Variational Classification for Visualization of 3D Ultrasound Data

Raanan Fattal* Dani Lischinski†

School of Computer Science and Engineering
The Hebrew University of Jerusalem

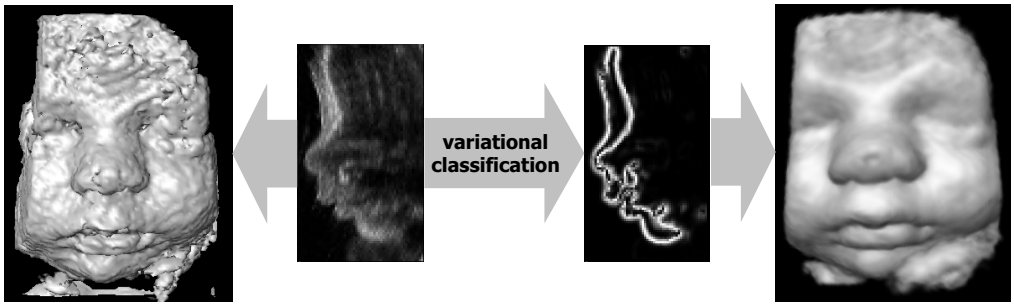

Figure 1: A surface extracted from 3D ultrasound data (left) vs. a surface displayed after variational opacity classification (right).

Abstract

We present a new technique for visualizing surfaces from 3D ultrasound data. 3D ultrasound datasets are typically fuzzy, contain a substantial amount of noise and speckle, and suffer from several other problems that make extraction of continuous and smooth surfaces extremely difficult. We propose a novel opacity classification algorithm for 3D ultrasound datasets, based on the variational principle. More specifically, we compute a volumetric opacity function that optimally satisfies a set of simultaneous requirements. One requirement makes the function attain nonzero values only in the vicinity of a user-specified value, resulting in soft shells of finite, approximately constant thickness around isosurfaces in the volume. Other requirements are designed to make the function smoother and less sensitive to noise and speckle. The computed opacity function lends itself well to explicit geometric surface extraction, as well as to direct volume rendering at interactive rates. We also describe a new splatting algorithm that is particularly well suited for displaying soft opacity shells. Several examples and comparisons are included to illustrate our approach and demonstrate its effectiveness on real 3D ultrasound datasets.

CR Categories: I.3.3 [Computer Graphics]: Picture/image generation—display algorithms, viewing algorithms; I.4.3 [Image Processing and Computer Vision]: Enhancement—filtering, sharpening and deblurring, smoothing

Keywords: 3D ultrasound, classification, isosurface extraction, opacity function, splatting, the variational principle, volume rendering

*raananf@cs.huji.ac.il
†danix@cs.huji.ac.il

1 Introduction

Three-dimensional ultrasound (3DUS) is a relatively new technology for imaging the distribution of ultrasonic echo information throughout a 3D volume of interest inside a patient. Over the past decade 3DUS has become advanced enough to be used in clinical practice. It is used mostly in obstetrics and gynecology [1], as well as for visualization of vessels and tumors in soft tissue, and cardiac imaging. The main advantages of 3DUS over other imaging technologies is that the acquisition procedure is fast, non-invasive, non-radiative, and relatively inexpensive, enabling acquisition to be performed in the clinicians' offices.

Unfortunately, compared to other volumetric medical imaging technologies, such as CT and MRI, high quality visualization of 3DUS datasets is an extremely challenging task, since such datasets typically suffer from considerable noise and speckle, low dynamic range, fuzzy boundaries, and several other problems, which will be described in more detail later. These properties make extraction of smooth, or even continuous, surfaces extremely difficult. For example, the leftmost image in Figure 1 shows a surface that was extracted using Marching Cubes [10] from a 3DUS dataset of a fetus' head. Even though the input dataset was relatively clean in this case, the extracted surface is quite rough, contains holes, and the view of the face is partially occluded by spurious erroneously extracted surface fragments.

Levoy's pioneering work [9] demonstrated that assigning the voxels of a volumetric dataset with non-binary, continuous opacity values is essential for displaying smooth and anti-aliased surfaces from volumetric data. We refer to this process as *opacity classification*.

In this paper, we present a new approach for opacity classification of 3DUS datasets, based on the *Variational Principle*, a mathematical framework for optimization over function spaces. More specifically, we design a functional that effectively imposes several simultaneous requirements on the opacity function. One requirement makes the function attain nonzero values only in the vicinity of a user-specified value, resulting in soft shells of finite, approximately constant thickness around isosurfaces in the volume. Other requirements are designed to make the function smoother and less sensitive

to noise and speckle. An optimal opacity function that minimizes the integral of the functional over the entire volume is computed by solving a large but sparse system of linear equations.

Once the opacity function has been computed, it becomes possible to visualize the corresponding surfaces at interactive rates, using existing techniques, such as Marching Cubes surface extraction [10] or shear-warp volume rendering [7]. We also present *oriented splatting*, a new splatting variant that uses the computed opacity values and opacity gradients to associate an oriented splat polygon with each voxel. The rasterization and texture mapping hardware is then used to compute the corresponding footprints on the image plane. The rightmost image in Figure 1 shows a rendering produced from a classified 3DUS volume using this technique. Such renderings are generated at interactive rates on today's commodity 3D graphics accelerators.

2 Background and Previous Work

2D ultrasound imaging operates by emitting high-frequency sound waves into the body of a patient from a linear array of sources. Sound waves propagating inside the body are partially reflected when they encounter a transition between two different types of tissue (with different sound impedances). Such reflections are recorded by sensors located next to the sound sources, and imaged on the plane using brightness modulation. 3D ultrasound data is acquired from a large number of consecutive 2D slices, obtained by moving a linear array ultrasonic probe or by using a 2D array probe. For further details on the principles of 2D and 3D ultrasound and its uses the reader is referred to [1, 12, 17].

Many different approaches exist for the visualization of 3D structures present in a volumetric dataset. Some techniques, such as the Marching Cubes algorithm [10] extract explicit geometric surfaces (triangle meshes) from volumetric data. Other techniques display surfaces or semi-transparent volumetric 3D structures by tracing rays through the volume [9], or by traversing the volumetric data in object order and splatting each voxel onto the image plane [19, 8]. Most volume rendering techniques require each voxel to have an associated opacity value.

Unfortunately, standard volumetric visualization techniques fail to produce satisfactory results when applied to raw 3DUS data. The main reasons for the failure were reported by Sakas [17, 16], and are quoted here:

1. significant amount of noise and speckle
2. much lower dynamic range compared to CT or MR
3. high variations in the intensity of neighboring voxels, even within homogeneous tissue areas
4. boundaries with varying grey level caused by the variation of surface curvature and orientation to the sound source
5. partially or completely shadowed surfaces by objects closer to the source
6. the regions representing boundaries are not sharp, but show a width of several pixels
7. poor alignment between subsequent images (parallel scan devices only)
8. pixels representing varying geometric resolutions depending on the distance from the sound source (fan-scanning devices only)

These problems make the tasks of volume classification and surface extraction very challenging. For example, larger speckles in the datasets might be detected as opaque blobs occluding the "real" organ surfaces, while finer grain noise makes visualized surfaces appear rougher than they should. High variations in intensity and fuzzy boundaries with varying grey level make it difficult to select the proper parameters for iso-value surface extraction. The goal of our research was to develop a new classification technique that is more robust and designed specifically to address some of the problems typical to 3DUS data.

The best results in visualization of surfaces from 3DUS data to date are presented in the pioneering work of Sakas and Walter [17]. They describe a multistage visualization pipeline. The first stage uses the BLTP (binarize, low-pass, threshold, propagate) filter and a multi-resolution volume pyramid to identify and mask out noisy areas that are believed to contain no useful information, while leaving the data in the remaining areas unaltered. Next, Levoy's opacity classifier [9] is used to indicate surfaces in the unmasked regions of the volume. Finally, the classified unmasked regions are volume-rendered. To achieve smoother rendering, the normals used for shading are obtained from a low-passed version of the volume.

The variational opacity classification algorithm presented in this paper could be viewed as an improved alternative to the first two stages of Sakas' pipeline, achieving noise removal and opacity classification simultaneously using the variational framework. More specifically, the advantages of our approach are:

1. Noise attenuation and smooth surface detection are performed simultaneously, with mutual reinforcement between the two processes. Thus, in addition to attenuation of noise and speckle away from the boundaries, explicit measures are also taken to reduce the noise on or around the boundaries, resulting in smoother surfaces.
2. All of the voxels in the dataset are treated in a uniform fashion. For example, the opacity value of a voxel does not depend on its location with respect to an arbitrary octree subdivision of the space, nor is it subject to binary classification decisions, which may have been erroneously taken at a coarser resolution.
3. The opacity function we compute contains all of the information needed for the subsequent rendering step, including the normals that are needed for shading calculations.

It should be noted that both the BLTP filter and our approach can only remove unwanted noise, but user-assisted segmentation may still be required to remove occluding organs, such as a wall of the uterus in fetal ultrasound.

It should also be pointed out that there have been several recent works dealing with automatic iso-value determination and volume classification (e.g., [2, 6]). However, none of these methods, as far as we know, addressed the specific problems of 3D ultrasound data that were listed earlier.

Splatting

Splatting is a technique originally developed by Westover [19] in order to efficiently render volume data using forward mapping. The main idea is to perform the reconstruction of the projected volume samples in the screen space of the desired view, by spreading the contribution of each volume sample onto every pixel that lies within the screen space footprint of the projected volume sample. Laur and Hanrahan [8] use graphics hardware to render each splat as a collection of polygons with linearly interpolated alpha values. On platforms with hardware support for texture mapping each splat can be rendered as a single polygon, using a texture map to modulate its alpha channel [3].

To our knowledge, all of the splatting-based volume rendering approaches treat the opacity of a voxel as isotropic; in other words, the shape of the opacity reconstruction kernel is the same for all

voxels and all directions. In the case of a parallel projection the image space footprint is also identical for all voxels. In contrast, the splatting approach presented in this paper uses the gradients of the computed opacity function to define, for each voxel, a 2D reconstruction kernel on a plane perpendicular to the opacity gradient. Thus, we treat each voxel as a sample on a fuzzy, partially transparent surface, resulting in a splatting approach that is closer in spirit to those used in recent point-based rendering techniques [13, 15, 21].

The Variational Principle

In this work we make use of the Variational Principle, a mathematical framework for solving a certain class of maximum-minimum problems over function spaces. In the context of computer graphics and geometric modeling the variational framework has mostly been used for surface design and fairing, see e.g. [4]. It has also been used for medical image analysis (see the excellent survey by McInerney and Terzopoulos [11]). In particular, parametric deformable contour models (snakes) [5] have been used to track contours in medical images and volumes. Our approach differs from these previous works in that we are not attempting to fit a parameterizeable surface to the volumetric data, but rather a soft implicit 3D opacity function. Thus, in our approach, we don't have to deal with complicated surface topologies.

3 Variational Opacity Fitting

Interpreting the input volume as a discretized version of some continuous function $v : V \to \mathbb{R}$, where V is the subset of $\mathbb{R}^3$ corresponding to the input volume, we seek another function defined over the same domain $u : V \to \mathbb{R}$, which clearly indicates the features that are of interest to us (organ boundary surfaces), while getting rid of speckle and several of the other problems mentioned earlier. The search for u is formulated as an optimization problem over the space of all twice differentiable functions from V to $\mathbb{R}$. More specifically, our goal is to find a function u that minimizes the integral

$$\int_V F(u, \nabla u, \mathbf{x})\, d\mathbf{x}, \tag{1}$$

subject to the condition that $u \equiv 0$ on the boundary of V, where F is a suitably defined functional, and $\nabla u = (u_x, u_y, u_z)^{\mathrm{T}}$ is the gradient of u.

According to the Variational Principle, a function u that minimizes equation (1) must satisfy the Euler-Lagrange equation

$$\frac{\partial F}{\partial u} - \frac{d}{dx}\frac{\partial F}{\partial u_x} - \frac{d}{dy}\frac{\partial F}{\partial u_y} - \frac{d}{dz}\frac{\partial F}{\partial u_z} = 0, \tag{2}$$

which is a partial differential equation (PDE) in u (with the same boundary condition $u \equiv 0$).

3.1 The functional F

Our next step is to design an appropriate functional F. The functional should satisfy the following two general requirements:

1. The integral of F in equation (1) must be bounded from below, otherwise a solution might not exist;
2. The resulting PDE (eq. (2)) should be easy to solve. In practice, we would like to limit ourselves to solving linear equations, so the PDE should be linear in u and in its derivatives.

Both of these requirements are easily satisfied by designing F as a sum of positive terms, where in each term either u or one of its derivatives appears squared.

In this work, we define F as a weighted sum of three terms:

$$F = \alpha F_{\mathrm{iso}} + \beta F_{\mathrm{tan}} + \gamma F_{\mathrm{ind}}. \tag{3}$$

The isovalue term F_{iso} allows u to have a high value in areas of V where the original input function v has values near a user-specified isovalue, and pulls it to zero elsewhere. The tangential term F_{tan} forces the gradients of u to point in the same directions as the gradients of a smoothed version of the volume v. The F_{ind} term requires u to have a user-specified non-zero value u_{ind} whenever it can. In particular, it pulls u towards u_{ind} in areas of the volume where v has a high gradient. Each of these terms is defined and described in more detail below.

Note that in contrast to methods that preprocess a volume by sequential application of various filters (e.g., smoothing filter, followed by median filter, followed by thresholding), the variational fitting of u attempts to find a function that satisfies several requirements simultaneously in an optimal manner. The requirements are represented by the different terms in F and their relative importance depends on the weights α, β, and γ.

3.2 The isovalue term F_{iso}

The isovalue term is designed to pull u to zero everywhere, except at locations where v is close to a user-specified value v_{iso}. A simple definition for such a term could be

$$F_{\mathrm{iso}} = u^2 (v - v_{\mathrm{iso}})^2 \left(\frac{1}{|\nabla v|^2 + \varepsilon} \right).$$

Note that u, v, and ∇v are all functions of the location $\mathbf{x}$, which is omitted for clarity.

Since u^2 is multiplied by a term that rapidly increases as v gets farther away from v_{iso}, a solution that minimizes the integral in equation (1) must be as close to zero as possible in such regions. On the other hand, in locations where v equals v_{iso}, the solution u can take any value without increasing the integral. As a result, u is mostly zero, but it has a non-zero "shells" of finite thickness around the isovalue surfaces. The multiplier $\left(|\nabla v|^2 + \varepsilon\right)^{-1}$ is there to ensure a uniform width of these non-zero shells. The ε is a small number that prevents division by zero and also pulls u to zero in constant areas with value v_{iso}. This term could be viewed as our framework's analog to Levoy's opacity function [9], which linearly depends on $|v - v_{\mathrm{iso}}|/|\nabla v|$.

However, we use a somewhat more sophisticated term. Since 3DUS data suffers from noise and speckles, blindly following isovalue contours might result in very rough surfaces and in spurious blobs disconnected from the "true" surface. We found that these effects can be significantly reduced by replacing $(v - v_{\mathrm{iso}})^2$ with the term $\omega(v - v_{\mathrm{iso}})^2 + (1 - \omega)(\tilde{v} - v_{\mathrm{iso}})^2$, where $\tilde{v}$ is a smoothed version of v (e.g., the result of convolving v with a Gaussian filter). In other words, we blend between the isovalue contour in the original volume and the contour in the lowpassed volume. Thus, the definition of F_{iso} becomes

$$F_{\mathrm{iso}} = u^2 \left(\omega\, |v - v_{\mathrm{iso}}|^2 + (1 - \omega)\, |\tilde{v} - v_{\mathrm{iso}}|^2 \right) \left(\frac{1}{|\nabla v|^2 + \varepsilon} \right).$$

3.3 The tangential term F_{tan}

So far, we have not introduced any explicit constraints to make u smooth. Such constraints are needed because of the high level of

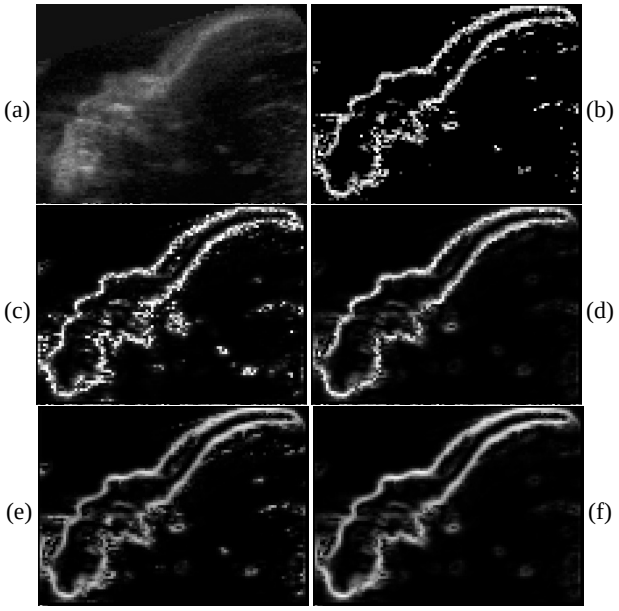

Figure 2: An example. (a) input slice; (b) Levoy's opacity classifier; (c) $\omega = 1, \beta = 0$; (d) $\omega = 0.6, \beta = 0$; (e) $\omega = 1, \beta = 0.03$; (f) $\omega = 0.6, \beta = 0.03$.

noise in 3DUS data. However, we must be careful not to oversmooth, eliminating important fine features. We chose to use a term that forces the gradient of u to point, as much as possible, in the direction of the gradient of $\tilde{v}$, a lowpassed version of v, by minimizing the cross product between the two gradients:

$$F_{\text{tan}} = |\nabla u \times \nabla \tilde{v}|^2$$

In other words, we would like to make iso-surfaces in u to be locally tangent to iso-surfaces in $\tilde{v}$.

3.4 The indication term F_{ind}

In order to prevent the existence of a trivial minimum $u \equiv 0$ we introduce another term into the functional. This term penalizes u for deviating from a user-specified non-zero value u_{ind}. This effect could be very simply achieved by setting

$$F_{\text{ind}} = (u - u_{\text{ind}})^2$$

The two previous terms prevent the solution $u \equiv u_{\text{ind}}$, and in practice the solution u achieves the value u_{ind} only in regions where the constraints imposed by F_{iso} and F_{tan} are satisfied.

Better results are achieved when we modify this term such that it also pulls u towards the value u_{ind} in regions where the volume v has a high gradient. In this manner, we effectively combine in our functional two different ways of detecting a surface: tracing an isovalue and detecting high gradients. Thus, the term actually used in our implementation is defined as

$$F_{\text{ind}} = \left(\delta + (1 - \delta)\,|\nabla v|\right)(u - u_{\text{ind}})^2$$

The parameter δ allows us to control the effect of $|\nabla v|$ on u. δ must be greater than zero to prevent the vanishing of the F_{ind} terms whenever $\nabla v = 0$, which in turn causes the resulting linear system to become singular.

3.5 Example

We illustrate the effect of the different terms in our functional using Figure 2. Image (a) in the figure is a slice from a 3DUS dataset of a fetus' head. Image (b) shows the results obtained by running Levoy's opacity classifier [9] on this dataset. This is also the classifier that Sakas and Walter [17] use to detect surfaces in their system. It can be seen that the surfaces have indeed been detected, but the contour is noisy, and there are also several blobs caused by speckles in the data.

Image (c) shows the result we obtained without the tangential term ($\beta = 0$) and without making use of a lowpassed version of v ($\omega = 1$). As mentioned earlier, in this case the F_{iso} term resembles Levoy's classifier, and the resulting image (c) appears qualitatively similar to (b). Setting $\omega = 0.6$ causes the lowpassed volume $\tilde{v}$ to be taken into account as well, causing a significant reduction in the noise in image (d). Note that the main contour remains largely unchanged with respect to (c). Image (e) shows the effect of the tangential term F_{tan} (using $\beta = 0.03$). Here $\omega = 1$ again, so the noise has not been removed, but the contour is noticeably smoother than in (c). Finally, image (f) was generated using $\omega = 0.6$ and $\beta = 0.03$, resulting in an image that is both cleaner and has a smoother contour than (c).

3.6 System solver

Substituting the functional F described above into the Euler-Lagrange equation (2), for each point $\mathbf{x} \in V$ we obtain a linear equation in u, u_x, u_y, u_z,

$$\begin{aligned}\alpha\, u \left(\omega\,|v - v_{\text{iso}}|^2 + (1-\omega)\,|\tilde{v} - v_{\text{iso}}|^2\right)\left(\frac{1}{|\nabla v|^2 + \varepsilon}\right) \\ + \beta\ (\boldsymbol{\nabla} \times \nabla \tilde{v}) \cdot (\nabla u \times \nabla \tilde{v}) \\ + \gamma\ \left(\delta + (1-\delta)|\nabla v|\right)\ (u - u_{\text{ind}}) \quad = \quad 0,\end{aligned}$$

where $\boldsymbol{\nabla}$ is the vector of differentiation operators

$$\boldsymbol{\nabla} = \left(\frac{d}{dx}, \frac{d}{dy}, \frac{d}{dz}\right)^{\mathrm{T}}.$$

Additionally, we have the boundary conditions that $u \equiv 0$ on the boundary of the volume. Thus, we are faced with a boundary value problem. We solve the problem on the same discrete sampling grid of the input volume using finite differences and relaxation on a multigrid [14].

More specifically, let $\mathbf{x} = (i, j, k)$ on the sampling grid. In the equation corresponding to $\mathbf{x}$ we approximate the partial derivative u_x as

$$u_x = \frac{u(i+1, j, k) - u(i-1, j, k)}{2h_x},$$

where h_x is the grid spacing along the x axis. A similar formula is used for u_y and u_z. The finite differences approximation produces a diagonally dominant linear system of equations where the unknowns are the values of u on the grid points (i, j, k). This system is sparse, as each equation couples each grid point only with its twenty-six neighbors.

To solve the resulting diagonally dominant linear system we use Gauss-Seidel relaxation, accelerated using a two-level grid. The equations are first solved on a half-resolution grid. The result of the coarse solution is interpolated to the finer grid and used as an initial guess for another round of relaxation.

In our experiments, we used 2 to 4 Gauss-Seidel iterations on the coarse grid, followed by 5 to 10 iterations on the fine grid. The

number of iterations depends mostly on the weight β of the tangential term. Solution times for datasets with size ranging from 400,000 to 1,150,000 voxels were between 4 and 11 seconds (measured on a 600 MHz Pentium III PC running Linux) including the pre-computation of ∇v, $\tilde{v}$, and $\nabla \tilde{v}$, setting up the equations, and solving them. Detailed statistics are provided in Table 1.

4 Surface Visualization

The result of the variational opacity classification stage described in the previous section could be thought of as a fuzzy surface of roughly fixed thickness, indicated by continuously varying non-zero opacities. Our goal is to visualize this surface. Since the values in the original ultrasound data only indicate transitions between regions with different sound impedances, once the surfaces have been identified, the original data is of no further use to us in the visualization process. Therefore, in the rendering stage we only make use of the computed opacity function. As we shall demonstrate in Section 5 the opacity function is quite sparse; typically over 80 percent of the voxels are either zero, or have a negligibly small value. This sparsity supports rapid visualization of the indicated surfaces.

The fuzzy surfaces are typically quite thin. Thus, it is possible to apply Marching Cubes, or a similar geometric surface extraction algorithm to the opacity function. This results in sharp opaque surfaces, but fails to properly convey their fuzzy nature. Alternatively, a direct volume rendering algorithm based on ray casting, shear-warp factorization, or splatting can be used. In principle, such algorithms are able to better account for partial opacities.

In this section, we present *oriented splatting*, a new modified splatting algorithm that is particularly well-suited for visualizing opacity functions computed with our variational classification procedure.

4.1 Oriented splatting

Our splatting approach operates as follows. Each voxel $\mathbf{x}$ whose computed opacity $u(\mathbf{x})$ exceeds some user-specified threshold ϵ is assigned a single *splat triangle*, centered at $\mathbf{x}$ and oriented perpendicular to ∇u. The splat triangle alpha value is set to $u(\mathbf{x})$, which is modulated by means of a texture map that creates a Gaussian decay in the alpha channel away from the triangle's center. Once created, the resulting collection of texture-mapped triangles is rendered from any desired viewpoint in back-to-front order. The rendering is done at interactive rates with today's commodity 3D graphics accelerator cards, which provide support for texture mapping and alpha blending in hardware.

Note that in contrast to classical splatting algorithms that treat the opacity of each voxel as isotropic and use the same reconstruction kernel for all voxels and in all directions, we use a 2D reconstruction kernel defined on the plane of each splat triangle, which is oriented perpendicularly to the opacity gradient. The correct footprint on the image plane is computed by the rasterizing hardware, provided that it supports perspective-corrected texture mapping. Since reconstruction kernels are oriented perpendicularly to opacity gradients, they result in sharper reconstruction than the usual isotropic splats, and avoid incorrect thickening around surface silhouettes, as demonstrated in Figure 3.

Now it remains to explain how to define an appropriate 2D Gaussian texture function, establish the appropriate texture coordinates for the vertices of the splat triangle, and set an appropriate size for the splat triangles. Our goal is to find values for these parameters such that the overlap between the footprints of adjacent and identically oriented splat triangles creates the appearance of a constant transparency surface on the image plane.

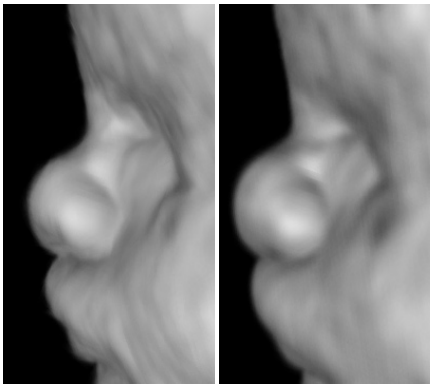

Figure 3: Oriented splats (left) vs. isotropic splats (right).

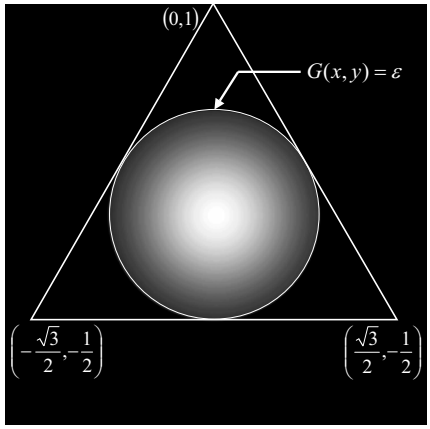

Figure 4: Splat triangle

Our construction is illustrated in Figure 4. We define a windowed Gaussian opacity modulation texture function over the 2D texture space $[-1,1]^2$. Specifically, the texture function is defined as the Gaussian

$$G(x,y) = e^{-(x^2+y^2)/\lambda}$$

inside the circle $x^2+y^2 = 0.5^2$, and zero outside that circle. Next, we construct an equilateral triangle, in texture space, that circumscribes this circle. The texture coordinates at the vertices of such a triangle are indicated in Figure 4.

The Gaussian parameter λ is chosen such that $G(x,y) = \varepsilon$ on the boundary of the inscribed circle, where ε is the desired cutoff value (in our implementation $\varepsilon = 0.2$ was empirically determined to work well). It is easily verified that this condition is satisfied by setting $\lambda = -0.25 \ln \varepsilon$.

The world coordinates of the splat triangle's vertices are determined as follows. We start by constructing two perpendicular unit vectors $\mathbf{s}$ and $\mathbf{t}$, which lie in the plane perpendicular to ∇u, the gradient of our opacity function. The vertices of the splat triangle are then computed as

$$\mathbf{x}_1 = \mathbf{x} + a\,\mathbf{s} \qquad \mathbf{x}_2 = \mathbf{x} - \frac{a}{2}\left(\mathbf{s} + \sqrt{3}\mathbf{t}\right) \qquad \mathbf{x}_3 = \mathbf{x} - \frac{a}{2}\left(\mathbf{s} - \sqrt{3}\mathbf{t}\right)$$

The size of the splat triangle is controlled by the parameter a. We use $a = \sqrt{1/\lambda}$, resulting in roughly constant reconstructed opacity when two adjacent splat triangles parallel to the image plane are projected and alpha-composited onto it. More precisely, the recon-

structed opacity midway between the centers of the splats is roughly 0.95.

Splat triangles are shaded using OpenGL's standard Phong-like shading model [20]. We have experimented with two different methods for computing the normals necessary for OpenGL's lighting computations.

The first method uses the normalized opacity gradient $\nabla u/|\nabla u|$, computed at the center of the voxel $\mathbf{x}$ (which, by our construction, is also the center of the splat triangle). This single normal is specified for each splat triangle vertex, resulting in essentially flat shading for the triangle, which is smoothed to some degree by the Gaussian opacity function. This works well in relatively flat areas, but can still create somewhat facetted appearance in regions of high curvature.

Smoother results are obtained by assigning a different normal to each splat triangle vertex. Rather than computing the normal at the center of the triangle, we compute three normals, at locations midway between the center and each of the three vertices. As a result, three different color values are computed at the vertices of the triangle, which are then linearly interpolated across the triangle.

Since the splat triangles are rendered with alpha blending turned on, the rendering must be performed in a back-to-front order. In our current implementation we create six OpenGL display lists, each containing the splat triangles sorted along the positive or the negative direction of one of the three coordinate axes. During interactive display the renderer selects and calls the display list that corresponds to the direction closest to the look-at vector. It should be noted that this strategy is not guaranteed to produce the correct back-to-front ordering because the splat triangles are larger than the voxels, and they overlap and intersect each other. However, we have not observed any artifacts caused by these occasional sorting errors.

5 Results

The techniques described in this paper were implemented in C++ and tested on a 600 MHz Pentium III PC with 256 MB of RAM, under the Linux operating system. Hardware-accelerated OpenGL rendering was performed on an NVIDIA GeForce2 GTS graphics board.

We have experimented with several different 3D ultrasound datasets. The variational opacity classification times for these datasets were between 4.0 and 10.5 seconds. The time includes low-pass filtering of the input volume, the computation of gradients, setting up the system of equation, and solving it. The classification parameters α, β, γ and v_{iso} were user-specified. Additional statistics pertaining to all four datasets are summarized in Table 1.

Figure 5 demonstrates the effect of variational classification on two different datasets: each dataset was rendered using ray casting before and after variational classification. It can be seen that variational classification is quite effective at removing noise and revealing the surfaces of interest.

Figure 6 shows the the same two datasets, as well as two other examples. In this figure all of the images were generated *after* performing variational classification, and it compares between different visualization techniques of the *same* classified volumes. The leftmost column shows the dataset rendered using ray casting, the next column shows a rendering of a polygon mesh extracted from the volumetric opacity function using Marching Cubes. The third column shows a software rendering of our opacity function using Lacroute and Levoy's shear-warp factorization algorithm [7] (we used Stanford's VolPack volume rendering library [18]). The right

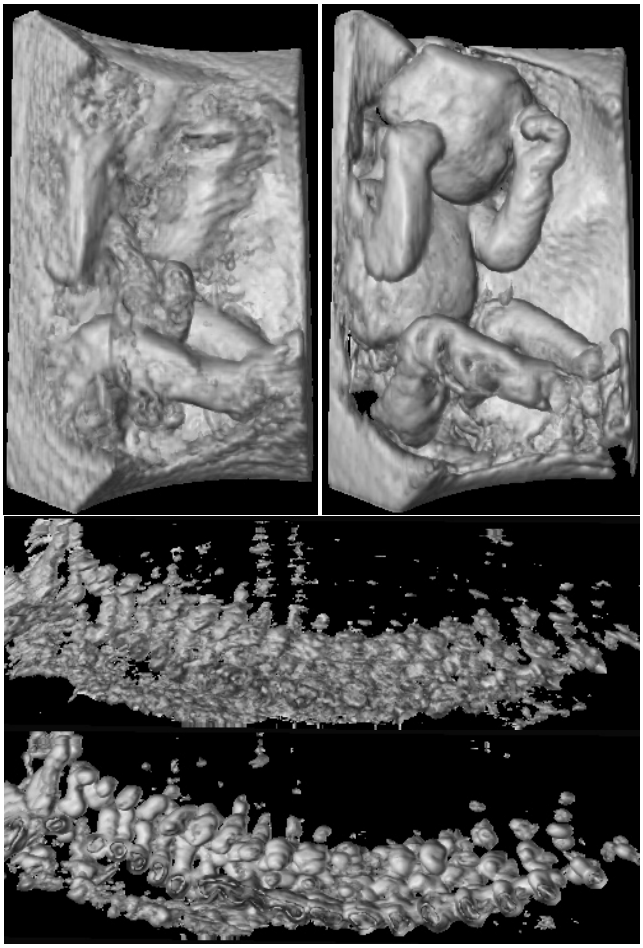

Figure 5: The top two images show a visualization of a fetal 3D ultrasound visualization using ray casting before (left) and after (right) variational classification. The two images below show a similar comparison for a dataset of a spine (top image before, bottom image after variational classification.

column rendered the opacity function using the oriented splatting technique described in Section 4.1. With each rendering technique except ray casting the rendering time is fast enough to allow interactive, or near-interactive changes in viewpoint[1].

As expected, explicitly extracted surfaces give the sharpest results, however they also exhibit the most roughness, as a result of having to make a binary decision for each voxel whether or not a surface passes through it. The oriented splatting technique appears to give the smoothest reconstruction without loosing any essential details. This is not surprising, since it was explicitly designed to visualize the soft, fuzzy surfaces represented by our opacity function. Shear-warp renderings are slightly inferior in quality to those produced using the other two techniques, but overall it seems a reasonable alternative if hardware-accelerated rendering is not available. While we believe that the smooth renderings produced by oriented splatting are the most aesthetically pleasing, in a clinical application it might be beneficial to present doctors with several different visualizations, alongside with the ability to examine the original unprocessed data.

We were unable to perform an exact comparison with previous

[1]Movies in MPEG and QuickTime formats can be found under `http://www.cs.huji.ac.il/~danix/3dus`.

Dataset	dimensions	classification time (seconds)	% opaque	# splats	rendering time shear-warp / splatting
Fetal face I	77×97×56	5.7	16	32,188	0.13 / 0.1
Fetal face II	77×60×83	4.0	17.5	30,428	0.12 / 0.11
Whole fetus	123×71×85	10.5	12.5	44,908	0.2 / 0.12
Spine	314×85×43	5.7	16	96,349	0.21 / 0.26

Table 1: Dataset statistics. All times are in seconds on a 600 MHz Pentium III PC equipped with a GeForce2 GTS graphics board. Rendering times refer to 400 × 400 images. At this resolution our unoptimized ray caster takes between 10 to 20 seconds per image.

Figure 6: Rendering examples. The top row shows a fetal face rendered using ray-casting (left), Marching Cubes, shear-warp, and oriented splats (right). The second row shows another fetal face rendered using the same four methods. The third row shows a view of an entire fetus. The bottom row shows a spine rendered using Marching Cubes (left) and oriented splats (right).

results reported in the literature [17], as the datasets were not available to us. Our images appear to be somewhat cleaner and smoother, and the processing and rendering times are shorter (although this can be partially attributed to faster hardware).

In summary, variational classification and oriented splatting appear to work rather well, particularly on datasets where the object of interest is large and bounded by a continuous surface, which is frequently the case in fetal ultrasound. The spine dataset shown in the two bottom rows of Figure 6 is an example where these conditions are not satisfied. The object of interest here consists of many small components, so it is more difficult to distinguish between small parts of the spine and nearby speckles if they are close to the selected isovalue.

6 Conclusions

We have presented a new approach for opacity classification of 3DUS datasets using the variational principle. Our approach attempts to fit a continuous opacity function to the volume by optimally balancing between several simultaneous requirements. The approach is robust in the presence of noise and speckle, and results in fuzzy, soft surfaces indicated by continuously varying non-zero opacities. We have also described oriented splatting, a new variant of splatting, which is particularly well suited for interactive visualization of such surfaces using fast texture mapping and compositing hardware. Using our approach we have been able to produce some of the cleanest and smoothest 3DUS visualizations to date.

There are several directions for future work:

- Automatic determination of the various weights used in our functional for obtaining optimal results with minimal user intervention.
- Enabling users to interactively experiment with different classification parameters by performing fast incremental updates to the solution.
- Experimenting with additional or alternative terms for further improvements in the quality of the extracted surfaces. For example, we would like to allow the target isovalue to vary across the volume, for improved adaptation to the intensities present in the data.
- Exploring usage of non-linear terms in the functional instead of or in addition to the current linear terms for added flexibility and control.

Acknowledgements

This work was supported in part by the Izmel Consortium on Image-Guided Therapy, and by the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities. The authors would also like to thank Biomedicom for their assistance.

References

[1] K. Baba and D. Jurkovic, editors. *Three-Dimensional Ultrasound in Obstetrics and Gynecology*. Progress in Obstetric and Gynecological Sonography Series. The Parthenon Publishing Group, 1997.

[2] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In *Proceedings IEEE Visualization '97*, pages 167–173, Phoenix, AZ, Oct. 1997.

[3] R. A. Crawfis and N. Max. Textured splats for 3D scalar and vector field visualization. In Nielson and Bergeron, editors, *Proceedings of Visualization '93*, pages 261–266, Oct. 1993.

[4] G. Greiner. Variational design and fairing of spline surfaces. *Computer Graphics Forum (Proc. Eurographics '94)*, 13(3):143–154, 1994.

[5] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[6] G. Kindlmann and J. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Proceedings IEEE Symposium on Volume Visualization*, pages 79–86, Research Triangle, NC, Oct. 1998.

[7] P. Lacroute and M. Levoy. Fast volume rendering using a shear–warp factorization of the viewing transformation. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '94)*, pages 451–458, July 1994.

[8] D. Laur and P. Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. In T. W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 285–288, July 1991.

[9] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[10] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In M. C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, July 1987.

[11] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: A survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[12] T. R. Nelson and T. T. Elvins. Visualization of 3D ultrasound data. *IEEE Computer Graphics and Applications*, 13(6):50–57, Nov. 1993.

[13] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of SIGGRAPH 2000*, pages 335–342, 2000.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

[15] S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of SIGGRAPH 2000*, pages 343–352, 2000.

[16] G. Sakas, L.-A. Schreyer, and M. Grimm. Preprocessing and volume rendering of 3D ultrasonic data. *IEEE Computer Graphics and Applications*, 15(4):47–54, July 1995.

[17] G. Sakas and S. Walter. Extracting surfaces from fuzzy 3D-Ultrasound data. In R. Cook, editor, *SIGGRAPH '95 Proceedings*, Annual Conference Series, pages 465–474. ACM SIGGRAPH, Aug. 1995.

[18] The VolPack volume rendering library. http://www.graphics.stanford.edu/software/volpack.

[19] L. Westover. Footprint evaluation for volume rendering. In F. Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 367–376, Aug. 1990.

[20] M. Woo, J. Neider, and T. Davis. *OpengGL Programming Guide*. Addison Wesley Developers Press, second edition, 1997.

[21] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Proceedings of SIGGRAPH 2001*, 2001.