Contour stencils for edge-adaptive image interpolation

Pascal Getreuer

University of California Los Angeles, Mathematics Department, U.S.A.

ABSTRACT

We first develop a simple method for detecting the local orientation of image contours and then use this detection to design an edge-adaptive image interpolation strategy. The detection is based on total variation: small total variation along a candidate curve implies that the image is approximately constant along that curve, which suggests it is a good approximation to the contours. The proposed strategy is to measure the total variation over a "contour stencil," a set of parallel curves localized over a small patch in the image. This contour stencil detection is used to design an edge-adaptive image interpolation strategy. The interpolation is computationally efficient, operates robustly over a variety of image features, and performs competitively in a comparison against existing methods. The method extends readily to vector-valued data and is demonstrated for color image interpolation. Other applications of contour stencils are also discussed.

Keywords: Interpolation, contour stencils, total variation

1. INTRODUCTION

Digital multimedia frequently involves rescaling an image to another resolution. This task, called *image zooming* (also known as *image scaling* and *single-frame super-resolution*), arises for example when viewing a digital image and asking to "zoom in" for a closer look, or when rescaling a video to fullscreen view. *Image interpolation* refers to the special case when the process is reversible by subsampling.

1.1 Recent Work

As an overview of recent image zooming methods, we briefly describe three prominent categories.

1.1.1 Identifying similar image patches

Several high-quality methods involve identifying similar image patches.

Freeman et al.[1] and similarly other "example-based" methods[2,3] performed image zooming using a large database of image patches of low resolution and high resolution pairs. For each patch in a given low resolution image, they search the database for the corresponding high resolution patch. Jiji et al.[4] performed a similar process in the contourlet domain, where they identify similar patches of contourlet coefficients.

The idea in fractal image zooming[5,6] is to identify self-similar structures in an image to extrapolate the detail to finer scales. The image is first encoded as an iterated function system (IFS), a contraction that maps patches to similar patches such that the image is the fixed point of the IFS. Scalings of the original image are then obtained by fixed point iteration of the IFS at the desired resolution.

Several methods[7–9] have been inspired by the nonlocal means method[10] for image noise reduction. These methods use the nonlocal means weights to identify similar patches in the given image and combine them to estimate a high resolution patch.

In all of these methods, expanding the search for similar patches usually improves the results. Thus they compromise between speed and quality—or accept extreme run times.

1.1.2 PDE-based

Total variation (TV) based regularization, introduced by Rudin, Osher, and Fatemi,[11] has found success in image zooming as applied by Malgouyres and Guichard[12] and Chan and Shen.[13,14]

Visual Communications and Image Processing 2009, edited by Majid Rabbani, Robert L. Stevenson, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7257, 725718 · © 2009 SPIE-IS&T · CCC code: 0277-786X/08/$18 · doi: 10.1117/12.806014

The total variation of an image is $\|u\|_{\mathrm{TV}} := \int |\nabla u|$, where the gradient is interpreted in a distributional sense. Malgouyres posed image zooming as an optimization problem for the minimum-TV image u that agrees with the low-resolution v data,

$$\hat{u} = \arg\min_{u \in BV} \|u\|_{\mathrm{TV}} \quad \text{subject to } \mathrm{sample}(h * u) = v.$$

The minimization may be solved by evolving a PDE on the image. Chan and Shen considered a similar optimization problem for image inpainting, where image zooming is a special case.

Another TV-inspired PDE was applied by Luong et al.[15] They begin by zooming the image with a linear method, then attempt to remove the artifacts. As one step in this postprocessing, the TV-flow PDE is used to smooth out jagged edges.

1.1.3 Edge-directed

Edge-directed methods[16–20] try to interpolate along the direction of edges rather than across them, producing interpolations with sharp edges. In contrast to the PDE- and patch-based methods, edge-directed methods are often designed from the outset for computational efficiency.

The difficulty in edge directed methods is in how they determine the direction of the edge, and this is primarily where they differ. Stepin[20] compares a pixel with each of its eight neighbors and decides by a threshold whether each neighbor is alike or different. For each of the 256 possible outcomes, the method assigns a tailored linear formula to interpolate the neighborhood.

1.2 Outline

This work introduces *contour stencil interpolation*, a fast, edge-adaptive method for image interpolation.

- Section 2 defines the problem (§2.1) and a design objective in terms of the image contours (§2.2).
- Section 3 presents the contour stencil interpolation method: *contour stencils* are proposed (§3.1) as a method for detecting contour orientations, then applied in developing the interpolation method (§3.2). Section 3.3 discusses efficient implementation.
- Section 4 shows examples and a comparison against several state-of-the-art methods.
- Section 5 applies contour stencils in an image enhancement method.

2. FORMULATION

Like any design task, image scaling benefits from a mathematical statement of the problem and identifying suitable objectives. Here we shall clarify a mission statement of our goals in designing an image scaling method.

2.1 Problem Model

The image interpolation problem is to solve

$$v = \mathrm{sample}(u) \tag{1}$$

for u, where v is the given image. Alternatively, sharper results are possible by solving the deconvolution problem

$$v = \mathrm{sample}(h * u) \tag{2}$$

where h is the point spread function (PSF) of the imaging device. The choice of h can be problematic, however, since overestimating the support of the PSF results in overshoot artifacts and for most applications the exact PSF is unknown and spatially varying.

Moreover, deconvolution is generally more demanding—both computationally and conceptually—than interpolation. For this reason, we develop the proposed method as an interpolation with hopes of simplicity and efficiency.

2.2 Objectives

This work shall consider the following objectives.

Objective 1) *Smoothness.* In smooth regions, scaling should preserve low-order polynomials.

Objective 2) *Step response.* Edges should scale without excessive blurring or overshoots.

Objective 3) *Directional selectivity.* Edges of different angles should scale without becoming jagged.

Additionally, of course, any other artifacts should be visually tolerable, and the method must be computationally fast enough for the application.

The challenge with multiple objectives is how they should balance. For linear methods, Objective 1 is in opposition with Objective 2. We simplify these objectives into the single objective that contours in the scaled image should be "consistent" with those in the given image.

Objective) *Contour consistency.* The local orientations of the contours should match and no new contour lines should be created.

Such an objective has been applied fruitfully in previous work.[15,17] Contour consistency implies that edges are scaled without overshoots or jagged edge artifacts (Objectives 2 and 3). It also implies that constants are preserved, which is just enough to satisfy Objective 1 for constant order smoothness.

3. CONTOUR STENCIL INTERPOLATION

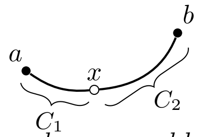
This section describes the contour stencil interpolation method. For a sufficiently regular image u, denote by $\mathrm{TV}(C)$ its total variation along a smooth curve C,

$$\mathrm{TV}(C) := \int_0^T \left|\tfrac{\partial}{\partial t} u\big(\gamma(t)\big)\right| \, dt \tag{3}$$

where $\gamma(t)$, $0 \le t \le T$, parameterizes the curve C. The heart of our approach is the following theorem.

Theorem 3.1. *Consider the approximation*

$$u(x) \approx \hat{u}(x) = (1-\lambda)u(a) + \lambda u(b)$$



with $0 \le \lambda \le 1$ *and let* $C = C_1 \cup C_2$ *be a smooth curve passing through* a, x, *and* b. *If* u *is differentiable along* C, *then the approximation error is bounded by*

$$|\hat{u}(x) - u(x)| \le \max\big\{|1-\lambda|\,,|\lambda|\big\}\,\mathrm{TV}(C)\,.$$

Proof. We have

$$|u(a) - u(x)| = \left|\int_{C_1} \tfrac{\partial}{\partial t} u\big(\gamma(t)\big)\, dt\right| \le \mathrm{TV}(C_1)\,,$$

and similarly $|u(b) - u(x)| \le \mathrm{TV}(C_2)$. Combining these bounds yields the result. ☐

An obvious corollary is that if $\mathrm{TV}(C) = 0$, then u is constant along C and the error is zero; interpolation along contours is exact. This fact is the motivation for edge-directed methods. However, the exact image contours are unknown and can only be estimated. The value of the theorem is that it quantifies the suitability of an estimated contour by its total variation. If $\mathrm{TV}(C)$ is small, interpolation error is small.

The proposed approach to interpolation comes directly from Theorem 3.1: minimize $\mathrm{TV}(C)$. If $\mathrm{TV}(C)$ is the smallest among a set of candidate curves, then the interpolation error has the smallest bound when interpolating along C. To apply these ideas, we must translate curves and differentials to the pixels; this is where the "contour stencils" will come in.

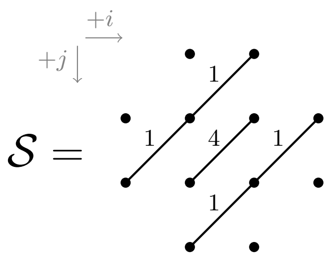
3.1 Contour Stencils

Let v be the given image where $v_{i,j}$ denotes the value of the (i,j)th pixel. Define the (i,j)th cell as the square whose corners correspond to $v_{i,j}$, $v_{i+1,j}$, $v_{i,j+1}$, $v_{i+1,j+1}$. For each cell, we want to find a curve $C_{i,j}$ passing through the cell such that $\mathrm{TV}(C_{i,j})$ is small.

On the given image, a curve may be represented as a sequence of pixels $\alpha_0, \ldots, \alpha_N$. The total variation may then be discretized as

$$\sum_{n=1}^{N} |v(\alpha_n) - v(\alpha_{n-1})| \approx \mathrm{TV}(C). \tag{4}$$

For sufficiently smooth u, nearby parallel curves have approximately the same TV. Observing this property, the TV estimate (4) may be improved by averaging over several parallel curves.



$$\begin{aligned}\mathrm{TV}_{i,j}(\mathcal{S}) = \tfrac{1}{|\mathcal{S}|}\big(&|v_{i,j} - v_{i-1,j+1}| + |v_{i,j} - v_{i+1,j-1}| \\ &+ 4\,|v_{i,j+1} - v_{i+1,j}| \\ &+ |v_{i+1,j+1} - v_{i,j+2}| + |v_{i+1,j+1} - v_{i+2,j}|\big) \\ |\mathcal{S}| = {}& (1+1+4+1+1)\sqrt{2}\end{aligned}$$

Figure 1. An example contour stencil $\mathcal{S}$ and its total variation computed at cell (i,j).

A *contour stencil* $\mathcal{S}$ is a set of edges between pixels. We choose the edges such that they approximate several parallel curves localized over a small patch of the image (see Figure 1). Define the total variation of $\mathcal{S}$ at cell (i,j) as

$$\mathrm{TV}_{i,j}(\mathcal{S}) := \tfrac{1}{|\mathcal{S}|} \sum_{\mathit{edges}} w_{\alpha,\beta}\,|v_\alpha - v_\beta|, \tag{5}$$

where the $w_{\alpha,\beta}$ denote the edge weights and $|\mathcal{S}|$ sums the edge lengths, $|\mathcal{S}| := \sum w_{\alpha,\beta}\,|x_\alpha - x_\beta|$.

Now we can find an estimate of the contours by comparing the total variations of different candidate stencils. The proposed stencil set Σ is shown in Figure 2. The best-fitting stencil is the one with the minimum TV,

$$\mathcal{S}^*_{i,j} = \arg\min_{\mathcal{S}\in\Sigma} \mathrm{TV}_{i,j}(\mathcal{S}). \tag{6}$$

This stencil serves as a model of the underlying image contours. Since $\mathcal{S}^*_{i,j}$ is the stencil with the lowest estimated total variation, the estimated error bound is minimized when interpolating along its modeled contours:

$$|\hat{u}(x) - u(x)| \le \mathrm{TV}(C_{i,j}) \approx \mathrm{TV}_{i,j}(\mathcal{S}^*_{i,j}) = \min_{\mathcal{S}\in\Sigma} \mathrm{TV}_{i,j}(\mathcal{S}).$$

Nevertheless, the interpolation error may yet be large since this is an *estimated* error bound.

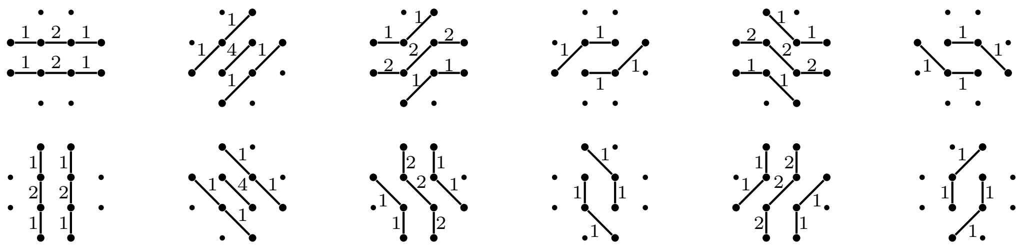


Figure 2. The 12 contour stencils used in the proposed method. Together they distinguish 8 directions.

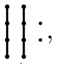
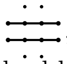
Remark 1. *If the given image is noisy, spatially filtering* $\mathrm{TV}_{i,j}(\mathcal{S})$ *can improve the accuracy of the contour detection. For example, let* $\mathrm{TV}^\rho_{i,j}(\mathcal{S}) = \big(G_\rho * \mathrm{TV}(\mathcal{S})\big)_{i,j}$*, where* G_ρ *is a Gaussian with standard deviation* ρ*. Even for clean images, light filtering can improve the results.*

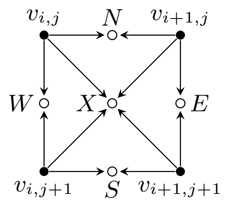
Remark 2. *The weights on the stencil edges have only a small effect on the detection. Results are similar if, for example, all edges have the same weight. The intention is to improve spatial accuracy by giving more significance to edges in the center of the stencils. The choice of weights affects the results most where there is a near tie in* (6) *for the minimum* $\mathrm{TV}(\mathcal{S})$*, that is, the weights can help where the detection gets confused.*

Remark 3. *Generalizing beyond scalar-valued images, the absolute value in* (5) *may be replaced by a vector norm or a metric. On color images, for example, a choice is* $\|\vec{u} - \vec{w}\| = |u_{red} - w_{red}| + |u_{green} - w_{green}| + |u_{blue} - w_{blue}|$ *or a vector norm in some other color space.*

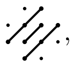
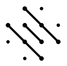
3.2 Interpolation

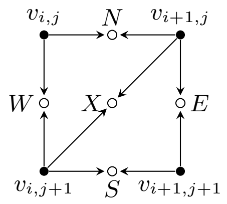
We now apply the contour stencil detection (§3.1) to interpolation. The following formulas give a simple way to perform factor 2 interpolation for the stencil set shown in Figure 2, which we call *contour stencil interpolation.* In the diagrams, filled circles • are pixels in the given image v; open circles ∘ are interpolated pixels, which are computed as the average of all incoming arrows.

For $\mathcal{S}^*_{i,j} =$ , , and the isotropic fallback, do standard bilinear interpolation:

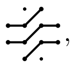
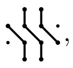
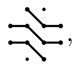
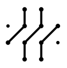


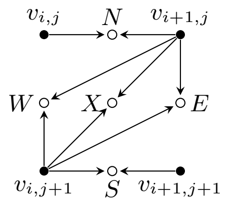
$$N = \tfrac{1}{2}(v_{i,j} + v_{i+1,j})$$
$$S = \tfrac{1}{2}(v_{i,j+1} + v_{i+1,j+1})$$
$$W = \tfrac{1}{2}(v_{i,j} + v_{i,j+1})$$
$$E = \tfrac{1}{2}(v_{i+1,j} + v_{i+1,j+1})$$
$$X = \tfrac{1}{4}(v_{i,j} + v_{i+1,j} + v_{i,j+1} + v_{i+1,j+1})$$

For $\mathcal{S}^*_{i,j} =$ , similarly for 

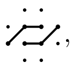
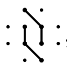
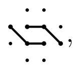
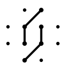


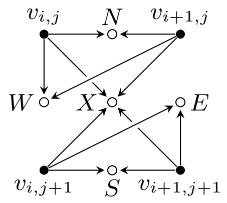
$$N = \tfrac{1}{2}(v_{i,j} + v_{i+1,j})$$
$$S = \tfrac{1}{2}(v_{i,j+1} + v_{i+1,j+1})$$
$$W = \tfrac{1}{2}(v_{i,j} + v_{i,j+1})$$
$$E = \tfrac{1}{2}(v_{i+1,j} + v_{i+1,j+1})$$
$$X = \tfrac{1}{2}(v_{i,j+1} + v_{i+1,j})$$

For $\mathcal{S}^*_{i,j} =$ , similarly for , , 



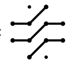
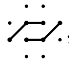
$$N = \tfrac{1}{2}(v_{i,j} + v_{i+1,j})$$
$$S = \tfrac{1}{2}(v_{i,j+1} + v_{i+1,j+1})$$
$$X = W = E = \tfrac{1}{2}(v_{i,j+1} + v_{i+1,j})$$

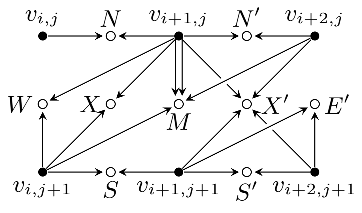
For $\mathcal{S}^*_{i,j} =$ , similarly for , , 



$$N = W = \tfrac{1}{2}(v_{i,j} + v_{i+1,j})$$
$$S = E = \tfrac{1}{2}(v_{i,j+1} + v_{i+1,j+1})$$
$$X = \tfrac{1}{4}(v_{i,j} + v_{i+1,j} + v_{i,j+1} + v_{i+1,j+1})$$

In the typical case that an edge midpoint is shared between adjacent cells, its interpolated value is the average of the two assigned values.

For example, if $\mathcal{S}^*_{i,j} =$  and $\mathcal{S}^*_{i+1,j} =$ , then



$$E = \tfrac{1}{2}(v_{i,j+1} + v_{i+1,j})$$
$$W' = \tfrac{1}{2}(v_{i+1,j} + v_{i+2,j})$$
$$M = \tfrac{1}{2}(E + W')$$

Again, these interpolation formulas are just a simple implementation of the guiding principle that values should be interpolated along the contours. The same principle may be applied to derive interpolation formulas for other stencils and interpolation factors other than 2.

To summarize, contour stencil interpolation is done by performing the following steps for each cell of the input image: compute the TV estimates (5), determine the best-fitting stencil $\mathcal{S}^*_{i,j}$, compute the interpolated pixels, and average shared midpoint values. The next section discusses the implementation in more detail.

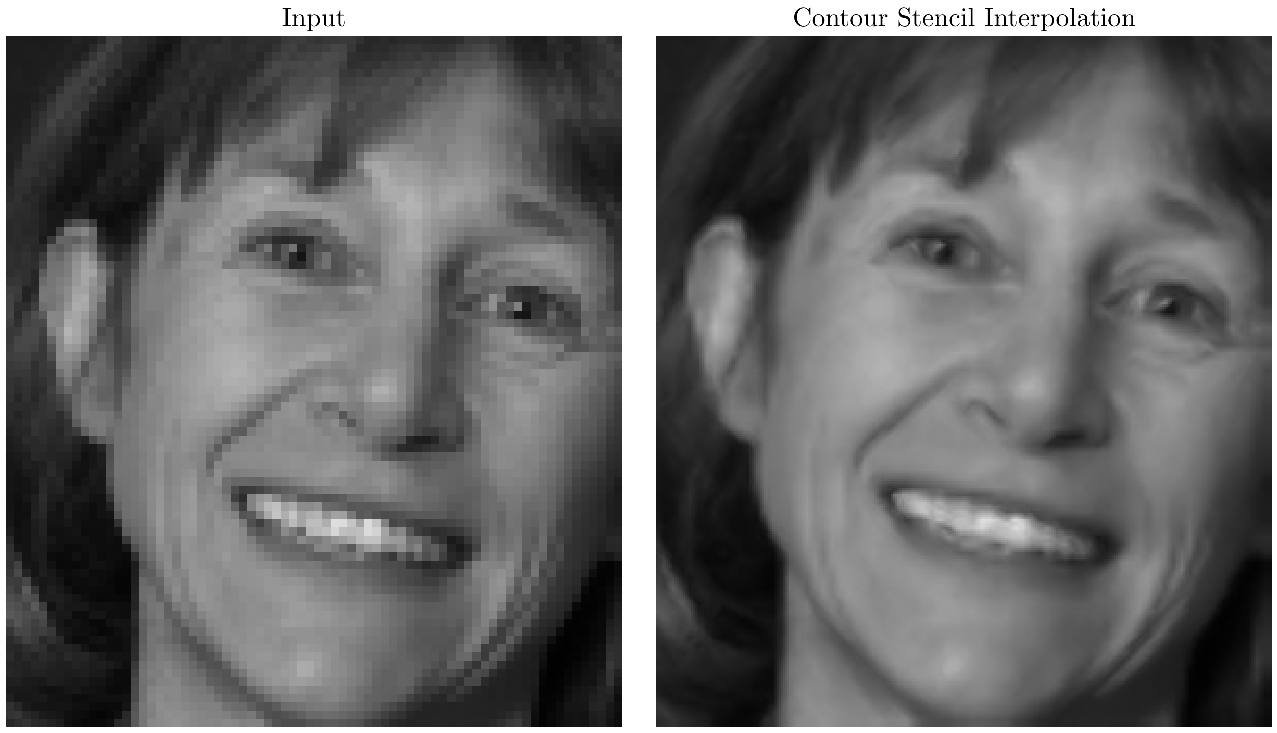


Figure 3. The proposed factor 2 interpolation strategy has been applied twice for a factor 4 increase in resolution. Notice the effective interpolation of fine features like the hair, eyes, and teeth. Such quality on one-pixel wide features is possible thanks to the contour stencil detection strategy.

The proposed method resembles the one dimensional Essentially Non-Oscillatory (ENO) interpolation technique.[21] In ENO, a piecewise polynomial interpolant is constructed in such a way as to avoid oscillations near singularities. For each interval, a set of possible interpolation stencils are considered, and the stencil with the smallest divided difference is selected for interpolation. The total variation estimates $\mathrm{TV}(\mathcal{S})$ in the proposed method have the same role: testing how well the stencils fit the data. In both methods, interpolation then proceeds using the best stencil as a model for the underlying function.

Remark 4. *A factor 2 scaling method may be extended to other scale factors by "pyramid magnification."*[18] *Apply the factor 2 method k times for a factor 2^k resolution increase. For a nondyadic factor R, apply the method $k = \log_2 \lceil R \rceil$ times and then reduce to the desired resolution with bilinear scaling.*

3.3 Fast Implementation

This section discusses fast implementation of contour stencil interpolation. In the following, we make a couple sacrifices in quality to improve speed: we forgo stencil denoising by setting $\rho = 0$ and implement all computations in integer arithmetic. (Similar implementation is still possible without these sacrifices.) On a modern computer*, the implementation presented here performs the interpolation at about 70ns per output pixel.

*Computations were timed on a 2.40GHz Intel® Core™ 2 Duo T7700 with 2GB RAM.

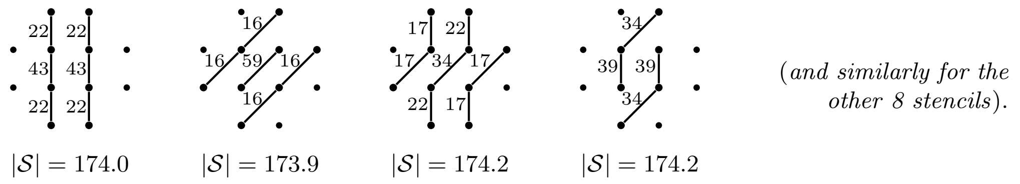
The bottleneck computations are the differences $|v_\alpha - v_\beta|$ appearing in the TV computations (5). Define the differences

$$\begin{array}{ll} D^H_{m,n} = |v_{m,n} - v_{m+1,n}| & m = i-1,\ i,\ i+1 \\ D^V_{m,n} = |v_{m,n} - v_{m,n+1}| & n = j-1,\ j,\ j+1 \\ D^A_{m,n} = |v_{m,n} - v_{m+1,n+1}| & \\ D^B_{m,n} = |v_{m,n+1} - v_{m+1,n}| & \end{array}.$$

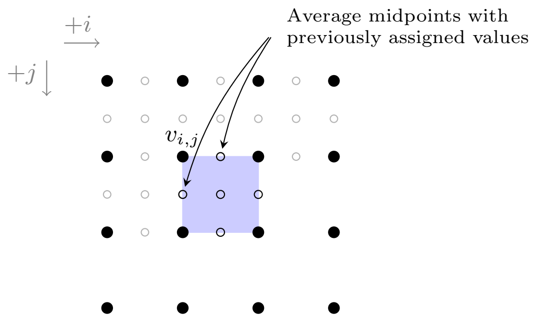
The differences are computed over a window $m = i-1, i, i+1$, $n = j-1, j, j+1$. (For a color image, substitute $|\cdot|$ with a vector norm as in Remark 3.) These differences can be combined to compute the TV estimates,

$$\mathrm{TV}_{i,j}(\mathcal{S}) := \tfrac{1}{|\mathcal{S}|} \sum_{edges} w_{\alpha,\beta} \, |v_\alpha - v_\beta| \, .$$

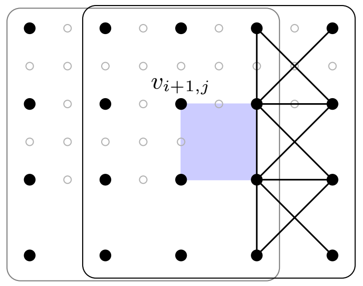
To eliminate the division by $|\mathcal{S}|$, we can use a set of integer edge weights such that $|\mathcal{S}|$ is approximately the same for every stencil—then the division is unnecessary. One possible choice is



```
Algorithm. Contour stencil interpolation by factor 2
Loop j,
    0 → i
    Compute the differences in the window
    Loop i,
        Use the differences to compute TV_{i,j}(S) for each stencil
        Find the minimum-TV stencil S*_{i,j}
        Interpolate cell (i, j) according to S*_{i,j} (§3.2)
        Average the edge midpoints with previously assigned values
        Compute the new differences on the right edge of the window
        i + 1 → i
    j + 1 → j
```



Interpolating cell (i, j)



Computing new differences

4. EXAMPLES

Contour stencil interpolation is demonstrated in Figures 3 and 4. The interpolation can partially recover oriented textures like hair. In Figure 5, contour stencil interpolation is applied on a color image, by replacing the absolute value with the ℓ^1 norm in the $\mathrm{YC_bC_r}$ color space.

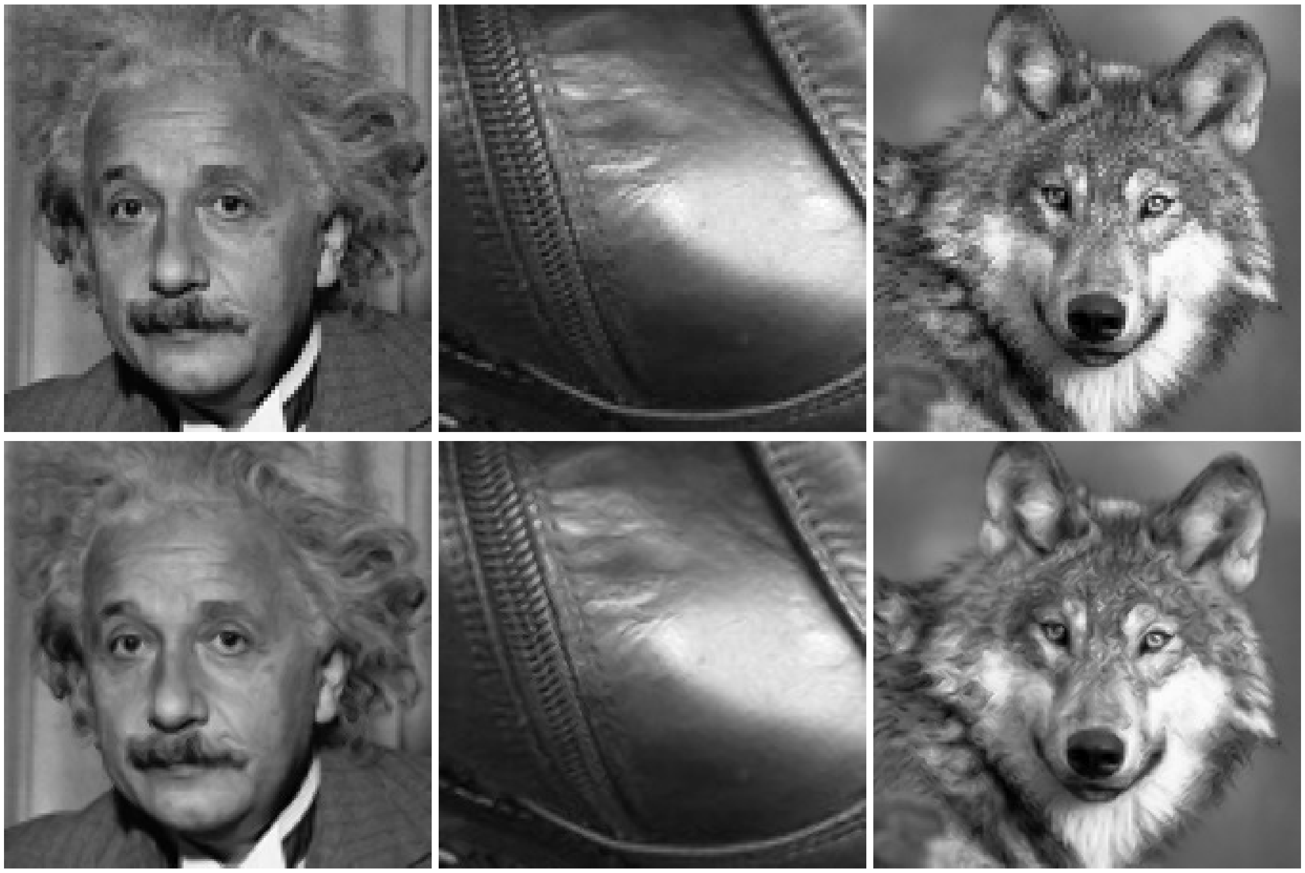


Figure 4. *First row:* input. *Second row:* contour stencil interpolation by factor 2. *Wolf photograph by Gary Kramer, U.S. Fish and Wildlife Service.*

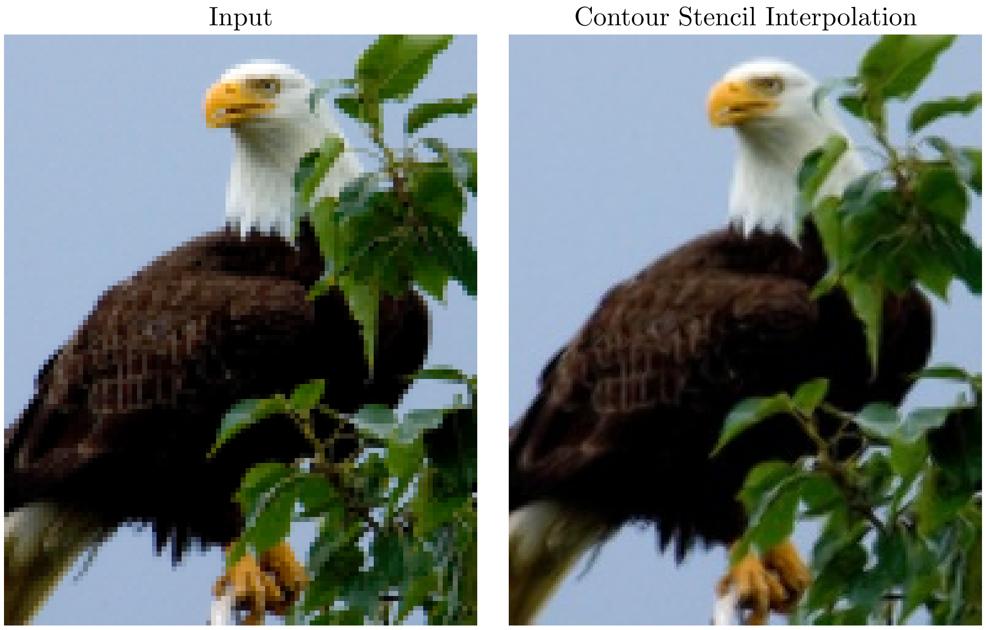


Figure 5. Color contour stencil interpolation, by replacing absolute value with a vector norm (see Remark 3). *Photograph by Steve Hillebrand, U.S. Fish and Wildlife Service.*

Figure 6 shows an experiment in which a high-resolution image is reduced by factor 2 and then approximately recovered by bicubic interpolation with PSNR 29.7 and contour stencil interpolation with PSNR 29.3. The result with contour stencil interpolation appears sharper and edges are less jagged than with bicubic interpolation, yet the PSNR values suggest that the bicubic result is better. This example illustrates that larger PSNR value (or SNR value) does not necessarily imply better visual quality.[22]

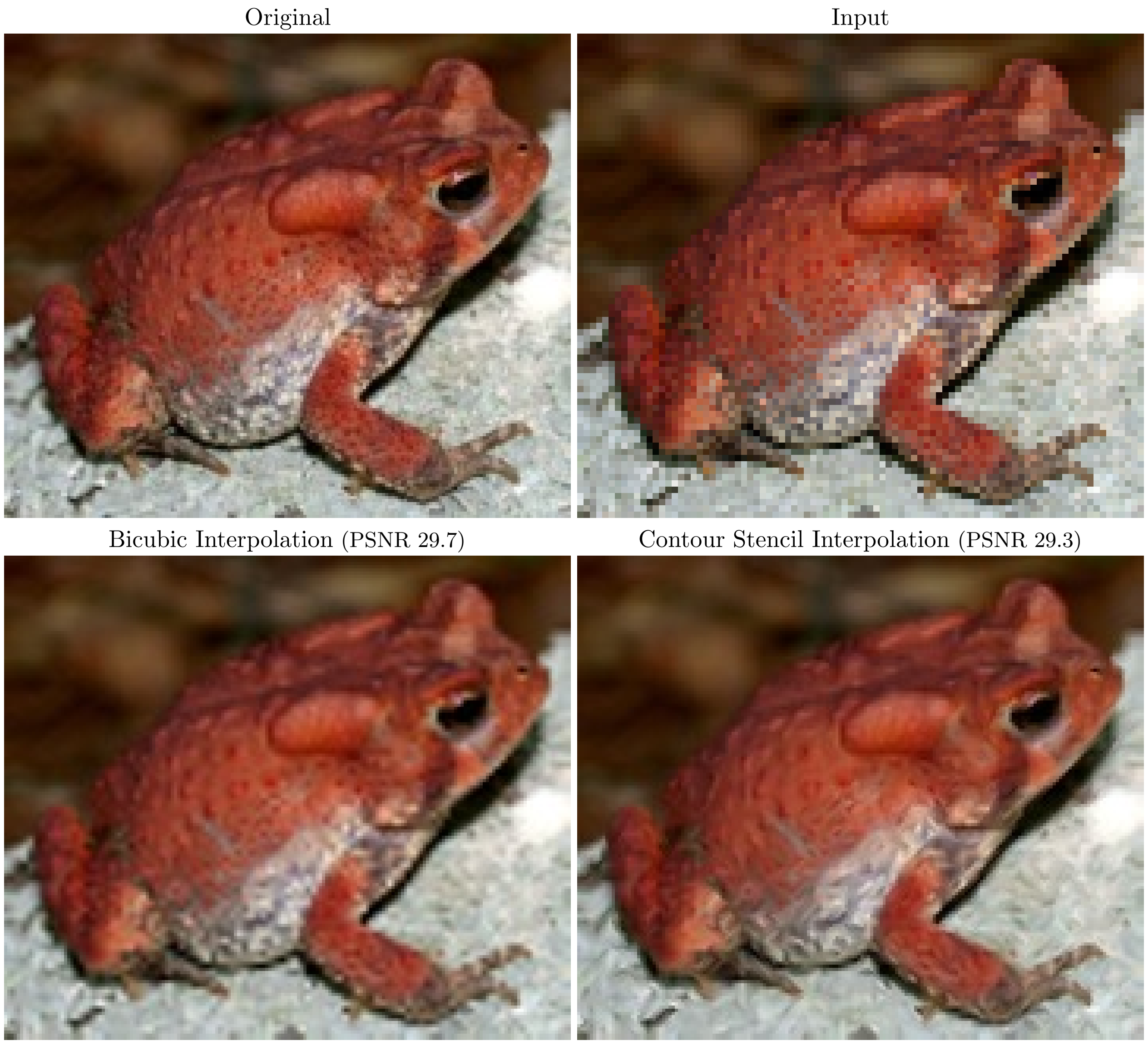

Original
Input
Bicubic Interpolation (PSNR 29.7)
Contour Stencil Interpolation (PSNR 29.3)

Figure 6. Interpolation experiment on a color image. The original image (top left) is reduced by factor 2 to create the input image (top right). The input image is interpolated to recover an image with the same resolution as the original. (Same experiment with bilinear interpolation: PSNR 29.2.) *Photograph by John D. Willson, Amphibian Research and Monitoring Initiative.*

Figure 7 compares the proposed method (with $\rho = 0.5$) alongside several existing edge-adaptive methods: the TV-based method of Malgouyres[12] (where h is the lowpass analysis filter of the CDF 9/7 wavelet), an implementation of IFS image scaling,[23] and the edge-directed method AQua-2 of Muresan.[16] Again, since the results are quite similar, PSNR comparison would be misleading. The methods should be evaluated by how well they achieve the design objectives (§2.2).

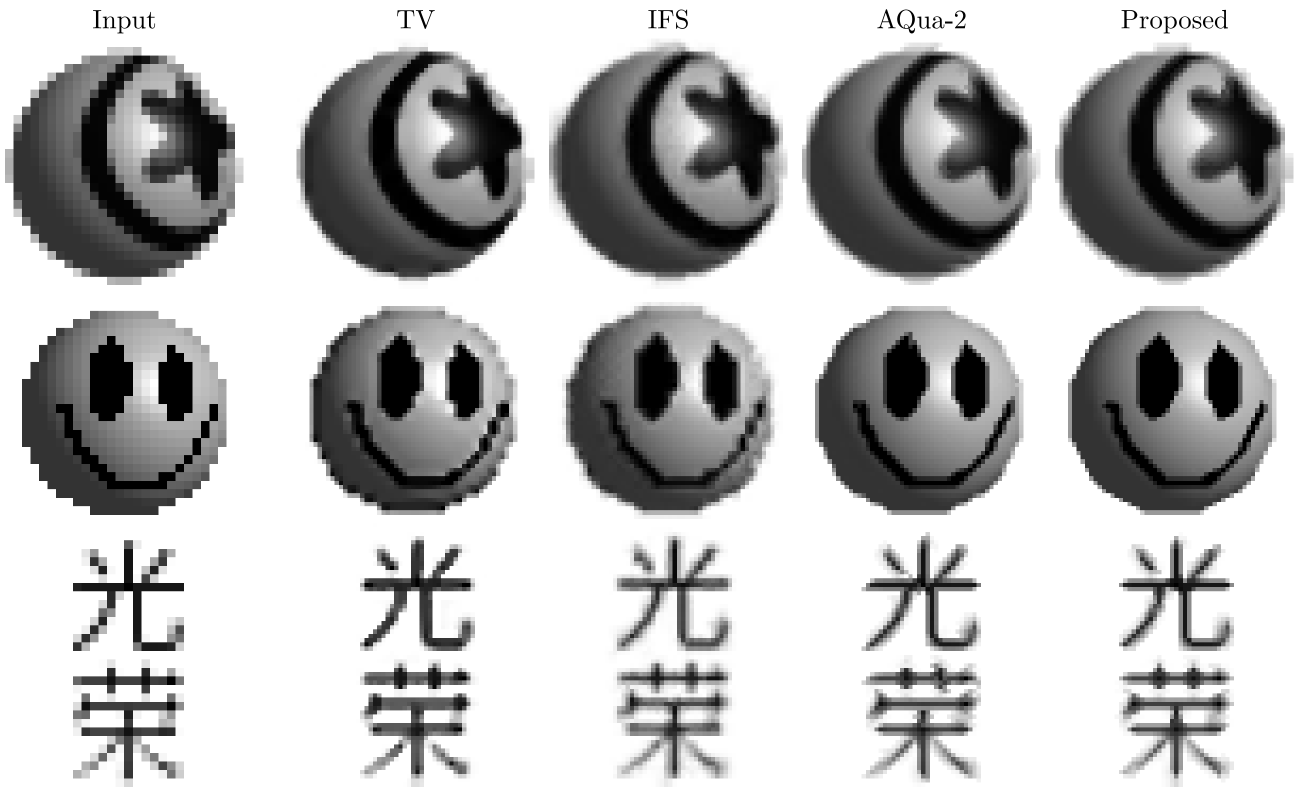


Figure 7. The proposed method is compared against existing image scaling methods. The test image was constructed to include the following atomic image elements: antialiased and non-antialiased steps and lines, smooth regions, and lines endings and intersections.

5. ENHANCEMENT WITH CONTOUR STENCILS

In the previous section, contour stencils were used for edge-adaptive interpolation. This section shows that contour stencils are also useful in other image processing tasks by developing a contour stencils based method for image enhancement, that is, simultaneous sharpening and denoising.

A simple method for sharpening a one-dimensional signal is to push the signal up where $u_{xx} < 0$ and down where $u_{xx} > 0$, that is,

$$u_t = -\operatorname{sign}(u_{xx})\,|u_x|\,.$$

Another point of view is that this PDE is the transport equation $u_t = -cu_x$ pushing the signal left and right with coefficient $c(t,x) = \operatorname{sign}(u_{xx})\operatorname{sign}(u_x)$. Discretizing according to the characteristics,

$$u^{n+1}(x) = u^n\big(x - h\operatorname{sign}(u^n_{xx})\operatorname{sign}(u^n_x)\big),$$

where h should be on the order of the grid spacing. Extending to two dimensions, we obtain a simple method for image sharpening,

$$u^{n+1}(x) = u^n\big(x - h\operatorname{sign}(u^n_{\eta\eta})\tfrac{\nabla u^n}{|\nabla u^n|}\big), \tag{7}$$

where $u_{\eta\eta} = \frac{\nabla u}{|\nabla u|} \cdot \nabla^2 u \cdot \frac{\nabla u}{|\nabla u|}$ denotes the second derivative of u in the direction of ∇u.

The proposed enhancement method combines (7) with ROF image denoising. For a given a noisy image f and parameter $\lambda > 0$, the u that solves

$$0 = \nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right) + \lambda(f - u)$$

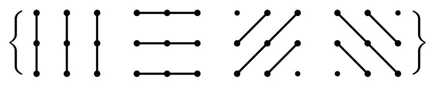
is the ROF denoised image. The first term encourages u to have low total variation while the second term encourages u to stay close to f. For simultaneous denoising and sharpening, the proposed method is to replace f as

$$0 = \nabla \cdot \left(\frac{\nabla u}{|\nabla u|}\right) + \lambda \left[u\big(x - h\,\mathrm{sign}(u_{\eta\eta})\tfrac{\nabla u}{|\nabla u|}\big) - u(x)\right]. \tag{8}$$

This modification does, of course, compromise many of ROF's properties. Uniqueness is lost, for example, since any constant function u satisfies (8). Instead, we seek an approximate solution nearby f. The idea is that if (8) is solved by an iterative method, then the second term incorporates the sharpening iteration from (7) while the first term provides smoothing as it does in the original ROF model.

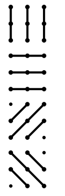
Usually, we think of sharpening and smoothing as opposing processes, but in (8) they have a fortuitous cooperation. The first term provides smoothing in the sense that it reduces the total variation, meaning it smooths out impulses and oscillations but does not change edges. The sharpening iteration sharpens the edges, but does not increase the size of the jumps, meaning it does not increase the total variation. So the smoothing and the sharpening do not undo one another's work.

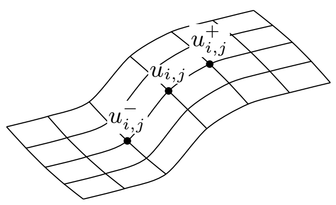
Equation (8) can be discretized with the help of contour stencils. For the purpose of enhancement, we use another set of stencils that are *node-oriented* (while the set in Figure 2 is cell-oriented).

$\Sigma =$


Let $\mathcal{S}^*_{i,j}$ denote the best-fitting stencil at (i,j). Depending on $\mathcal{S}^*_{i,j}$, define $u^-_{i,j}$ and $u^+_{i,j}$ by

$$u^\pm_{i,j} = \begin{cases} u_{i\pm1,j} & \text{if } \mathcal{S}^*_{i,j} = \\ u_{i,j\pm1} & \text{if } \mathcal{S}^*_{i,j} = \\ (u_{i\pm1,j} + u_{i,j\pm1})/2 & \text{if } \mathcal{S}^*_{i,j} = \\ (u_{i\mp1,j} + u_{i,j\pm1})/2 & \text{if } \mathcal{S}^*_{i,j} = \end{cases}$$





The values $u^-_{i,j}$, $u_{i,j}$, $u^+_{i,j}$ give a cross-section of the edge at (i,j), which enable the approximations

$$\frac{1}{|\nabla u|} \approx w_{i,j} = \frac{2h}{|u-u^-| + |u-u^+|}$$

$$u\big(x - h\,\mathrm{sign}(u_{\eta\eta})\tfrac{\nabla u}{|\nabla u|}\big) \approx \tilde{u}_{i,j} = \begin{cases} u^+_{i,j} & \text{if } u_{i,j} \text{ is close to } u^+_{i,j}, \\ u^-_{i,j} & \text{if } u_{i,j} \text{ is close to } u^-_{i,j}. \end{cases}$$

For determining between the two cases, let $\xi = \mathrm{sign}(|u-u^-| - |u-u^+|)$, then $\tilde{u}$ is found as $\tilde{u}_{i,j} = u^{\mathrm{sign}(\xi_{i,j})}_{i,j}$. Unfortunately, the edges obtained using this approximation for $u\big(x - h\,\mathrm{sign}(u_{\eta\eta})\tfrac{\nabla u}{|\nabla u|}\big)$ tend to be unnaturally sharp and jagged. Smoother edges are obtained using a soft sign, for example, let

$$\xi = \tfrac{2}{\pi}\arctan\big(\kappa(|u-u^-| - |u-u^+|)\big)$$

where parameter κ determines edge steepness, and let

$$\tilde{u} = (1 - |\xi|)u + |\xi|\,u^{\mathrm{sign}(\xi)}.$$

Let $\mathcal{N}_{i,j} = \big\{(i+1,j), (i-1,j), (i,j+1), (i,j-1)\big\}$ denote the neighborhood of pixel (i,j). Then, in imitation of the digital TV filter,[14] (8) may be discretized as

$$0 = \sum_{q\in\mathcal{N}_{i,j}} w^n_q(u^n_q - u^{n+1}_{i,j}) + \lambda(\tilde{u}^n_{i,j} - u^{n+1}_{i,j}),$$

yielding the iteration

$$u^0 = f, \quad u_{i,j}^{n+1} = \frac{\lambda \tilde{u}_{i,j}^n + \sum_{q \in \mathcal{N}_{i,j}} w_q^n u_q^n}{\lambda + \sum_{q \in \mathcal{N}_{i,j}} w_q^n}. \tag{9}$$

Like the digital TV filter, (9) converges quickly, and often produces satisfactory results with about 5 iterations.

Figure 8 tests the method's robustness with increasing levels of noise (additive white Gaussian noise with standard deviations 10, 20, and 40). In Figure 9, the method is extended to color images by replacing absolute value with a norm. The method is compared against the method of Horiuchi et al.[24]

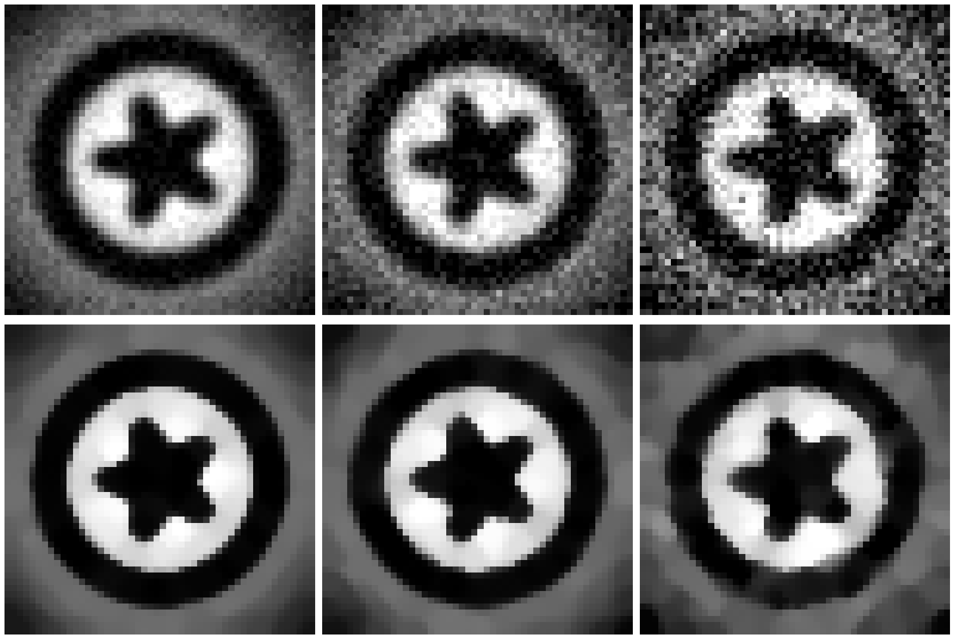


Figure 8. *Top row:* input with increasing levels of noise. *Bottom row:* enhanced (5 iterations, $\kappa = 3$, $\lambda = 20$).

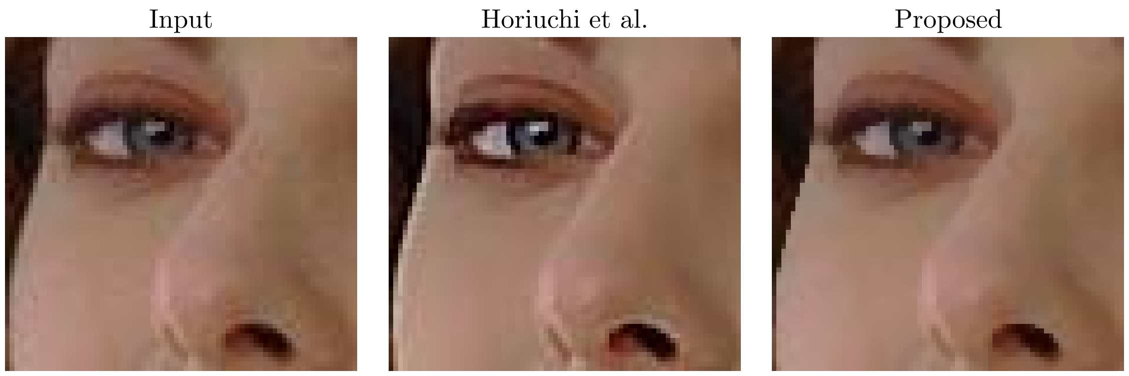


Figure 9. Comparison with Horiuchi et al.[24]

6. CONCLUSIONS

We have introduced contour stencils (§3.1), a simple method for detecting the local orientation of image contours. Using this tool, we developed an edge-adaptive interpolation method that is computationally efficient and competitive with the state-of-the-art. We also applied contour stencils in estimating the differential quantities in an image enhancement method (§5), demonstrating the potential of contour stencils in other low-level imaging tasks.

In this development, the choice of stencil set and the interpolation formulas were ad hoc, and other choices are left to be explored. For example, larger line stencils could be designed to distinguish between more than 8 directions for better directional selectivity. Stencils do not have to be lines, they could also model other features like corners and junctions. These possibilities will be considered in future research.

REFERENCES

[1] Freeman, W. T., Jones, T. R., and Pasztor, E. C., "Example-based super-resolution," *IEEE Computer Graphics and Applications* **22**(2), 56–65 (2002).

[2] Ebrahimi, M. and Vrscay, E., "Solving the inverse problem of image zooming using 'self-examples'," in [*ICIAR07*], 117–130 (2007).

[3] Li, D., Simske, S. J., and Mersereau, R. M., "Single image super-resolution based on support vector regression," in [*IJCNN*], 2898–2901 (2007).

[4] Jiji, C. V. and Chaudhuri, S., "Single-frame image super-resolution through contourlet learning," *EURASIP J. Appl. Signal Process.* **2006**(1), 235–235 (2008).

[5] Fisher, Y., "Fractal image compression. SIGGRAPH'92 course notes," (1992).

[6] Chen, Y., Luo, Y., and Hu, D., "Image superresolution using fractal coding," *Optical Engineering* **47**(1), 017007 (2008).

[7] Luong, H. Q., Ledda, A., and Philips, W., "Non-local interpolation," in [*ICIP*], 693–696 (2006).

[8] Protter, M., Elad, M., Takeda, H., and Milanfar, P., "Generalizing the non-local-means to super-resolution reconstruction," *IEEE Transactions on Image Processing* (2008).

[9] Peyré, G., Bougleux, S., and Cohen, L., "Non-local regularization of inverse problems," in [*ECCV08*], III: 57–68 (2008).

[10] Buades, A., Coll, B., and Morel, J. M., "A review of image denoising algorithms, with a new one," *Multiscale Modeling & Simulation* **4**(2), 490–530 (2005).

[11] Rudin, L. I., Osher, S. J., and Fatemi, E., "Nonlinear total variation based noise removal algorithms," *Physica D* **60**, 259–268 (1992).

[12] Malgouyres, F. and Guichard, F., "Edge direction preserving image zooming: A mathematical and numerical analysis," *SIAM Journal on Numerical Analysis* **39**(1), 1–37 (2002).

[13] Chan, T. F. and Shen, J., "Mathematical models for local nontexture inpaintings," *SIAM Journal on Applied Mathematics* **62**(3), 1019–1043 (2002).

[14] Chan, T. F., Osher, S., and Shen, J., "The digital TV filter and non-linear denoising," *IEEE Transactions on Image Processing* **10**, 231–241 (2001).

[15] Luon, H. Q. and Philips, W., "Sharp image interpolation by mapping level curves," *Proceedings for Visual Communications and Image Processing* , 2012–2022 (2005).

[16] Muresan, D. D., "Fast edge directed polynomial interpolation," in [*ICIP (2)*], 990–993 (2005).

[17] Jiang, H. and Moloney, C., "A new direction adaptive scheme for image interpolation," in [*ICIP (3)*], 369–372 (2002).

[18] Kraus, M., Eissele, M., and Strengert, M., "GPU-based edge-directed image interpolation," in [*SCIA07*], 532–541 (2007).

[19] Li, X. and Orchard, M. T., "New edge-directed interpolation," *IEEE Transactions on Image Processing* **10**, 1521–1527 (2001).

[20] Stepin, M., "HQ2x magnification filter," www.hiend3d.com/hq2x.html (2003).

[21] Harten, A., Engquist, B., Osher, S., and Chakravarthy, S. R., "Uniformly high order accurate essentially non-oscillatory schemes, III," *J. Computational Physics* **71**(2), 231–303 (1987).

[22] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., "Image quality assessment: from error measurement to structural similarity," *IEEE Transactions on Image Processing* **13**, 600–612 (2004).

[23] onOne Software, "Genuine Fractals," www.ononesoftware.com (2008).

[24] Horiuchi, T., Watanabe, K., and Tominaga, S., "Adaptive filtering for color image sharpening and denoising," in [*ICIAPW '07*], 196–201, IEEE Computer Society (2007).