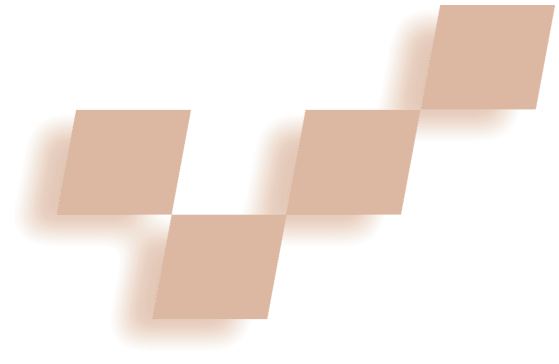


Image Reconstruction Using Data-Dependent Triangulation



Xiaohua Yu, Bryan S. Morse, and
Thomas W. Sederberg
Brigham Young University

Digital image reconstruction refers to the process of fitting a continuous intensity surface through discrete image samples. Reconstruction is a crucial first step in resampling operations like magnification. Such operations are basic to any commercial image-manipulation software. They prove useful in applications such as enhancing the resolution of low-

cost digital cameras or in preparing a low-resolution image for printing on a higher resolution printer.

This article considers the problem of how to best reconstruct an existing image using one sample per pixel. Figures 1b and 2 show examples created with our algorithm, contrasted with results from conventional methods in Figures 1a and 3.

Researchers have studied the image reconstruction problem extensively. Signal processing tools¹⁻³ can analyze the relative merits of approximating a discrete 1D signal using various methods such as nearest neighbor, linear interpolation, cubic splines, Gaussians, differences

of Gaussians, or windowed sinc functions. These 1D approximation methods are typically extended to 2D images by forming the tensor product of the basis functions. Thus, for example, linear interpolation from 1D signal processing becomes piecewise bilinear surface patches in 2D image reconstruction. However, because those bilinear patches all align with the image coordinate axes, their artifacts tend to reveal the underlying sampling grid, as Figure 1a shows.

This article presents a new method for image reconstruction using a piecewise linear intensity surface whose elements don't generally align with the coordinate axes. This method is based on the technique of data-dependent triangulation (DDT) that Dyn, Levin,

and Rippa⁴ introduced and has proven capable of producing more pleasing reconstructions than axis-aligned methods. For example, Figure 1a shows a digital image scaled by a factor of four using bicubic reconstruction. Figure 1b shows the same enlargement using this new reconstruction algorithm.

In addition to simple image magnification, DDT can also improve the appearance of texture maps viewed close up. The idea is to replace a traditional texture map with a mosaic of triangles with color varying linearly across each triangle. Figure 3 shows a texture-mapped plane rendered on SGI hardware; Figure 2 shows the texture map—at the same resolution—as a DDT mosaic, also rendered on SGI hardware.

This DDT-based reconstruction algorithm minimizes the visual impact of the reconstruction errors. However, no reconstruction method is perfect. If we create an image I_2 by reducing the pixel resolution of image I_1 by a factor of n and then create image I_3 by increasing the pixel resolution of I_2 by a factor of n , differences will generally exist between I_1 and I_3 , regardless of the reconstruction method used. While a DDT reconstruction may not dramatically reduce quantifiable error, it can significantly reduce the visibility of artifacts by moving them off of the sampling axes. This idea resembles how stochastic sampling, while not quantifiably reducing aliasing, reduces the visual effects by converting structured aliasing to apparent noise.

In addition to demonstrating how nicely DDT can address the reconstruction problem, this article also makes two contributions to the DDT literature: a new cost function and an improved optimization algorithm. These enhancements help improve DDT's performance with image reconstruction.

An important feature of our new cost function is that it yields pleasing results while being invariant to the range of intensities. Most suitable DDT cost functions are based on 3D geometric properties and hence are sensitive to the chosen range of intensity values $[0, I_{\max}]$.

Image reconstruction based on data-dependent triangulation with new cost functions and optimization can create higher quality images than traditional bilinear or bicubic spline reconstruction.

Depending on device- and software-specific properties, a given image may use $I_{\max} = 1$ in one situation or $I_{\max} = 2^{16}$ in another. At the same time, the x - y pixel coordinates lie fixed in, say, $[0,1024]$. Such intensity scaling leads most cost functions to produce different triangulations (and hence different interpolated images).

Related work

Triangulations are ubiquitous in computer graphics, and during the past decade researchers have made several significant theoretical advances in using triangulations for geometric modeling. The central method in this article—retriangulating a given set of vertices with the goal of minimizing a cost function—has several parallels in the geometric modeling literature. DDT adapts this idea to surfaces defined by scattered-data samples of functions $z = f(x, y)$.

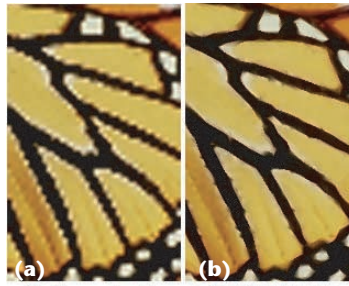
Image reconstruction using triangles isn't widely done, probably because bilinear reconstruction is slightly simpler to compute. When it's done, for applications such as mesh decimation, the triangulation chosen is almost always a Delaunay triangulation in which all triangles in the triangulation align with the coordinate axes. Bilinear reconstruction yields results arguably better than those from a piecewise linear Delaunay reconstruction. DDT has also been used for coding images using a sparse set of points. One approach⁵ adds successive points to the approximation until the desired encoding size is reached, in contrast to mesh decimation techniques commonly used in computer graphics.

This article's primary objective—to improve the ultimate appearance of resampled images, especially magnified images—has also been addressed using image-processing techniques. These methods typically fall into two categories: functional interpolation and filtering. Functional interpolation treats the intensity surfaces as a sampled 2D function and uses traditional polynomial spline (usually linear or cubic) fitting of the sampled points. For 2D images, these become bilinear and bicubic spline functions—an interpolation in x followed by an interpolation in y or vice versa. Filtering approaches build on sampling theory and attempt to undo the frequency-domain effects of the initial sampling.^{1,3}

A third class of methods has emerged recently, focusing on identifying visually significant properties of the image and reconstructing these first or instead. Edge-directed interpolation⁶ attempts to first identify significant edges in the image, performs a piecewise linear reconstruction of these edges, and then uses standard bilinear interpolation to interpolate the image data, taking care to avoid interpolating across image edges. More recent methods⁷ attempt not only to reconstruct edge contours but all level-set contours in the image. What's significant about these methods is that, like this article, they attempt to directly address and minimize the visual properties of the inevitable reconstruction error.

Data-dependent triangulation

Given a set of distinct (and not all collinear) points $V = \{(x_i, y_i)\}$ in the x - y plane, we're interested in a convex hull triangulation of V that's a set $T = \{T_i\}$ of nonde-



1 Four-power magnification of a butterfly image at 100 dpi: (a) bicubic reconstruction and (b) data-dependent triangulation (DDT) reconstruction.



2 DDT mosaic as a texture map.



3 Conventional texture map.

generate triangles that satisfies the following conditions:

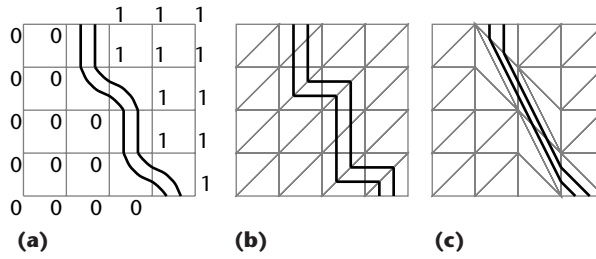
- Every triangle vertex is an element of V , and every element of V is a triangle vertex.
- Every edge of a triangle in T contains exactly two points from V .
- The union of all triangles in T is the convex hull of V .
- The intersection of any two different triangles in T is either empty, or is a shared edge or a shared vertex.

A convex hull triangulation proves the simplest way to produce a piecewise linear continuous surface that interpolates a set of data points. If each point (x_i, y_i) also has a z coordinate z_i , then any surface $z = f(x, y)$ for which $f(x_i, y_i) = z_i$ is called an interpolating surface of $f(x, y)$. Thus, any triangulation of V whose vertices move from $(x_i, y_i, 0)$ to (x_i, y_i, z_i) is a piecewise linear interpolating surface.

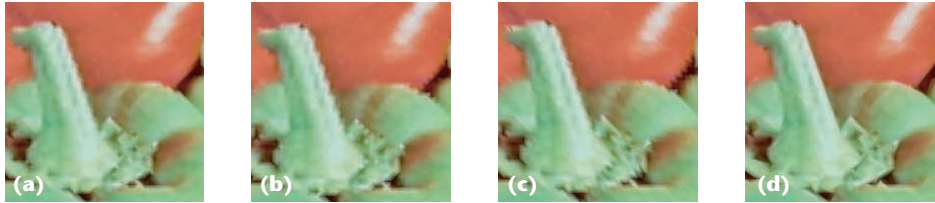
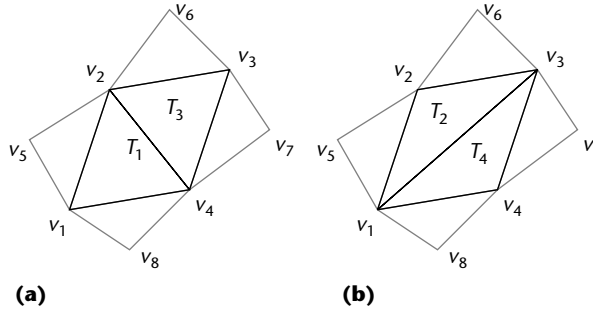
For an image, V is the set of pixel centers, $f(x, y)$ is the intensity of the pixel in which (x, y) is located (for a gray-scale image), and any piecewise linear interpolating surface of $f(x, y)$ over V can serve as an image reconstruction.

The number of triangulations is huge, and not all triangulations possess equally pleasing qualities. In many applications, the Delaunay triangulation is preferable.

4 Contours from different triangulations: (a) bilinear, (b) Delanauy, and (c) DDT.



5 Two ways to split a quadrilateral: (a) Initial triangulation and (b) after the edge swap.



6 Four-power magnification from DDT reconstructions using various cost functions: (a) angle between normals, (b) jump in normal derivatives, (c) deviations from linear polynomials, and (d) our cost function.

mines the relation $\gamma_1 < \gamma_2$ lexicographically. This process is a variation on an L_∞ norm.

Cost functions for gray-scale images

Our guiding strategy for devising a cost function is that reconstructions whose contours are smoothest tend to produce the least offensive artifacts. Figure 4a shows some contours resulting from bilinear reconstruction of a simple image of 25 pixels whose intensities are zero or one. Note that the vertices of the squares are pixel centers. (Since this is based on a bilinear reconstruction, it doesn't involve triangles.) Figure 4b shows some contours from a Delanauy triangulation of the image, and Figure 4c shows a triangulation whose contours are straighter.

Dyn et al.⁴ proposed numerous cost functions for DDT, of which the four classified as Nearly C^1 (NC1) appear to best meet our strategy. We'll review these with reference to the cost of edge $v_2 - v_4$ in Figure 5a. This edge is shared by triangles T_1 and T_3 , whose unit normals are n_1 and n_3 respectively, and whose interpolating linear functions are P_1 and P_3 respectively, where

$$P_i = a_i x + b_i y + c_i \tag{1}$$

This fact led to a common misconception that triangulations involving long, narrow triangles are bad in general. With their introduction of DDT, Dyn et al.⁴ showed that conventional wisdom is flawed.

In DDT, we aim to identify which triangulation of a given function $z = f(x, y)$ over a given set of points V will optimize some quality which, for our discussion, we'll refer to as smoothness. The smoothest triangulation will usually differ for two different functions over the same set of points V ; hence the name data-dependent triangulation.

We quantify smoothness in terms of a cost γ_i associated with interior edge i (or, in one approach, for each vertex⁸). The number of interior edges q is the same for any triangulation of a given V . Dyn et al.⁴ suggested three ways that the cost $C(T)$ of the entire triangulation T can be computed:

1. Use the L_1 norm, $C(T) = \sum_{i=1}^q |\gamma_i|$
2. Use the L_2 norm, $C(T) = \sum_{i=1}^q (\gamma_i)^2$
3. Arrange, in nondecreasing order, the edge costs of triangulation T_j in a vector Γ_j . Define an ordering such that for the edge-cost vectors Γ_1 and Γ_2 of triangulations T_1 and T_2 respectively, the order deter-

We denote the position and value of vertex v_k as (x_k, y_k) and I_k respectively, and the corresponding R^3 position as $w_k = (x_k, y_k, I_k)$. Also notice that in Figure 5a, v_k is the free (unshared) vertex of triangle T_k , ($k = 1, 3$). Using that notation, the four NC1 cost functions from Dyn et al.⁴ are as follows:

1. Angle between normals (ABN). This cost function is the cosine of the 3D angle between n_1 and n_3 , or $n_1 \cdot n_3$. Choi et al.⁹ also used this cost function.
2. Jump in normal derivatives (JND). This is given by

$$\|n_x(a_1 - a_3) + n_y(b_1 - b_3)\|$$

where (n_x, n_y) is a unit vector in the (x, y) plane orthogonal to the projection of edge $v_1 - v_2$ in the (x, y) plane.

3. Deviations from linear polynomials (DLP). This expresses how well P_1 predicts the value of v_3 and similarly, how well P_3 predicts the value of v_1 . DLP is given by $\|h\|$ where

$$h = \begin{bmatrix} P_1(x_3, y_3) - I_3 \\ P_3(x_1, y_1) - I_1 \end{bmatrix}$$

4. Distances from planes (DP). This measures the distance from v_4 to the plane of T_3 and from v_3 to the plane of T_4 . This distance is given by $\|g\|$ where

$$g = \begin{bmatrix} \text{dist}(P_1, w_3) \\ \text{dist}(P_3, w_1) \end{bmatrix}$$

$$\text{dist}(P_j, w_k) = \frac{|P_j(x_k, y_k) - I_k|}{(a_j^2 + b_j^2 + 1)^{1/2}}$$

We implemented these cost functions and found that ABN, JND, and DLP generally don't perform well for image reconstruction, as Figure 6 illustrates. The fourth cost function, DP, works pretty well for image reconstruction in many cases. But it produces somewhat different triangulations when we scale the intensity range, since it doesn't scale linearly with intensity.

Compared to the others, our cost function works well for image reconstruction and produces consistent triangulations regardless of the range used to represent intensities. Our cost function is based on the relationships between the level contours in the interpolated intensities of adjacent triangles. Because we interpolate the triangles linearly, these level contours are straight lines. We base the cost on the angle between the normals to the contour lines, which is invariant to the range of values used to represent intensity. Unlike Dyn's ABN cost function, we don't use the 3D normals to the triangles' planes. Instead, we use the image-plane projection of those normals.

We also weight the cost by the observability of these contour discontinuities based on the contour lines' relative contrast or density. Specifically, we weight the cost by the gradient magnitude of the intensity surface for each triangle.

Our cost function for the edge between triangles T_1 and T_3 (Figure 5a) with interpolating linear functions P_1 and P_3 respectively is

$$\text{cost} = \|\nabla P_1\| \|\nabla P_3\| (1 - \cos\theta) \quad (2)$$

where θ is the angle between the contour-line normals and

$$\|\nabla P_i\| = \sqrt{a_i^2 + b_i^2}$$

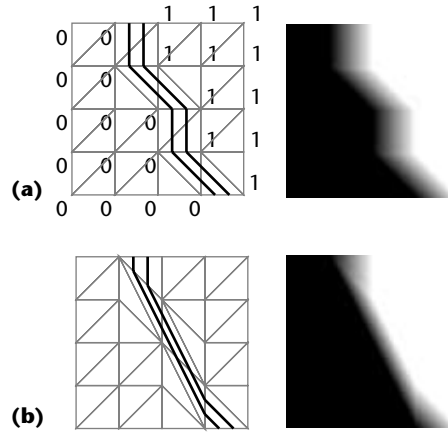
where a_i and b_i are the same as in Equation 1.

Recognizing that the directions of the contour normals and the gradient are identical, we can simplify this to

$$\text{cost} = \|\nabla P_1\| \|\nabla P_3\| \left(1 - \frac{\nabla P_1 \cdot \nabla P_3}{\|\nabla P_1\| \|\nabla P_3\|} \right)$$

$$= \|\nabla P_1\| \|\nabla P_3\| \frac{-\nabla P_1 \cdot \nabla P_3}{\|\nabla P_1\| \|\nabla P_3\|} \quad (3)$$

Although the gradient magnitude changes with the range used to represent intensities, scaling the range linearly causes linear scaling of the gradient magnitudes. Thus, a least-cost triangulation for one intensity range



7 Contours from two optimization algorithms: (a) Lawson's algorithm and (b) an edge swap with the look-ahead algorithm.



8 DDT using (a) Lawson's optimization, (b) bicubic reconstruction, and (c) DDT with look-ahead optimization.

is identical to the least-cost triangulation produced for a linear rescaling of that range.

Optimization

Ultimately, our cost function's success depends on our optimization routine's performance. Here we'll review the two standard published methods and present a new one that does a better job in finding optimal DDTs.

Lawson's local optimization

The simplest optimization algorithm for performing DDT is Lawson's algorithm.^{4,10} Each interior edge in a triangulation is shared by two triangles whose union forms a quadrilateral. If that quadrilateral is convex, we have two ways in which we can legally (according to the elements of a triangulation) split it into triangles. Lawson's algorithm visits each edge in a triangulation and checks whether its two neighboring triangles form a convex quadrilateral. If it's not convex, the algorithm does nothing. If it's convex (see Figure 5), the algorithm compares the sum of the five dark edges' costs in Figure 5a to those in Figure 5b. If a lower overall cost results, the algorithm swaps the edges. After the algorithm visits all edges in this manner, it conducts a second iteration of edge visits if it made any swaps during the first visit. This process continues until the overall cost can't be reduced.

This simple, fast algorithm converges to the globally optimal solution if the cost function is based on the Delaunay criterion. However, using most other cost functions, Lawson's algorithm only leads to a local optimum that usually isn't as satisfactory as what other optimizers can obtain. Figures 7a and 8b show some results from Lawson's algorithm.

9 Look-aheads for optimization: (a) first, (b) second, (c) third, and (d) fourth.

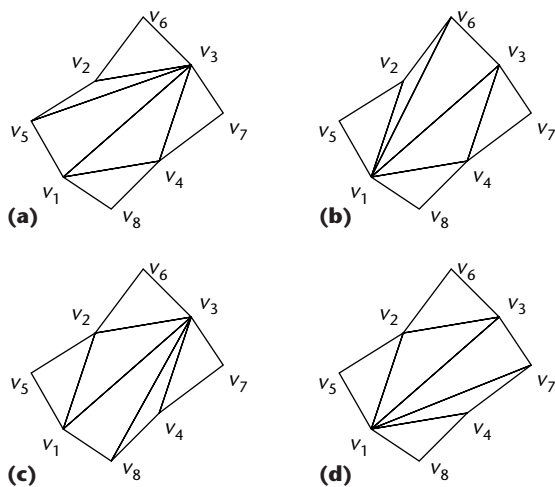


Table 1. Cost at each iteration of the optimization.

| Iteration | Lawson's Algorithm | Look-ahead Algorithm |
|-----------|--------------------|----------------------|
| 0 | 632.95 | 632.95 |
| 1 | 289.35 | 256.90 |
| 2 | 279.33 | 234.54 |
| 3 | 278.00 | 230.76 |
| 4 | 278.00 | 229.84 |

Simulated annealing

The only other optimization method in the DDT literature is Schumaker's discussion of simulated annealing.¹¹ We implemented the algorithm as described and found that it can generally find lower cost triangulations than Lawson's algorithm. However, the simulated annealing algorithm is $O(n^2m^2)$ whereas our optimization algorithm (presented in the next section) is $O(nm)$ for an image with an $n \times m$ pixel resolution. Not only does our algorithm run faster, it also usually finds a lower cost triangulation than simulated annealing (although neither can guarantee global optimality).

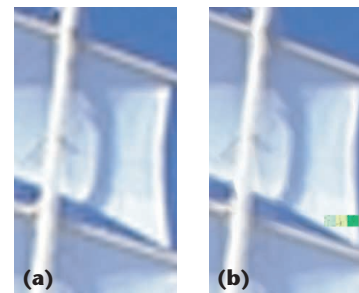
Edge swap with look-ahead

We've had good success with the following modification of Lawson's algorithm: If an edge swap can reduce the cost, then do it. If not, check whether that edge swap in conjunction with an edge swap of any of the four neighboring edges can reduce the cost.

Specifically, notice that we obtained the four triangulations in Figure 9 from the triangulation in Figure 5b by doing a single edge swap. Suppose the sum of the dark edges' costs in Figure 5a is less than the sum of the dark edges' costs in Figure 5b. Our look-ahead algorithm will then compare the total cost c_1 of all 13 edges in Figure 5b with the total cost c_2 of all 13 edges in Figure 9a. If $c_2 < c_1$, then the triangulation in Figure 9b replaces the triangulation in Figure 5a. Otherwise, the algorithm compares c_1 to the cost of all edges in Figure 9b, and then, if needed, to those in Figures 9c and 9d.

Using this look-ahead algorithm, we obtained the triangulation in Figures 7b and 8c. Contrast those with the

10 Reconstructions for the "sail" image: (a) bicubic and (b) DDT.



triangulations in Figures 7a and 8a that used Lawson's algorithm.

Table 1 shows the total triangulation cost for the triangulation used in Figure 3 (obtained by summing the cost of each edge in the triangulation) after the optimization algorithm performed each of the four iterations. The table shows the total cost decreases using Lawson's algorithm and the look-ahead algorithm. As you can see, the look-ahead algorithm produces a lower cost at each step of the iteration, and the optimization algorithm tends to converge in three or four iterations. In many cases, the results after one iteration may be satisfactory.

Each iteration takes between 0.5 and 5 seconds, depending on the cost function and optimization parameters used for an 80×80 image on a consumer-grade PC. This execution time scales linearly with the number of pixels in the image.

Color images

We can extend the results for gray-scale images to color in several different ways. One method is to compute a different triangulation for each color component and then evaluate each component independently. However, this is a bad idea because it can lead to color bleeding. Thus, we recommend using a single triangulation for all three components and suggest two DDT cost functions for finding that triangulation.

Cost function 1

One method that yields good results converts the RGB components of each pixel into a single gray-scale intensity using the standard conversion from RGB to luminance:

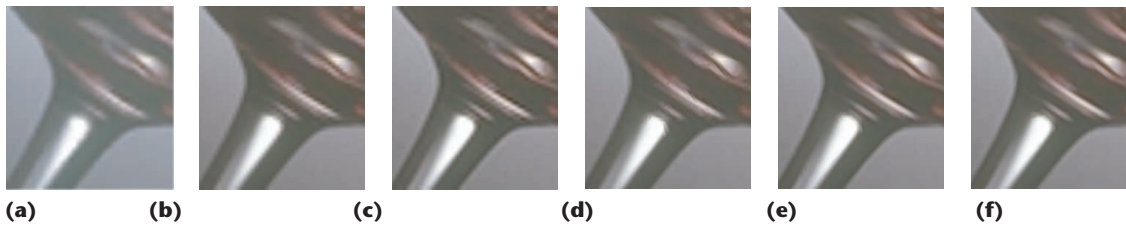
$$I = 0.21267R + 0.71516G + 0.07217B \tag{4}$$

You can compute the edge cost by using the cost function for gray-scale images discussed in the section "Cost functions for gray-scale images."

We can justify this method because in the human visual system, color doesn't contribute as significantly as intensity to the information content of images. Van Essen et al. noted that "visual acuity is many times worse for patterns differing only in spectral composition than for patterns differing in luminance."¹²

Cost function 2

Cost function 1 can be susceptible to problems if neighboring pixels have different hues but identical



11 Results of various reconstructions for the “glass” image at five-power magnification. (a) Original resolution, (b) bilinear reconstruction, (c) bicubic reconstruction, (d) Dyn’s difference-from-plane cost function, (e) Lawson’s optimization, and (f) our method.

intensities. Although such configurations are relatively rare, and the associated problems are less noticeable than if similar problems occurred in the intensity, they may still occur.

A more robust solution is to compute three costs— C_R , C_G , and C_B —for each edge (one for each of the RGB color components). To compute those costs, we treat each color component as an independent gray-scale image. The cost for the edge is

$$C = 0.21267C_R + 0.71516C_G + 0.07217C_B \quad (5)$$

We can easily contrive examples for which cost function 2 yields better results than cost function 1. However, cost function 2 takes more time to compute, and in the test images we examined, the difference in image quality is imperceptible.

Examples and discussion

Figures 10 and 11 show additional examples of DDT reconstruction. In Figure 10, as with most of the cases we tested, DDT produces more pleasing reconstructions than bicubic interpolation.

Figure 11 compares the various reconstruction methods discussed here. Bilinear (Figure 11b) and bicubic reconstruction (Figure 11c) show the reconstruction artifacts common to tensor-product reconstruction functions. Using DDT with Dyn’s difference-from-plane (DP) cost function (Figure 11d)—which in our experience is the previously published DDT cost function most suited to images—shows marked improvement over bilinear and bicubic interpolation, but it also has noticeable flaws. Our cost function, used with Lawson’s optimization algorithm (Figure 11e) improves the result over DP, but has small errors that appear as nicks in the stem of the glass. Our look-ahead optimization algorithm (Figure 11f) corrects these errors and produces a convincing reconstruction.

We produced each of these images in about 5 seconds, most of which we spent in computing the triangulations. Once we compute the triangulations, scaling the image simply occurs by interpolating triangles linearly from their vertices. If hardware is available for Gouraud shading, this can be accomplished at 30 frames per second or better. (See Figure 3 where we rendered a texture map in this manner.)

Of course, DDT reconstruction has limitations. In particular, single-pixel features don’t have enough information for meaningful triangulation. Extremely small

features or single-pixel thick lines aren’t reconstructed as well as larger features or thicker lines. Still, in all but extremely low-resolution images, DDT reconstruction performs well.

Future work

It’s possible to replace the triangles in a DDT with triangular surface patches. Quak and Schumaker¹³ showed how DDT can be used to create a C^1 surface. The basic idea is to replace each triangle with three cubic triangular patches using the Clough–Tocher split. Our image reconstruction algorithm could accommodate such a surface and would allow the interpolating surface to match gradients at pixel centers.

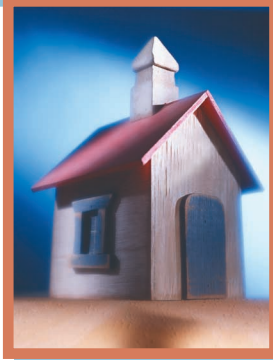
We’ve applied DDT to reconstruction of uniformly sampled images, but the method should extend equally well to nonuniformly sampled data. Indeed, we believe that triangulation of nonuniformly acquired samples followed by postfiltering might be an effective way to reconstruct images from these samples. ■

Acknowledgments

This work was funded in part by Adobe Systems.

References

1. A. Glassner, *Principles of Digital Image Synthesis*, Morgan Kaufmann, San Francisco, 1995.
2. D.P. Mitchell and A.N. Netravali, “Reconstruction Filters in Computer Graphics,” *Computer Graphics* (Proc. Siggraph 88), vol. 22, no. 4, ACM Press, New York, 1988, pp. 221-228.
3. G. Wolberg, *Digital Image Warping*, IEEE CS Press, Los Alamitos, Calif., 1990.
4. N. Dyn, D. Levin, and S. Rippa, “Data Dependent Triangulations for Piecewise Linear Interpolation,” *IMA J. Numerical Analysis*, vol. 10, 1990, pp. 137-154.
5. L. Rila and A.G. Constantinides, “Image Coding Using Data-Dependent Triangulation,” *Proc. 1997 13th Int’l Conf. Digital Signal Processing, Part 2*, IEEE Press, Piscataway, N.J., 1997, pp. 531-534.
6. J. Allebach and P.W. Wong, “Edge-Directed Interpolation,” *IEEE Int’l Conf Image Processing (ICIP 97)*, IEEE Press, Piscataway, N.J., 1996, pp. 707-710.
7. B.S. Morse and D. Schwartzwald, “Isophote-Based Interpolation,” *Int’l Conf. Image Processing (ICIP 98)*, IEEE Press, Piscataway, N.J., 1998.
8. J. Brown, “Vertex Based Data Dependent Triangulations,”



SCHOLARSHIP MONEY FOR STUDENT LEADERS

Student members active in IEEE Computer Society chapters are eligible for the Richard E. Merwin Student Scholarship.

Up to four \$3,000 scholarships are available.
Application deadline: 31 May



Investing in Students

computer.org/students/

Computer Aided Geometric Design, vol. 8, no. 3, 1991, pp. 239-251.

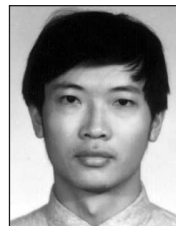
9. B. Choi et al., "Triangulation of Scattered Data in 3D-Space," *Computer Aided Design*, vol. 20, no. 5, 1988, pp. 239-248.

10. C.L. Lawson, "Software for c^1 Surface Interpolations," *Mathematical Software III*, J.R. Rice, ed., Academic Press, New York, July 1977, pp. 161-194.

11. L. Schumaker, "Computing Optimal Triangulations using Simulated Annealing," *Computer Aided Geometric Design*, vol. 10, nos. 3-4, 1993, pp. 329-345.

12. D.C. Van Essen et al., "Information Processing in the Primate Visual System: An Integrated Systems Perspective," *Science*, vol. 255, no. 5043, 1992, pp. 419-423.

13. E. Quak and L. Schumaker, "Cubic Spline Fitting Using Data Dependent Triangulations," *Computer Aided Geometric Design*, vol. 7, no. 1-4, 1990, pp. 293-302.



Xiaohua Yu is a senior software engineer at Intel. He completed a BS from the Modern Applied Physics Department of Tsinghua University in Beijing, China in 1991. In 1999 he earned both an MS in computer science and a PhD in physics from

Brigham Young University. His current research interests include network communication and industrial automation.

RENEW your Computer Society membership for

Regions 1-7
Renewal deadline
has passed.

- ✓ 12 issues of **Computer**
- ✓ Member discounts on periodicals, conferences, books, proceedings
- ✓ Free membership in Technical Committees
- ✓ Digital library access at the lowest prices
- ✓ Free e-mail alias @computer.org



<http://www.ieee.org/renewal>



Bryan S. Morse is an associate professor of computer science at Brigham Young University, where he conducts research involving multiresolution image analysis and applications of parallel processing to image analysis. He holds a PhD in

computer science from the University of North Carolina at Chapel Hill.



Thomas W. Sederberg is a professor of computer science at Brigham Young University. His research interests include computer graphics, computer-aided geometric design, computer algebra, and algorithms for genealogy visualization.

He has a PhD in mechanical engineering from Purdue University. He serves on the editorial boards of *Computer Aided Geometric Design* and *Computer-Aided Design*.

Readers may contact Sederberg at the Dept. of Computer Science, Brigham Young Univ., Provo, UT 84602, email tom@byu.edu.