Iso-Contour Volume Rendering

James Arvo
Program of Computer Graphics
Cornell University [1]

Kevin Novins
Laboratoire iMAGIS
Institut IMAG [2]

Abstract

In this paper we present a new approach to volume rendering in which curves of constant intensity in image space, or *iso-contours*, are computed directly for each view. The generated iso-contours can be used to drive various visualization and feature-detection algorithms. The approach imposes no restriction on the organization of the data points and can accommodate a large class of radially-symmetric filter functions. The technique works well for both perspective and orthographic viewing projections. Each iso-contour is defined by an ordinary differential equation, which is solved numerically using a predictor-corrector method. A key element of the algorithm is the use of image intensity gradients, which we compute by means of a closed-form expression that holds at every point on the image plane. A caching algorithm is described that dramatically accelerates the gradient computations on large datasets. The algorithm is demonstrated on emission-only datasets. We conclude by describing a number of possible enhancements.

1 Introduction

Current paradigms for direct volume rendering are oriented toward fixed-resolution images. In this paper, we introduce a new approach to volume rendering based on the direct computation of image-space curves of constant intensity, or *iso-contours*. Our approach is motivated by the observation that it is not always necessary to process an entire dataset or sample the entire image plane at a high level of detail to identify salient features of the resulting image. The iso-contour method of volume rendering can be used to efficiently extract one class of image-space features: curves of constant intensity.

The principal application of directly-computed iso-contours is the rapid computation of high-level image features. Other potential applications include the steering of computation to regions of interest, generation of families of contours for resolution-independent rendering, and progressive refinement of volume-rendered images.

We describe an algorithm for generating iso-contours of controllable accuracy. Given any point on the image plane, the algorithm uses a predictor-corrector method [1] to follow the contour along which the image intensity remains constant. A key element of the curve-following algorithm is analytic computation of image-space intensity gradients. In previous work, a similar approach has been applied to the direct computation of curves of constant irradiance for lighting calculations in polyhedral environments [2].

Intensity gradients may also be used to drive related image analysis tasks. We have implemented efficient algorithms for locating local extrema, and for semi-automatic edge detection based on the magnitude of the gradient. Because the algorithm treats the image as a continuous function rather than as a collection of discrete samples, we can perform these tasks with much greater reliability and accuracy than algorithms that operate by post-processing images.

Our current implementation is limited in the type of data projection and in the shape of the filter kernels. The data projection is restricted to the emission-only model, while the reconstruction requires spherically symmetric filter kernels. There is great latitude in selecting the filters, however, both in their radial spread function and in their spherical extent. For instance, different filter kernels with different radii may be applied at each data sample. The aspect in which our approach enjoys the greatest generality is in the location of the data samples; the iso-contour method imposes absolutely no restrictions on the spatial organization of the data.

1.1 Iso-Surfaces vs. Iso-Contours

The iso-contour approach differs fundamentally from iso-surface extraction. Iso-surface techniques convert volume data into surface data, based on analyses of the volume data in local neighborhoods. These techniques often operate in a totally view-independent manner [11]. During rendering, it is the extracted surface rather than the original volume data that is illuminated and projected onto the image plane. In contrast, the iso-contour method operates directly in continuous image space, driven by features of the final 2D *image* rather than those of a 3D surface. The contours thus depend on the viewing projection of the volume data as well as the line integrals used to map the 3D data to image intensities.

A similar object space versus screen space distinction can be made between our approach and that of Helman and Hesselink [7]. Their approach generates contour lines and other indicators of vector field topology in object space, which are later projected. In contrast, our contours exist in screen space and are useful in understanding the intensity variation of the data *after* a volume projection model has been applied.

[1] Address:580 Engineering and Theory Center, Ithaca, New York 14853 USA
[2] iMAGIS is a joint project between CNRS, INRIA, INPG, and UJF. Address: BP53, 38041 Grenoble Cedex 09 France

2 Intensity Gradients

A central component of the algorithms we describe is the computation of image intensity gradients. In this section we derive closed-form expressions for the intensity and intensity gradient at points in the image plane. Visualization techniques that make use of this information are presented in Section 3. We shall assume an emission-only projection model, such as employed in the Fourier volume rendering algorithms [12, 15]. This model allows the effect of each data point to be considered in isolation; filter kernel overlap and ordering are unimportant. This property allows us to compute both the intensity and the intensity gradient quite simply.

We note that emission line integrals can be easily re-interpreted as optical depth by applying an exponential function. Thus, all of our results also apply to absorption-only models, which can be used to compute simulated X-ray images that are useful in medical applications [13]. Such an exponential transformation affects the isovalues but not the shapes of the contours.

To state the problem more formally, we introduce some nomenclature. We assume that we are given n data points in $\mathbb{R}^3$, each with an associated weight indicating emission, and an associated spherically-symmetric point spread function. Thus, a volume dataset consists of a collection of position-weight-filter 3-tuples, denoted by (p_i, w_i, f_i), for $i = 1, 2, \ldots, n$. Here, $f_i(\sigma)$ is a function that returns the value of the filter for a given radial distance, σ. The emission function in space due to the i^{th} tuple, denoted $\epsilon_i(x)$, is determined by centering the filter kernel at p_i and scaling it by w_i. Thus,

$$\epsilon_i(x) = w_i \, f_i \left(|| \, p_i - x \, || \right), \tag{1}$$

where $|| \cdot ||$ is the Euclidean norm. Total emission for a point in space is determined by summing these individual emission functions.

Now, consider a single ray (r, v), where r is the ray origin and v is the unit direction vector. Assuming the emission-only model, the intensity ϕ associated with this ray is given by

$$\begin{aligned} \phi(r, v) &= \int_{-\infty}^{\infty} \sum_{i=1}^{n} \epsilon_i(r + tv) \, dt \\ &= \sum_{i=1}^{n} \int_{-\infty}^{\infty} \epsilon_i(r + tv) \, dt. \end{aligned} \tag{2}$$

By substituting the definition of $\epsilon_i(x)$ from equation (1), we obtain

$$\phi(r, v) = \sum_{i=1}^{n} w_i \int_{-\infty}^{\infty} f_i(|| \, r + tv - p_i \, ||) \, dt. \tag{3}$$

Note that the contribution of each data point is independent of its position *along* the ray. Because of this, and the spherical-symmetry of the filter kernels, the value of the integral in equation (3) depends only on the perpendicular distance from the ray to each sample point.

Equation (2) can be expressed in a more useful form by introducing two new functions. Let $s_i(r, v)$ be the distance from the point p_i to the ray (r, v), and let $\mathcal{F}_i(s)$ be the integral of the i^{th} filter along a ray passing within s of the filter's center. Note that $\mathcal{F}_i$ is independent of the ray orientation due to the spherical symmetry of the filters. In terms of $s_i(r, v)$ and $\mathcal{F}_i(s)$ we have

$$\phi(r, v) = \sum_{i=1}^{n} w_i \, \mathcal{F}_i(s_i(r, v)). \tag{4}$$

To obtain the object space intensity gradient, we differentiate this expression using the chain rule:

$$\nabla \phi(r, v) = \sum_{i=1}^{n} w_i \, \mathcal{F}_i'(s_i(r, v)) \, \nabla s_i(r, v), \tag{5}$$

where the gradient operator ∇ is taken with respect to either the position r or the direction v, the other being held fixed. See Figure 1. The functions $s_i(r, v)$, $\mathcal{F}_i(s)$ and their derivatives are given in the following two subsections. In Section 2.3, we present the method for obtaining an image space gradient from the object space gradient given in equation (5).

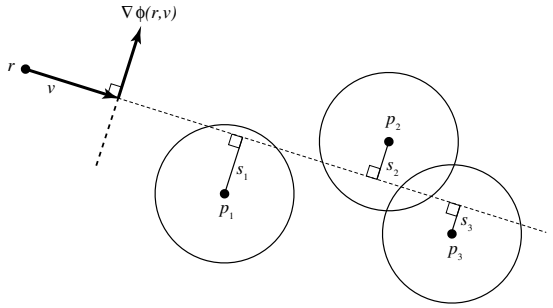


Figure 1: *The intensity gradient is a function of sample distances from the ray. The gradient is always in the plane perpendicular to v, for both perspective and orthographic projections.*

2.1 The Distance Function and its Gradient

The *distance function* $s_i(r, v)$ introduced in equation (4) can be defined as

$$s_i(r, v) \equiv \left|\left| \left(\mathbf{I} - vv^{\mathrm{T}} \right) \xi_i \right|\right|, \tag{6}$$

where

$$\xi_i \equiv p_i - r, \tag{7}$$

and $\mathbf{I}$ denotes the 3×3 identity matrix. The matrix $\mathbf{I} - vv^{\mathrm{T}}$ is a projection onto the plane orthogonal to the unit vector v. The function $s_i(r, v)$ can be viewed as a 3D scalar field over all ray origins by fixing the direction, or over all directions by fixing the origin; the interpretation we use depends on the type of viewing projection. For perspective projection, the eye point r is fixed. Taking the gradient of equation (6) with respect to v, we have

$$\nabla s_i^{\text{persp}}(v) = -\frac{\xi_i{}^{\mathrm{T}} \left(\mathbf{I} - vv^{\mathrm{T}} \right)}{|| \, \xi_i{}^{\mathrm{T}} \left(\mathbf{I} - vv^{\mathrm{T}} \right) \, ||} \, v^{\mathrm{T}} \xi_i. \tag{8}$$

For orthographic projections, we allow r to vary while keeping v fixed. This changes the result only slightly. Taking the gradient of equation (6) with respect to r, we have

$$\nabla s_i^{\text{ortho}}(r) = -\frac{\xi_i{}^{\mathrm{T}} \left(\mathbf{I} - vv^{\mathrm{T}} \right)}{|| \, \xi_i{}^{\mathrm{T}} \left(\mathbf{I} - vv^{\mathrm{T}} \right) \, ||}. \tag{9}$$

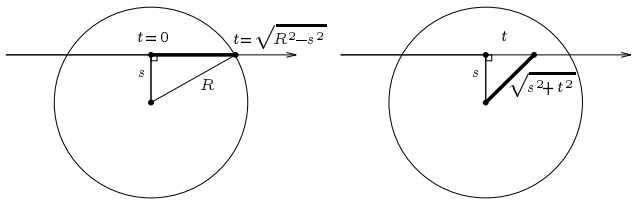
See Appendix A for derivations of the above expressions. Note that equation (9) is simply the normalized vector orthogonal to the ray (r, v) and directed away from the point p_i. The corresponding expression for perspective projection simply scales this vector by the distance along the ray, given by $v^{\mathrm{T}} \xi_i$.

2.2 Computing $\mathcal{F}_i$ and $\mathcal{F}_i'$

The function $\mathcal{F}_i(s)$ from equation (4) is the line integral of the filter function $f_i(\sigma)$ along a ray that passes within s of the data point p_i. Let R_i be the radius of support for $f_i(\sigma)$. Then $\mathcal{F}_i(s)$ is given by

$$\mathcal{F}_i(s) = 2 \int_0^{\sqrt{R_i^2 - s^2}} f_i(\sqrt{s^2 + t^2}) \, dt. \tag{10}$$

This expression results from some identities in the plane containing the ray and the data point p_i. These identities are demonstrated in Figure 2. Without loss of generality, we may assume that the point $t = 0$ on the ray is closest to p_i. This reparameterization is permissible since the emission-only line integral is independent of translation along the ray. The domain of integration is indicated by the bold line in Figure 2a. Note that because of symmetry, we can integrate along half of the ray's path through the spherical support region and multiply the result by two. At each point in the domain of integration, the filter is evaluated using the distance to the point p_i, given by $\sqrt{s^2+t^2}$. See Figure 2b.



(a) (b)

Figure 2: *Determination of the quantities in equation (10). The circles represent the support of the filter kernel. (a) The limits of integration. (b) The distance from a point t along the ray to the sample point.*

For arbitrary functions $f_i(\sigma)$, equation (10) can be neither evaluated nor differentiated in closed form. However, both operations can be performed symbolically for all polynomial filters, and the quadratic filter that we introduce below yields a particularly simple expression.

2.2.1 A Simple Polynomial Filter

In this section, we derive $\mathcal{F}(s)$ and $\mathcal{F}'(s)$ for a quadratic interpolation filter; the resulting images will be C^1 continuous when used in concert with emission-only rendering. This filter, which was used to compute all the results presented in this paper, is given by

$$f_i(\sigma) = 1 - \left(\frac{\sigma}{R_i}\right)^2. \tag{11}$$

Note that this filter approaches zero at the boundary of the spherical support, where $\sigma = R_i$.

The corresponding functions $\mathcal{F}_i(s)$ and $\mathcal{F}'_i(s)$ for this filter could be computed directly from equation (10); however, here we shall derive them in a way that simplifies the treatment of all polynomial filters. We observe that linear operators can be distributed across the terms of a polynomial. Thus, the integral operator in equation (10) applied to a polynomial of degree N is a linear combination of the transformed monomials, $\left\{s^0, s^1, \ldots, s^N\right\}$. For $f_i(\sigma)$ defined above, we have

$$f_i(\sigma) = \sigma^0 + \frac{1}{R_i^2}\sigma^2, \tag{12}$$

so we require expressions for the monomials of degree 0 and 2.

The zero$^{\text{th}}$-order monomial corresponds to a spherical filter of constant value. Thus, $\mathcal{F}_{i,0}(s)$ is simply the length of the ray segment that passes through the sphere. From Figure 2a we see that

$$\mathcal{F}_{i,0}(s) = 2\sqrt{R_i^2 - s^2}, \tag{13}$$

$$\mathcal{F}'_{i,0}(s) = \frac{-2s}{\sqrt{R_i^2 - s^2}}, \tag{14}$$

when $s \leq R_i$. Both functions are zero when $s > R_i$.

Next, we consider the second-order monomial, $f_i(\sigma) = \sigma^2$. By equation (10) we have

$$\mathcal{F}_{i,2}(s) = \frac{2}{3}\left(R_i^2 + 2s^2\right)\sqrt{R_i^2 - s^2}, \tag{15}$$

$$\mathcal{F}'_{i,2}(s) = 2\sqrt{\frac{R_i^2 s - 2s^3}{R_i^2 - s^2}} \tag{16}$$

for $s \leq R_i$. Note that both $\mathcal{F}_{i,0}(s)$ and $\mathcal{F}_{i,2}(s)$ are continuous, but that the derivative of each is unbounded as $s \to R_i$ from below. This singularity is removed by the quadratic filter in equation (11). From equations (13)-(16) we obtain

$$\mathcal{F}_i(s) = \mathcal{F}_{i,0}(s) - \frac{1}{R_i^2}\mathcal{F}_{i,2}(s) \tag{17}$$

$$= \frac{4\left(R_i^2 - s^2\right)^{\frac{3}{2}}}{3R_i^2}, \tag{18}$$

and

$$\mathcal{F}'_i(s) = \mathcal{F}'_{i,0}(s) - \frac{1}{R_i^2}\mathcal{F}'_{i,2}(s) \tag{19}$$

$$= \frac{-4s\sqrt{R_i^2 - s^2}}{R_i^2}. \tag{20}$$

When $s > R_i$ both $\mathcal{F}_i(s)$ and $\mathcal{F}'_i(s)$ are zero. Graphs of the functions f_i, $\mathcal{F}_i$, and $\mathcal{F}'_i$ for this filter are given in Figure 3.

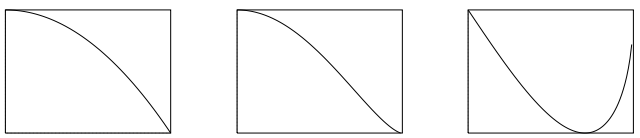


Figure 3: *Graphs of the quadratic filter function with a radius of 1. The horizontal axis for all graphs is $[0, 1]$. The vertical axis of the graphs of $f_i(\sigma)$ and $\mathcal{F}(s)$ is $[0, 1]$ while that of $\mathcal{F}'(s)$ is $[-2, 0]$.*

2.2.2 Other Filters

Aside from spherical-symmetry, the only requirement we place on our filter kernels that that they do not introduce value or derivative discontinuities in the image plane. These discontinuties can be avoided by using filters for which $\mathcal{F}(s)$ and $\mathcal{F}'(s)$ are continuous and approach zero at the boundary.

Many simple filters result in significantly more complicated expressions for $\mathcal{F}(s)$ and $\mathcal{F}'(s)$ than those given above. For example, the expressions for odd-powered monomials include logarithmic terms that make them more costly to evaluate. For the first-order monomial we have

$$\mathcal{F}_{i,1}(s) = \frac{1}{R_i}\left(cR_i - s^2\ln\left(c + R_i\right) + s^2\ln\left(s\right)\right), \tag{21}$$

where $c \equiv \sqrt{R_i^2 - s^2}$. Differentiating this expression complicates it further. Another filter that gives rise to complex analytic expressions is a windowed Gaussian. The resulting functions $\mathcal{F}_i(s)$ and $\mathcal{F}'_i(s)$ include multiple terms involving the error function.

In the above cases, as well as for many others where analytic expressions are unavailable, numerical integration becomes necessary. In these cases we may resort to precomputed tables and interpolation. For instance, both $\mathcal{F}(s)$ and $\mathcal{F}'(s)$ can be approximated to high accuracy using polynomial splines. This approach is similar in some respects to precomputing generic footprint tables for splatting [17].

2.3 Image Space Gradients

In equation (5), the intensity gradient is expressed in object space. In this section, we compute the 2D image-space gradient that is used by the applications of Section 3.

We express the image space gradient by

$$\nabla\phi(r(q), v(q)), \tag{22}$$

where q is a 2D vector representing image coordinates. Note that the gradient in equation (22) is taken with respect to q. The functions $r(q)$ and $v(q)$ give the object space origin and normalized direction vector for a view ray that passes through q.

To determine the relation between the image space gradient and the world space gradient, we need to specify the functions $r(q)$ and $v(q)$. Let $\mathbf{N}$ denote the view direction and let $\mathbf{A}$ be the 3×2 matrix that maps image coordinates to the 3D object space viewing plane; thus, the columns of $\mathbf{A}$ and the vector $\mathbf{N}$ are mutually orthogonal. For perspective projection we define the function $u : \mathbb{R}^2 \rightarrow \mathbb{R}^3$,

$$u(q) \equiv \mathbf{N} + \mathbf{A}q, \tag{23}$$

that maps image coordinates to their corresponding un-normalized ray direction vectors. Thus, we have

$$\begin{aligned} r(q) &= r_0 \\ v(q) &= \frac{u(q)}{||\,u(q)\,||}, \end{aligned} \tag{24}$$

for some fixed r_0. The image space gradient in equation (22) can now be expressed in terms of the object space Gradient $\phi(r_0, v)$ by applying the chain rule. After simplifying, we have

$$\nabla\phi(r(q), v(q)) = \nabla\phi(r_0, v)\,|_{v(q)}\,\frac{\mathbf{A}}{||\,u(q)\,||}. \tag{25}$$

In equation (25) the object space gradient, which can be computed via equations (5) and (8), is mapped to the 2D image plane by the matrix $\mathbf{A}$.

For orthographic projections, we define

$$\begin{aligned} r(q) &= \mathbf{A}q \\ v(q) &= \mathbf{N}, \end{aligned} \tag{26}$$

which results in

$$\nabla\phi(r(q), v(q)) = \nabla\phi(r, \mathbf{N})\,|_{r(q)}\,\mathbf{A}. \tag{27}$$

Here the 3D gradient is computed via equations (5) and (9).

3 Applications

The availability of an analytic expression for the intensity gradient in image space creates opportunities for new volume visualization techniques. In this section we describe the iso-contour method for volume rendering, which can be used to extract image features using only a sparse sampling of the image plane.

All of the approaches described here can also be implemented using discrete approximations to the gradient, computed by post-processing a regularly-sampled image. This type of approximation is fraught with difficulties, however, as the intensity gradient may vary more rapidly than the intensity values themselves. The use of analytic formulæ greatly enhances the numerical stability and accuracy of algorithms based on gradients. In addition, analytically computed gradients can obviate the need for computing an entire image, confining computation to small neighborhoods of the desired contours.

We present results using two sample datasets. The first is a simple contrived dataset consisting of 160 irregularly-spaced sample points arranged as 20 radial spines. We refer to this as the "urchin" dataset because it is reminiscent of the spiny sea creature. The second dataset consists of actual tomography data of a human pelvis. It contains 35,873 nonzero sample points on a regular rectilinear grid.

3.1 Generating Iso-Contours

A straightforward approach to following iso-contours begins by taking a small step from a given point in the image plane in a direction perpendicular to the image space gradient. While the gradient indicates the direction of maximum change, a vector perpendicular to the gradient indicates a direction in which the image intensity remains constant to first order. The new *predicted* point is likely to be near the iso-curve, although not exactly on it. See Figure 4. Starting from the predicted point, we then attempt to locate a nearby point that is within a specified tolerance of the curve. This second phase, known as the *corrector* step, also uses gradients to quickly converge to the desired isovalue.

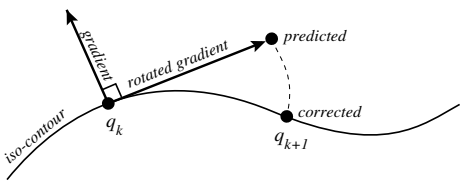


Figure 4: *The transformed intensity gradient is used to predict the next point along the curve. The accuracy is improved using a corrector.*

More formally, curves of constant intensity are functions of one variable, $q : \mathbb{R} \rightarrow \mathbb{R}^2$, which satisfy an ordinary differential equation (ODE) of the form

$$\dot{q} = \mathbf{M}\nabla\phi^{\mathrm{T}}(r(q), v(q)), \tag{28}$$

where $\mathbf{M}$ is a 2×2 90-degree rotation matrix, and $\nabla\phi(r(q), v(q))$ is the image space gradient, given by equation (25) for perspective projections, or equation (27) for orthographic projections. Given the initial condition $q(0) = q_0$, the solution q of the ODE is an iso-contour with intensity $\phi_0 = \phi(r(q_0), v(q_0))$.

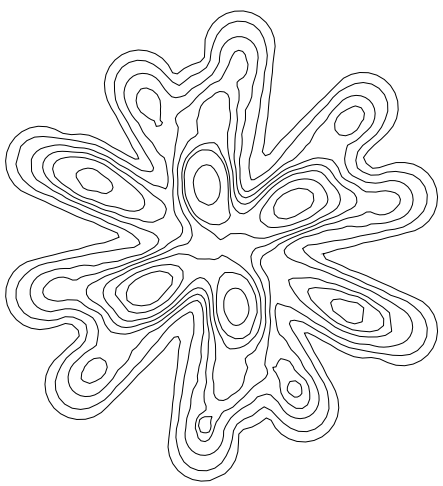


Figure 5: *Directly generated iso-contours for the non-rectilinear "urchin" test data, which consists of 20 radial spines.*

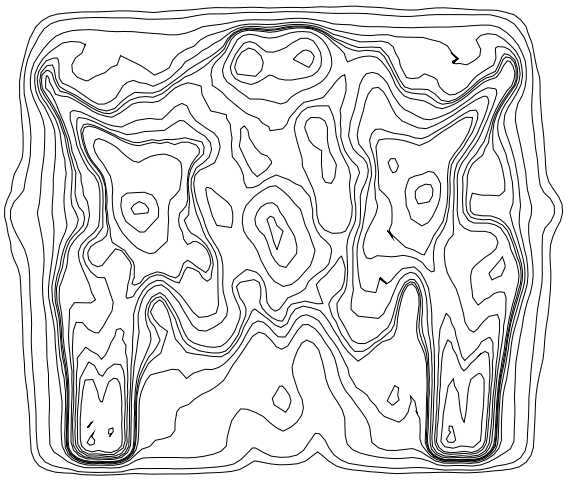


Figure 6: *A family of iso-contours generated directly from CT data of a human pelvis.*

Phrasing the problem as a first-order ODE is useful because there are many standard techniques for solving such equations numerically. The algorithm described above is the simplest in a larger class of methods known as *predictor-corrector methods*. These methods are well suited to iso-contour volume rendering because they require relatively few gradient evaluations, which are generally expensive to obtain in volume rendering applications. Our implementation, corresponds to an Euler predictor coupled with a Newton corrector. More sophisticated predictors may be used for greater efficiency [1, 2]. Several iso-contours that were identified in each of the test datasets are shown in Figures 5 and 6.

In the pseudo-code below, we present the curve following algorithm. The function "ComputeIntensity" corresponds to equation (4) while "ComputeGradient" generates an image-space gradient using either equation (25) or equation (27). To enhance numerical stability, the "GenerateIsoContour" routine contains an inner loop that implements a damping algorithm. It progressively halves the predictor's step size until the corrector converges. Therefore, the corrector must signal a convergence failure after some number of iterations have been exceeded.

```
GenerateIsoContour( ImagePoint q )
    Scalar φ_0 ← ComputeIntensity( q )
    begin a new contour of intensity φ_0 at the point q
    repeat
        Gradient g ← ComputeGradient(q)
        Scalar α ← user-supplied step size
        repeat
            q̂ ← Predictor(q, g, α)
            q̂ ← Corrector(q̂, φ_0)
            α ← α/2
        until corrector converges
        q ← q̂
        Add q to the contour
    until contour closes
    Add completed contour to family

Predictor( ImagePoint q, Gradient g, Scalar α )
    Gradient g' ← Rotate( g )
    ImagePoint q̂ ← q + α g'/ || g' ||
    return q̂

Corrector( ImagePoint q, Scalar φ )
    ImagePoint q̂ ← q
    Scalar φ̂ ← ComputeIntensity( q )
    while |φ − φ̂| > ε do
        g ← ComputeGradient(q̂)
        q̂ ← q̂ + (φ − φ̂) g/ || g ||^2
        φ̂ ← ComputeIntensity( q̂ )
        end while
    return q̂
```

3.2 Locating Important Curves

Thus far our discussion has not addressed the problem of finding starting points for generating useful iso-contours. One approach is to generate contours starting from interactively specified points in the image plane. In this way, useful contours can be discovered by trial and error; each new contour revealing more about the image and guiding subsequent trials. To further assist the user, we have implemented several semi-automatic tools as part of our interactive iso-contour exploration environment.

The first tool is an image extremum locator. Extrema provide a quick impression of image topology, since the final contours will encircle these points. Using an extremum locator, we can detect the border between data and background, as well as data hot-spots. Extremal values are stored as single-point contours in our data structures for use in subsequent rendering. Our current algorithm employs the gradient as part of a descent method to locate local extrema. Efficiency and accuracy can be enhanced by using other numeric techniques such as more sophisticated line searches [1]. Ultimately, second derivatives should be used to accelerate the convergence of the search algorithm, as simple descent methods converge slowly at extrema and perform poorly in flat regions.

The magnitude of the image intensity gradient is often used as an "edge strength" operator in image processing applications. In volume rendering, a three-dimensional analog to this idea has been used to find surfaces in object-space [5, 9]. We make use of this by generating more contours in regions where the gradient is large, since these are likely to be important feature boundaries in image space. We have implemented a tool that allows the user to examine the gradient magnitude along lines in screen-space. By choosing a peak gradient value, a contour through the corresponding screen-space point is traced.

Another tool that has proven to be valuable is a contour "in-betweener". This tool searches an existing family of contours for the two nested contours with maximal value difference and creates a new contour with intensity midway between them. The seed point for the new contour is located using Newton's method. In-betweening can be used to automatically refine a coarse set of contours. After repeated application of the algorithm, the contours become approximately equally-spaced in value; the density of the contours then gives a visual indication of slope, much as a topographic map.

3.3 Rendering

Given any set of iso-contours the underlying image can be rendered in several ways, then updated progressively as more contours are added. Because the contours are not associated with any particular rasterization of the image plane, the rendering may be performed at any resolution.

The simplest method of rendering iso-contours is to sort them by inclusion and draw them in back-to-front order, filled according to the corresponding intensity level. An example of this technique is shown in Figure 7. While this method is very simple, the value discontinuities can be visually distracting. Smoother renderings may

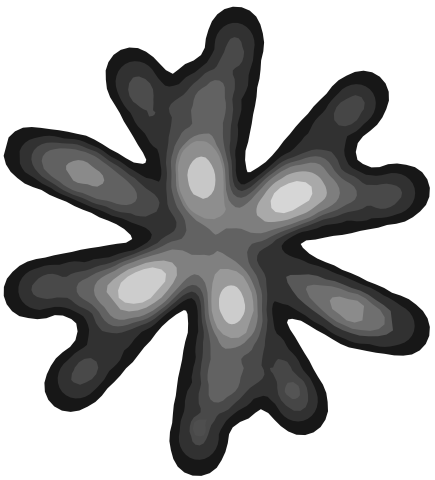


Figure 7: *Filled iso-contours for the "urchin" test data.*

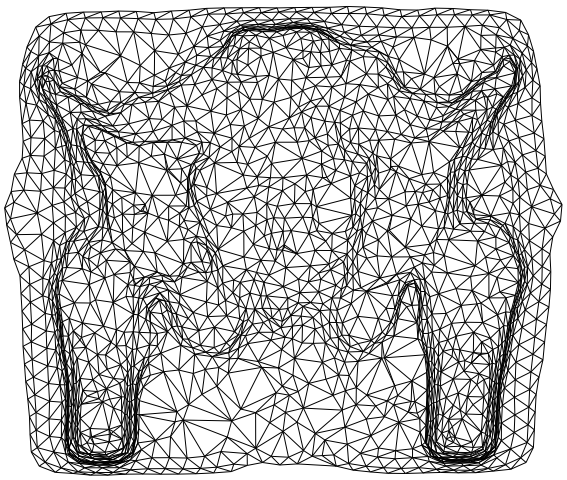


Figure 8: *A mesh for a front view of the CT pelvis data, generated from iso-contours with constrained Delaunay triangulation.*

be produced by first meshing the iso-contour grid into triangles, then rendering the triangles by interpolating the vertex values. We use Delaunay triangulation [6] to create the triangle mesh from the set of points defining the contours. The segments along the contours are constrained to be edges of triangles, which guarantees that no triangle will cross an iso-contour. This constraint method was used earlier by Lischinski et al. [10] in connection with discontinuity meshing for radiosity. An example of a mesh generated in this way is given in Figure 8. Images of the "urchin" and pelvis datasets made by linearly interpolating the triangular meshes are shown in Figures 9 and 10. Note that Figures 7 and 9 were computed using the same set of iso-contour data. Comparing the two images, the benefits of interpolation become apparent.

The image generation methods presented in this section are useful for generating images based on a sparse sampling of the data. Yet, unless the contours are spaced at intervals approximating the gray levels of the display device, the detail falls short of that attainable via traditional volume rendering techniques based on pixel-by-pixel reconstruction. We anticipate that this disparity can be lessened by more intelligent placement of the iso-curves coupled with better interpolation of the resulting triangular mesh. Currently, the gradient information this is available at the vertices of the triangles is ignored; this information could be used to advantage with higher-order interpolation methods.

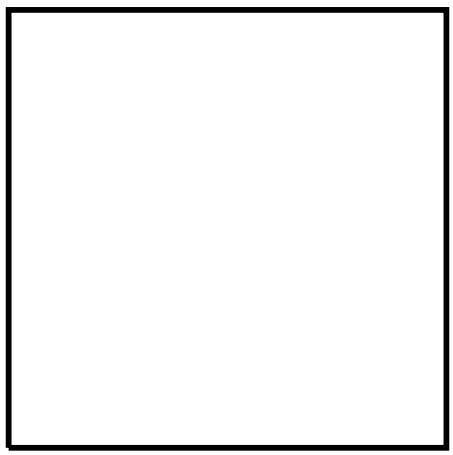


Figure 9: *An image created by linearly interpolating the mesh elements generated from the "urchin" iso-contours.*

4 Optimizations for Large Datasets

A key consideration in all volume rendering applications is the ability to handle massive amounts of data. In this section, we consider optimization of the gradient computation algorithm with respect to both computation time and memory consumption. The optimizations exploit the coherence of the data access inherent in iso-contour volume rendering. Other potential optimizations are discussed in Section 5.

As we saw in Section 2, the intensity and the intensity gradient at each point in screen space can be affected by any or all of the data samples; this is evident in the summations in equations (4) and (5), which are taken over all data points. For filters with finite support, however, the summations need only be taken over data points whose spheres of influence are intersected by the given ray. If this subset of points were known *a priori* for each ray, the savings could be considerable. For uniformly-distributed data points, the average-case complexity of computing ϕ and $\nabla\phi$ would drop from $O(n)$ to $O(\sqrt[3]{n})$. A practical means of approaching this is to associate with each ray a substantially reduced subset of data points, or *candidate set*, which is guaranteed to contain all relevant data.

There are two classes of algorithms for quickly generating candidate sets: those that operate in screen-space, and those that operate in object-space. Screen-space algorithms generate candidate sets that are valid for a given view. During image or iso-curve generation, a candidate set is retrieved from the screen-space region pierced by a given ray. All candidate sets are invalidated whenever the view changes.

In contrast, object-space approaches classify the data points into 3D regions, which remain valid for all views. For a given ray, the candidate set of data points can be constructed using any ray tracing acceleration scheme for locating the regions pierced by the ray [4]; for instance, the data may be organized into an octree, kd-tree, or a uniform lattice. While the object-space organization need only be computed once, the candidate set must be created anew for each ray by tracing it through the volume. Consequently, object-space methods do not fully exploit image-space coherence.

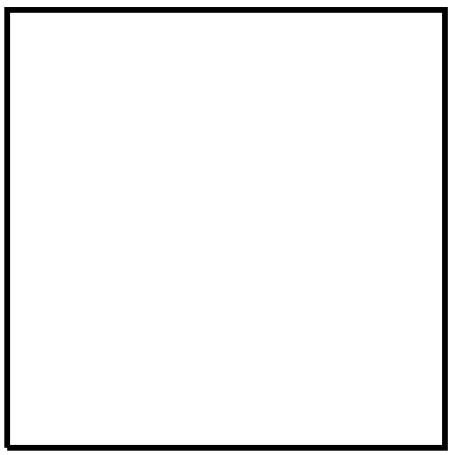


Figure 10: *An image created by linearly interpolating the pelvis contours using a constrained Delaunay triangulation.*

We have employed screen-space algorithms because they very naturally exploit the coherence in the data access pattern that results from following a curve. Aside from the seed point, each evaluation of ϕ and $\nabla\phi$ occurs at a point very near the previous point. Top-down subdivision of the screen into regions can be done using a Warnock-style divide-and-conquer approach [16]. If used to lazily create candidate sets, this approach is essentially a 2D version of the ray-classification algorithm [3].

One disadvantage of the screen-space approach is that each data point typically belongs to multiple candidate sets. For small datasets, this replication poses no problem. For large datasets, however, it is often impractical to store all the candidate sets at once. To remedy this, we have introduced a cache that stores only a few candidate sets at a time; the localized data access pattern virtually ensures a high cache hit rate. In the current implementation, we divide the screen into 16×16 rectangular regions. The cache consists of a list of pointers to the data points that affect a 3×3 block of cells centered on a given ray. Each successive ray is first tested against the cache; if it is within the 9 cells, only the cached data points are tested. When a ray leaves the block of 9 cells, the cache list is updated by re-centering it on the current ray, and a bit-vector called the *cache mask* is reset to indicate the cells currently covered by the cache.

To be effective, it is essential that the cache lists be computed quickly. The process can be streamlined by associating a bit vector with each data sample as a view-dependent preprocess. This bit vector indicates which screen-space cells are affected by the data sample. The cache lists can then be quickly computed as needed using both the bit vectors and the cache mask. The process still requires an $O(n)$ search, but a distance comparison is completely avoided except for the relatively small number of points whose masks indicate a potential hit.

Some results using these optimizations are summarized in the table below. All tests were run on a 55 MIPS HP-720 workstation. Three different contours of the pelvis selected from the family shown in Figure 6 were considered. Loop A is a contour near the perimeter of the dataset. Loops B and C are interior contours. Loop B is short and oval-shaped, whereas Loop C is long and convoluted. The combination of masking and caching results in a speedup of more than an order of magnitude for all three cases.

Loop	A	B	C
Predictor steps	219	125	273
Corrector steps	181	232	487
Unoptimized trace time (sec)	66.0	53.5	130.5
Trace time with mask	15.5	13.2	36.0
Trace time with mask and cache	2.2	3.6	7.6

The modified algorithm for computing the gradient based on caching and bit vectors is outlined in pseudo-code below.

```
ComputeObjectSpaceGradient( ScreenPoint q )

    m ← bit mask encoding screen position of q
    if m ∉ Cache.Mask then Cache.Update(m)
    Gradient g ← (0, 0, 0)
    for each data point p ∈ Cache.List do
        if m ∈ p.Mask then
            s ← distance from p to ray through q
            if s < radius of filter about p then
                compute F'(s)
                compute ∇s
                g ← g + p.Weight F'(s) ∇s
                end if
            end if
        end for
    return g
```

5 Conclusions and Future Work

We have presented a method for computing screen-space curves of constant intensity, or iso-curves, for emission-only projections of volume data. The method operates by computing intensity gradients at discrete points in the image plane. The gradients are computed analytically for quadratic radially symmetric filter kernels, with arbitrary radii, and applied to data with any spatial organization. The curves are solved using a predictor-corrector method that can maintain any given tolerance along arbitrarily long iso-contours. The computed contours can be used to locate specific features of the image, or to create resolution-independent renderings.

The most severe limitation of the current implementation is its restriction to emission-only rendering, which can be likened to X-ray imaging. While absorption rendering is a minor extension, requiring the application of an exponential function to the final intensities, combining both emission and absorption within a single model is significantly more difficult. Use of both emission and absorption implies that the data points cannot be processed independently: both ordering and overlap of neighboring filter kernels must be accounted for. These complications make the analytic computation of the intensity gradient a significant challenge.

More general filter kernels are another avenue to pursue. While it may be impractical to compute intensity gradients for some reconstruction methods, such as tricubic interpolation applied to regular grids, there are other kernel types that can be accomodated quite naturally. For instance, elliptical kernels can be handled as easily as spherical kernels by applying a linear distortion to the rays and resulting gradients.

There are many efficiency schemes that have yet to be investigated. The curve-following algorithm would benefit greatly from a higher-order predictor and corrector. A multi-step predictor, such as Milne's method [1, 2] could greatly reduce the number of corrector steps needed. A look-ahead cache could reduce the number of cache updates by using the current direction and curvature of the contour. A hierarchical cache structure could further reduce the number of $O(n)$ cache updates by using a nested sequence of cache regions. Finally, inexact- and quasi-Newton methods could be employed in the corrector step to reduce the number and cost of the gradient computations, especially while the predicted point is relatively far from the curve.

Our current algorithms start with a seed point and find either individual iso-contours or local intensity extrema. Ideally, the algorithm would automatically produce global information, such as a list of all level curves for a given value or a list of all image extrema. With appropriately chosen contour intensities, this would result in truly resolution-independent images. One way to obtaining this information is by means of interval arithmetic [8, 14]. As demonstrated by Snyder [14], interval arithmetic can also be used to produce a more robust curve follower, eliminating the need for the user-supplied step-size parameter that we currently employ.

Appendix A: Computing the Distance Gradient

To derive equations (8) and (9) we introduce two vector identities. Let a and b be vectors, and let $f(a) \equiv ||\, a \,||$ and $g(a) \equiv aa^{\mathrm{T}}b$. Then

$$Df(a) = \frac{a^{\mathrm{T}}}{||\, a \,||}, \tag{29}$$

and

$$Dg(a) = \mathbf{I}a^{\mathrm{T}}b + ab^{\mathrm{T}}. \tag{30}$$

Using these identities, equation (6), and the chain rule, we have

$$\begin{aligned} \nabla s_i^{\text{persp}}(v) &= \nabla_v \left|\left| \, (\mathbf{I} - vv^{\mathrm{T}})\xi_i \, \right|\right| \\ &= \frac{{\xi_i}^{\mathrm{T}}(\mathbf{I} - vv^{\mathrm{T}})}{||\, {\xi_i}^{\mathrm{T}}(\mathbf{I} - vv^{\mathrm{T}}) \,||} D_v(\mathbf{I} - vv^{\mathrm{T}})\xi_i \\ &= \frac{{\xi_i}^{\mathrm{T}}(\mathbf{I} - vv^{\mathrm{T}})}{||\, {\xi_i}^{\mathrm{T}}(\mathbf{I} - vv^{\mathrm{T}}) \,||} \left(-\mathbf{I}v^{\mathrm{T}}\xi_i - v{\xi_i}^{\mathrm{T}}\right). \end{aligned} \tag{31}$$

But $\mathbf{I} - vv^{\mathrm{T}}$ is a projection that takes the vector v to zero. Therefore, the $v{\xi_i}^{\mathrm{T}}$ term in the right-most factor drops out, and we are left with

$$\nabla s_i^{\text{persp}}(v) = -\frac{{\xi_i}^{\mathrm{T}}(\mathbf{I} - vv^{\mathrm{T}})}{||\, {\xi_i}^{\mathrm{T}}(\mathbf{I} - vv^{\mathrm{T}}) \,||} \, v^{\mathrm{T}}\xi_i. \tag{32}$$

Similarly, for orthographic projection, we have

$$\begin{aligned} \nabla s_i^{\text{ortho}}(r) &= \nabla_r \left|\left| \, (\mathbf{I} - vv^{\mathrm{T}})\xi_i \, \right|\right| \\ &= \frac{{\xi_i}^{\mathrm{T}}(\mathbf{I} - vv^{\mathrm{T}})}{||\, {\xi_i}^{\mathrm{T}}(\mathbf{I} - vv^{\mathrm{T}}) \,||} (\mathbf{I} - vv^{\mathrm{T}}) \, D_r\xi_i. \end{aligned} \tag{33}$$

Since $D_r\xi_i = -\mathbf{I}$, and the projection $\mathbf{I} - vv^{\mathrm{T}}$ is idempotent, this simplifies to

$$\nabla s_i^{\text{ortho}}(r) = -\frac{{\xi_i}^{\mathrm{T}}(\mathbf{I} - vv^{\mathrm{T}})}{||\, {\xi_i}^{\mathrm{T}}(\mathbf{I} - vv^{\mathrm{T}}) \,||}. \tag{34}$$

Acknowledgments

James Arvo was supported by the NSF grant "Interactive Computer Graphics Input and Display Techniques" (CCR-8617880), by the NSF/ARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219) and an equipment grant from the Hewlett-Packard Corporation. Kevin Novins was supported by a French government Chateaubriand Scholarship. The Delauney triangulation code was provided by Dani Lischinski, and the CT data was provided by the Department of Radiation Oncology at the North Carolina Memorial Hospital. The authors also wish to thank Erin Shaw and François Sillion for their careful reading of the manuscript.

REFERENCES

[1] ACTON, F. S. *Numerical Methods that Work*. Harper & Row, New York, 1970.

[2] ARVO, J. The irradiance Jacobian for partially occluded polyhedral sources. In *Computer Graphics* Proceedings (1994), Annual Conference Series, ACM SIGGRAPH.

[3] ARVO, J., AND KIRK, D. Fast ray tracing by ray classification. *Computer Graphics 21*, 4 (July 1987).

[4] ARVO, J., AND KIRK, D. A survey of ray tracing acceleration techniques. In *An Introduction to Ray Tracing*, A. S. Glassner, Ed. Academic Press, New York, 1989, ch. 6.

[5] DREBIN, R. A., CARPENTER, L., AND HANRAHAN, P. Volume rendering. *Computer Graphics 22*, 4 (Aug. 1988), 65–74.

[6] GUIBAS, L., AND STOLFI, J. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Transactions on Graphics 4*, 2 (April 1985), 74–123.

[7] HELMAN, J. L., AND HESSELINK, L. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications 11*, 3 (May 1991).

[8] KALRA, D., AND BARR, A. H. Guaranteed ray intersections with implicit surfaces. *Computer Graphics 23*, 3 (July 1989), 297–306.

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

[10] LISCHINSKI, D., TAMPIERI, F., AND GREENBERG, D. P. Combining hierarchical radiosity and discontinuity meshing. In *Computer Graphics* Proceedings (1993), Annual Conference Series, ACM SIGGRAPH, pp. 199–208.

[11] LORENSEN, W. E., AND CLINE, H. E. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics 21*, 4 (July 1987).

[12] MALZBENDER, T. Fourier volume rendering. *ACM Transactions on Graphics 12*, 3 (July 1993).

[13] SHEROUSE, G. W., NOVINS, K. L., AND CHANEY, E. L. Computation of digitally reconstructed radiographs for use in radiotherapy treatment design. *International Journal of Radiation Oncology, Biology and Physics 18*, 3 (March 1990), 651–658.

[14] SNYDER, J. M. Interval analysis for computer graphics. *Computer Graphics 26*, 2 (July 1992), 121–130.

[15] TOTSUKA, T., AND LEVOY, M. Frequency domain volume rendering. *Computer Graphics* (August 1993), 271–278.

[16] WARNOCK, J. E. *A Hidden-Surface Algorithm for Computer Generated Half-tone Pictures*. PhD thesis, University of Utah, 1969. TR-4-15.

[17] WESTOVER, L. Footprint evaluation for volume rendering. *Computer Graphics 24*, 4 (August 1990), 367–376.