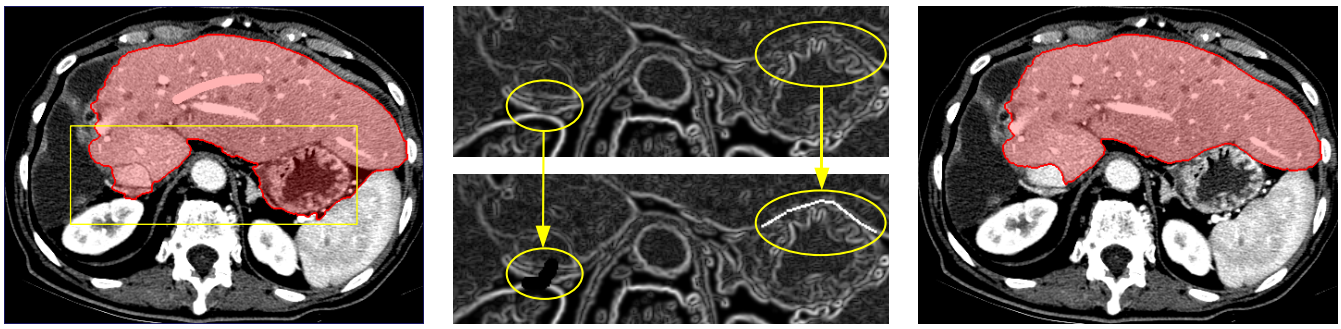


Figure 6: Segmentation of the liver in the *Liver* MRI dataset.



(a) Only seed regions (b) Local constraints (c) Final segmentation

Figure 4: Segmentation process of a liver in a CT image. (a) The segmentation using only a foreground seed point and the border set to background. In the highlighted area segmentation is wrong. (b) Modifying the edge information with the erase and the draw tool. (c) The final segmentation.

MRI images. However, contrast variations do not affect the performance of our algorithm since we solely rely on edge information.

The second data set is a $512 \times 512 \times 96$ CT data set of the *Abdomen*. The voxel size is $0.55 \times 0.55 \times 2$ mm. We segmented the liver, the spleen and the kidneys. Fig. 7 shows a 3D rendering of the segmented viscera. It took us about 8 minutes to obtain a segmentation of the liver. The kidneys and the spleen were obtained after approximately 5 minutes each.

The third 3D dataset presented here is a $256 \times 256 \times 256$ MRI data set of the *Knee*. The voxel size of this data set is $1.36 \times 1.56 \times 1$ mm. Fig. 8 depicts one characteristic slice of this data set. Note that this data set has a low signal to noise ratio and the edges are very weak. Fig. 9 shows a 3D rendering of the segmented femur superimposed to a ortho-slice view of the original data set.

6 Conclusion and Future Work

In this paper, we proposed an interactive general purpose segmentation algorithm. Our approach is based on minimizing an energy functional incorporating a weighted Total Variation regularization and a data term taking into account high-level information from the user. Being convex our algorithm is guaranteed to yield a globally optimal solution. Moreover, since our energy is defined in a continuous setting we do not suffer from metrication errors. We also showed that our algorithm can be seen as a generalization of the continuous maximal flow algorithm of Appleton and Talbot [1].

We have implemented our algorithm on state-of-the-art graphics hardware leading to a interactive segmentation tool usable for 2D and 3D segmentation problems. We proposed to incorporate different types of local constraints provided by the user: Foreground and background seed regions, removal and adding of wrong or missing edge information. If a new constraint is incorporated into the segmentation process, the algorithm immediately adapts the current segmentation to the new situation.

In experimental results we showed that reasonable segmentations of different datasets can be obtained in a few minutes.

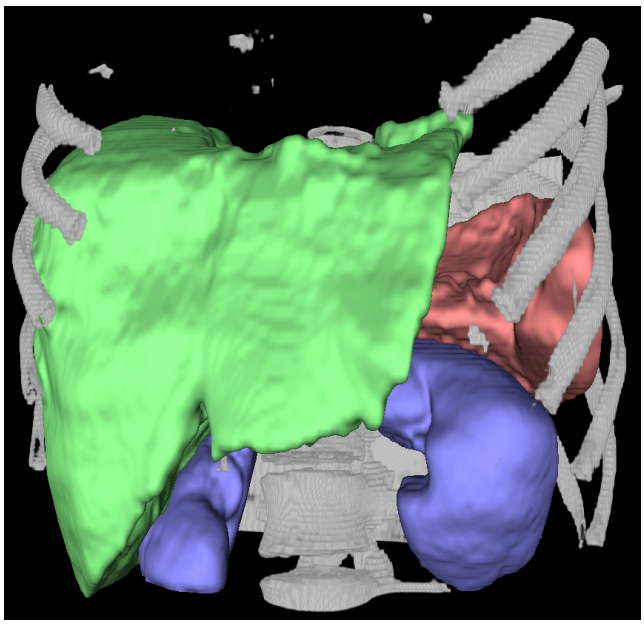


Figure 7: Segmentation of various viscera in the *Abdomen* dataset.

Currently, the memory of the GPU is a limiting factor for the maximum size of the data set that can be processed. However we are currently able to process a maximum size of $512 \times 512 \times 128$ voxels.

In future work we will concentrate on the simultaneous segmentation and rendering of the result by distributing the data over several GPUs. We are also planning to improve the user interface for 3D applications.

Acknowledgement

This work was supported by the Austrian Research Promotion Agency (FFG) within the **VM-GPU** Project No. 813396 .

References

[1] B. Appleton and H. Talbot. Globally minimal surfaces by continuous maximal flows. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(1):106–118, 2006.

[2] A. Bornik, R. Beichel, E. Kruijff, B. Reitinger, and D. Schmalstieg. A hybrid user interface for manipulation of volumetric medical data. In *IEEE Symposium on 3D User Interfaces 2006 (3DUI 2006)*, Alexandria, USA, 2006.

[3] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *9th IEEE International Conference on Computer Vision (CVPR 2003)*, pages 26–33, Washington, DC, USA, 2003. IEEE Computer Society.

[4] X. Bresson, S. Esedoglu, P. Vandergheynst, J. Thiran, and S. Osher. Global minimizers of the active contour/snake model. In *Free Boundary Problems (FBP): Theory and Applications*, 2005.

[5] X. Bresson, S. Esedoglu, P. Vandergheynst, J. Thiran, and S. Osher. Fast global minimization of the active contour/snake model. *J. Math. Imaging and Vision*, 28(2):151–167, 2007.

[6] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *Intl. J. of Computer Vision*, 22(1):61–79, 1997.

[7] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert. Minimal surfaces: A three dimensional segmentation approach. *Nummer. Math.*, 77(4):423–451, 1997.

[8] A. Chambolle. An algorithm for total variation minimizations and applications. *J. Math. Imaging and Vision*, 2004.

[9] A. Chambolle. Total variation minimization and a class of binary MRF models. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 136–152, 2005.

[10] T. Chan and L. Vese. Active contours without edges. *IEEE Trans. Image Processing*, 10(2):266–277, February 2001.

[11] NVIDA Corp. NVidia GeForce 8800 GPU Architecture Overview. Technical report, NVIDA, Santa Clara, CA, USA, 2006.

[12] NVIDA Corp. NVidia CUDA Compute Unified device architecture programming guide 1.1. Technical report, NVIDA, Santa Clara, CA, USA, 2007.

[13] L. Grady. Random walks for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, Nov. 2006.

[14] M. Kass, A.P. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Intl. J. of Computer Vision*, 1(4):321–331, 1988.

[15] S. Leung and S. Osher. Fast global minimization of the active contour model with tv-inpainting and two-phase denoising. In *3rd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, pages 149–160, 2005.

[16] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[17] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992.

[18] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, 2004.

[19] M. Sonka and J. M. Fitzpatrick, editors. *Handbook of Medical Imaging - Volume 2. Medical Image Processing and Analysis*. SPIE Press, Bellingham, Washington, USA, 2000.

Figure 8: A slice of the *Knee* dataset.

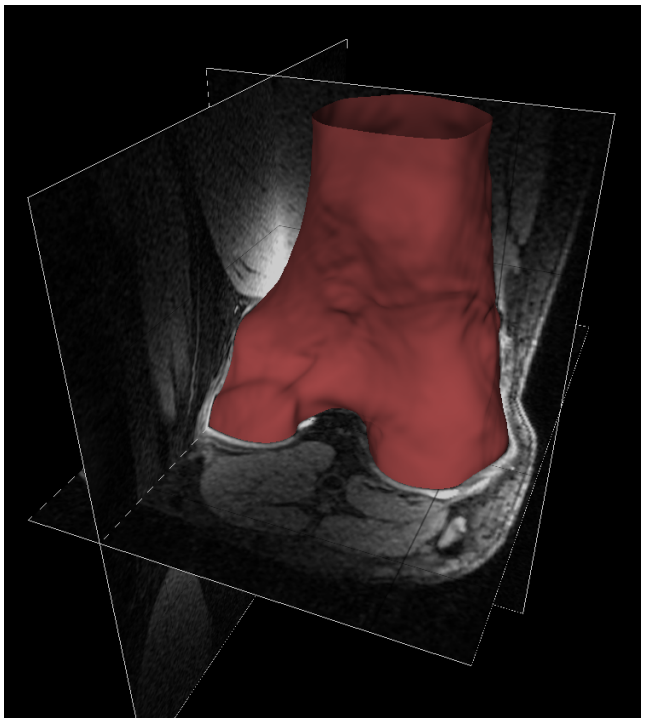


Figure 9: Segmentation of the femur in the *Knee* MRI dataset.