Published in Image Processing On Line on 2011–09–01.
Submitted on 2011–00–00, accepted on 2011–00–00.
ISSN 2105–1232 © 2011 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2011.g_igcs

Image Interpolation with Geometric Contour Stencils

Pascal Getreuer

Yale University (pascal.getreuer@yale.edu)

Communicated by Gabriele Facciolo *Demo edited by* Pascal Getreuer

This IPOL article is related to a companion publication in the SIAM Journal on Imaging Sciences:

P. Getreuer. "Contour Stencils: Total Variation along Curves for Adaptive Image Interpolation." *SIAM Journal on Imaging Sciences*, vol. 4, no. 3, pp. 954–979, 2011. http://dx.doi.org/10.1137/100802785

Abstract

We consider the image interpolation problem where, given an image v with uniformly-sampled pixels $v_{m,n}$ and point spread function h, the goal is to find a function $u(x, y)$ satisfying

$$v_{m,n} = (h * u)(m, n) \quad \text{for all} \quad m, n \in \mathbb{Z} \tag{1}$$

so that u approximates the underlying function from which v was sampled.

This article improves upon the IPOL article "Image Interpolation with Contour Stencils" [7]. In the previous work, contour stencils are used to estimate the image contours locally as short line segments. This article begins with a continuous formulation of total variation integrated over a collection of curves and defines contour stencils as a consistent discretization. This discretization is more reliable than the previous approach and can effectively distinguish contours that are locally shaped like lines, curves, corners, and circles. These improved contour stencils sense more of the geometry in the image.

Interpolation is performed using an extension of the method described in the previous article. Using the improved contour stencils, there is an increase in image quality while maintaining similar computational efficiency.

Source Code

ANSI C source code to produce the same results as the demo is accessible on the article web page https://doi.org/10.5201/ipol.2011.g_igcs. Future software releases and updates will be posted at http://dev.ipol.im/~getreuer/code.

Keywords:

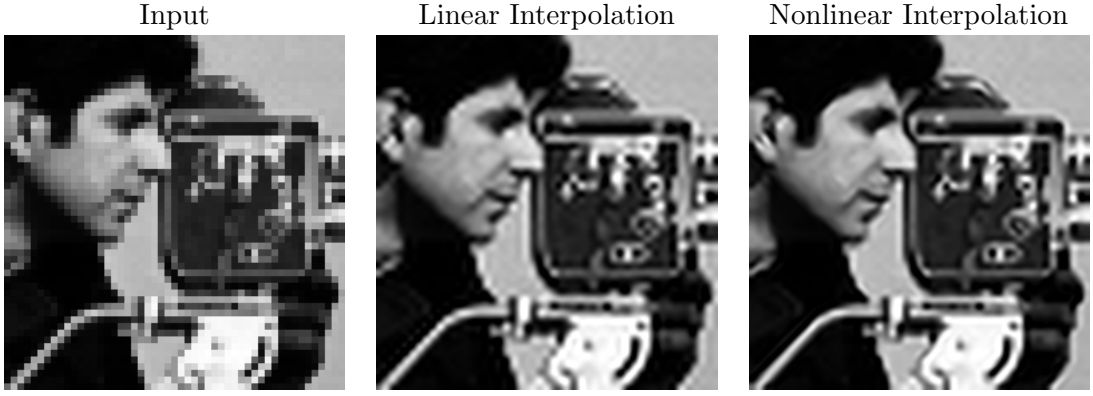
1 Introduction

The goal of the method is, given image $v_{m,n}$, to construct a function $u(x, y)$ that approximately satisfies

$$v_{m,n} = (h * u)(m, n) \quad \text{for all } m, n \in \mathbb{Z} \tag{2}$$

in such a way that the contours of u are close to the contours in v. The function h is a point spread function (PSF) that models how v was sampled. The reason for including it is that interpolation in this manner incorporates deconvolution, which sharpens the image and usually improves perceived image quality.

The proposed method only solves $v_{m,n} = (h * u)(m, n)$ *approximately*. Solving it exactly is a global and computationally formidable problem, yet it can be accurately and efficiently approximated locally.



Comparison of linear cubic B-spline interpolation and the proposed nonlinear method. Nonlinear interpolation adapts to the edge orientations, significantly reducing jagged edge artifacts and overshoots.

Two desirable properties of image interpolation are to limit jagged edge artifacts as well as overshoots and ringing artifacts. In other words, the interpolation should reproduce edges at any orientation and it should not overshoot or oscillate unnecessarily. To capture both of these objectives, we aim to have *contour consistency*: at any point, the orientation of the contours in u are the same as the orientation of the contours in v. It is not possible to achieve this with a linear method, nonlinearity is crucial to adapt to the image edges and geometry.

The proposed method has two steps: first, estimate the contours in v, and second, interpolate the image according to the estimated contours.

For the first step, we develop a method called *contour stencils* based on measuring total variation along curves. For the second step, we construct the interpolant as a linear combination of oriented Gaussian functions. To find a solution that accurately satisfies $v_{m,n} = (h * u)(m, n)$, we apply the iterative refinement algorithm: an initial interpolation is performed, and this is used to compute a prefiltered input that is adjusted such that its interpolation satisfies $v_{m,n} = (h * u)(m, n)$ more accurately. Multiple passes of prefiltering can be performed to reach a desired level of accuracy.

2 Total Variation Along Curves

Define the *total variation (TV) along curve C*

$$\|u\|_{\mathrm{TV}(C)} = \int_0^T \left| \tfrac{\partial}{\partial t} u\big(\gamma(t)\big) \right| dt, \quad \gamma : [0, T] \to C \tag{3}$$

where γ is a smooth parameterization of C. The motivating idea of contour stencils is that if the TV along a curve C is small, then C is approximately a contour.

In earlier works [5, 6, 7], contour stencils are defined as the discrete TV estimate

$$\|v\|_{\mathrm{TV}(C)} \approx \sum_{m,n\in\mathbb{Z}^2} \mathcal{S}(m,n)|v_m - v_n|, \tag{4}$$

where $\mathcal{S} : \mathbb{Z}^2 \times \mathbb{Z}^2 \to \mathbb{R}^+$ describes weighted edges, that is, the edge weight between pixels m and n is $\mathcal{S}(m,n)$, where a larger value implies a stronger connection. These edges are selected so that the stencil approximately follows C.

3 Discretization

The key improvement compared to the previous method [7] is a revised discretization of contour stencils. Instead of considering the TV along a single curve, we consider the integral over a collection of curves

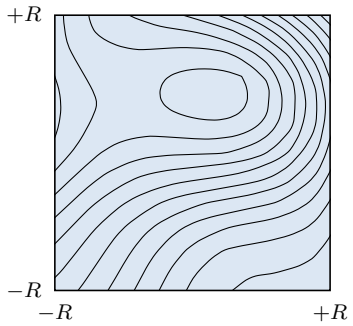
$$\int_{\mathbb{R}} \|u\|_{\mathrm{TV}(\psi^{-1}(\lambda))}\, d\lambda, \tag{5}$$

where the collection of curves are the contours of a function $\psi : \Omega \to \mathbb{R}$. Provided regularity conditions on u and ψ, the coarea formula [1] allows the integral to be rewritten as

$$\int_{\Omega} |\nabla\psi| \left| \tfrac{\nabla\psi^\perp}{|\nabla\psi|} \cdot \nabla u \right| dx. \tag{6}$$

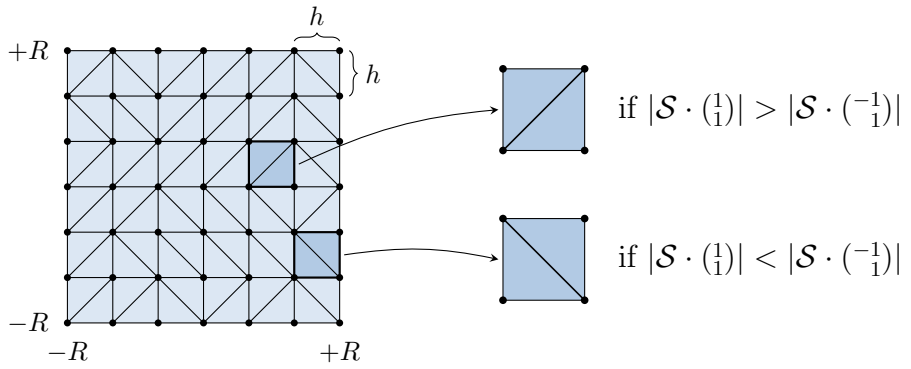
We define contour stencils as a discretization of this integral. Suppose that the domain Ω is the square $\{x : |x_1| < R, |x_2| < R\}$. Given a resolution $h = R/N$, Ω is partitioned into cells $\Omega_{i,j}$ of size $h \times h$. Defined the average of $\nabla\psi^\perp$ over each cell,

$$\mathcal{S}(x) = \frac{1}{h^2} \int_{\Omega_{i,j}} \nabla\psi(x)^\perp\, dx, \quad x \in \Omega_{i,j}. \tag{7}$$



Curves C_λ in Ω

An example collection of curves $\psi^{-1}(\lambda)$ of a function $\psi : \Omega \to \mathbb{R}$.



Triangulation of Ω Splitting the cells $\Omega_{i,j}$

Let T be a triangulation of Ω that splits each $\Omega_{i,j}$ into two triangles. A cell can be split into two triangles in two ways. The choice of splitting is along the diagonal which is closer to the direction of $\mathcal{S}$ in that cell. Let $\mathcal{I}u$ denote the piecewise linear interpolation that agrees with u at the grid points (hi, hj) and $\mathcal{I}u$ is affine on each triangle of T. Then the TV integral is discretized as

$$\int_\Omega |\mathcal{S}\cdot\nabla\mathcal{I}u|\,dx \approx \int_\Omega |\nabla\psi^\perp\cdot\nabla u|\,dx. \tag{8}$$

We call the discretized vector field $\mathcal{S}$ a *contour stencil* because it describes a predetermined pattern of contours to test on the image. Over a single cell, the integral can be evaluated in closed form as

$$V_{i,j}(\mathcal{S},u) := \int_{\Omega_{i,j}} |\mathcal{S}\cdot\nabla\mathcal{I}u|\,dx = \begin{cases} \frac{h}{2}\big(\big|\alpha u_{i+1,j+1} - (\alpha-\beta)u_{i,j+1} - \beta u_{i,j}\big| & \alpha\beta \geq 0, \\ \quad + \big|\beta u_{i+1,j+1} + (\alpha-\beta)u_{i+1,j} - \alpha u_{i,j}\big|\big) & \\ \frac{h}{2}\big(\big|\alpha u_{i,j+1} - (\alpha+\beta)u_{i+1,j+1} + \beta u_{i+1,j}\big| & \alpha\beta \leq 0. \\ \quad + \big|\beta u_{i,j+1} - (\alpha+\beta)u_{i,j} + \alpha u_{i+1,j}\big|\big) & \end{cases} \tag{9}$$

where $(\alpha,\beta)^T$ is the value of $\mathcal{S}$ in $\Omega_{i,j}$ and $u_{i,j} := u(hi, hj)$. The choice of splitting the cell is such that the $(\alpha \pm \beta)$ factors cancel at diagonal orientations where $|\alpha| = |\beta|$. The integral over the whole domain is

$$V(\mathcal{S},u) := \int_\Omega |\mathcal{S}\cdot\nabla\mathcal{I}u|\,dx = \sum_{i=-N}^{N-1}\sum_{j=-N}^{N-1} V_{i,j}(\mathcal{S},u). \tag{10}$$

As shown in the joint SIIMS article [8], $V(\mathcal{S},u)$ is a consistent discretization with first-order accuracy.

To extend to vector-valued images, the TV is computed as the sum of $V(\mathcal{S},u)$ applied to each component. For color images, we use the color components in $\mathrm{YP_BP_R}$ representation,

$$\begin{pmatrix} Y \\ P_B \\ P_R \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \tag{11}$$

4 Geometric Features

To design a contour stencil $\mathcal{S}$, we must select a function ψ. Since the observed image is supposed as having been blurred by PSF h, we use $\psi = h * \varphi$ to estimate the contours of the observed image and then refer to φ as a model of the contours of the underlying image.

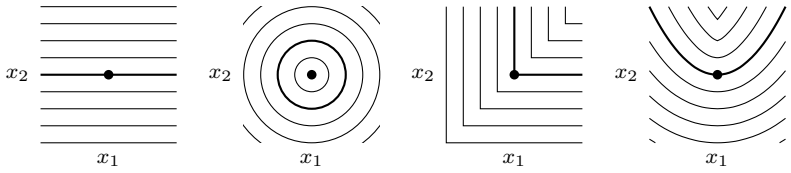
The function φ is selected as a signed distance function from an elementary curve C, that is, φ is a solution of

$$\begin{cases} |\nabla\varphi(x)| = 1 & \text{in } \Omega, \\ \varphi(x) = 0 & \text{on } C. \end{cases} \tag{12}$$

A distance function is a good choice for φ since its contours are uniformly spaced and approximately parallel. The stencil is then obtained by discretizing $\psi = h * \varphi$ as

$$\mathcal{S}(x) = \frac{1}{h^2}\int_{\Omega_{i,j}} \nabla\psi(x)^\perp\,dx, \quad x \in \Omega_{i,j}. \tag{13}$$

Then, the stencil is divided by $\|\mathcal{S}\|_1$ so that the vector field has normalized average magnitude. This way the TV estimates of different stencils are fairly compared.



Contours of φ for several different curves.

Lines. The most basic curve is a straight line, $C = \{x_2 = 0\}$, which has distance function $\varphi(x) = x_2$.

Circles. A circle $C = \{x_1^2 + x_2^2 = 1\}$ has distance function $\varphi(x) = \sqrt{x_1^2 + x_2^2} - 1$. Notice that the contours of φ includes all circles centered at the origin, so stencils based on this distance function have a scale invariance. For this reason, we treat the circle as an isotropic model of the contours, as it prefers no direction and has arbitrary curvature.

Corners. A right-angle corner $C = \{x_1x_2 = 0, x_1, x_2 \geq 0\}$ has distance function $\varphi(x) = \min\{x_1, x_2\}$.

Parabolas. To represent curves, it is desirable to have the distance function for a parabola $C = \{x_2 = \frac{a}{2}x_1^2\}$. To compute its distance function, consider a starting location $(s, \frac{a}{2}s^2)$ on the curve. The orthogonal direction from the curve at this point is $(-\partial_s\frac{a}{2}s^2, 1) = (-as, 1)$. Therefore, solving along characteristic lines, the distance function is

$$\varphi(s - ast, \tfrac{a}{2}s^2 + t) = t\sqrt{1 + a^2s^2}$$

for t sufficiently close to zero. The characteristics can cross for large t, where the characteristic producing the minimum value is the correct solution.

To express φ directly, we must solve for s and t in terms of x in

$$\begin{cases} x_1 = s - ast, \\ x_2 = \frac{a}{2}s^2 + t. \end{cases}$$

Eliminating t yields a cubic polynomial in s,

$$(x_1 - s) + (x_2 - \tfrac{a}{2}s^2)as = 0.$$

For positive a, the solution selecting the appropriate root is

$$|s| = \begin{cases} \left|\frac{|x_1|}{a^2} + \sqrt{\frac{x_1^2}{a^4} + z^3}\right|^{1/3} + \operatorname{sign}(x_2 - \frac{1}{a})\left|\frac{|x_1|}{a^2} - \sqrt{\frac{x_1^2}{a^4} + z^3}\right|^{1/3} & \text{if } \dfrac{x_1^2}{a^4} + z^3 \geq 0, \\ 2|z|^{1/2}\cos\Big(\frac{1}{3}\arccos\big(\frac{|x_1|}{a^2|z|^{3/2}}\big)\Big) & \text{if } \dfrac{x_1^2}{a^4} + z^3 < 0, \end{cases} \tag{14}$$

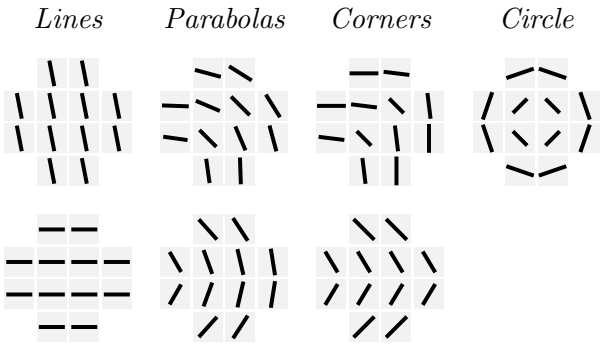
where $z = \frac{2}{3a^2}(1 - ax_2)$. Finally, the signed distance is obtained as $\varphi(x) = (x_2 - \frac{a}{2}s^2)\sqrt{1 + (as)^2}$.

5 Stencil Selection

At every pixel k in the image, contour stencils are tested over a neighborhood of k. The contour stencil with the smallest TV estimate is selected as a model of the underlying contours,

$$\mathcal{S}^\star(k) = \operatorname*{arg\,min}_{\mathcal{S}\in\Sigma} V\big(\mathcal{S}, v(\cdot - k)\big), \tag{15}$$

where Σ is a set of candidate stencils.



Stencils for different kinds of features.

To detect geometric features at different orientations, stencils from multiple rotations of the distance functions are used

$$\varphi_\theta(x) = \varphi\big(\left(\begin{smallmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{smallmatrix}\right)x\big). \tag{16}$$

The stencil set Σ used in the examples comprises 32 line-shaped stencils at orientations $\theta = j\frac{\pi}{32}$, 8 corner-shaped stencils, 8 parabola-shaped stencils with $a = \sqrt{2}$, and another 8 parabola-shaped stencils with $a = 1$ at orientations $j\frac{\pi}{4}$, and a circle-shaped stencil for a total of 57 stencils. Instead of a square neighborhood, we use a 12-cell neighborhood approximating a disk for better isotropy.

A small neighborhood size is desirable so that TV estimates are localized and computationally efficient. On the other hand, a larger neighborhood aggregates more samples, which improves robustness against noise and accuracy in distinguishing different structures. The proposed 12-cell neighborhood seems to be a good balance of these considerations.

It is possible that the minimum in

$$\mathcal{S}^\star(k) = \operatorname*{arg\,min}_{\mathcal{S}\in\Sigma} V\big(\mathcal{S}, v(\cdot - k)\big) \tag{17}$$

is not unique. To ensure a stable minimum, the separation is tested between the TV estimates of the best $\mathcal{S}^\star$ and second best $\mathcal{S}^{\star 2}$ stencils,

$$\frac{V(\mathcal{S}^{\star 2}, v) - V(\mathcal{S}^\star, v)}{4\sqrt{2}} \geq T. \tag{18}$$

The best-fitting stencil $\mathcal{S}^\star$ is accepted if this quantity is above the threshold, otherwise, the estimation falls back to an isotropic model. This ensures that a perturbation v' of v with $\|v' - v\|_\infty < T$ does not change the best-fitting stencil. The threshold is set at $T = 10^{-4}$ relative to pixel intensities in the range $[0, 1]$.

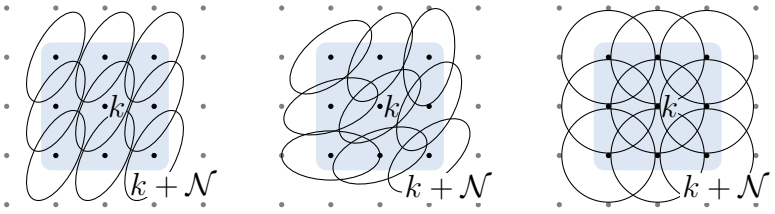
Although the stencil set Σ used here has 57 stencils, reasonably good results are possible with far fewer stencils. For example, the previous work [7] uses only 8 line-shaped stencils, and this is sufficient for a rough estimation of the contours and simplifies implementation. The advantage of

using a larger stencils set is that more rotations and more geometric structures allow for a finer estimation, which improves image interpolation quality.

Computationally, the cost of contour estimation with contour stencils is small compared to interpolation itself. The TVs can be efficiently approximated by first computing the $V_{i,j}$ cell TVs at uniformly spaced orientations and then combining them to obtain the stencil TVs (see Stencil Selection in the Algorithm section). With this approach, the cost does not depend significantly on the size of the stencil set.

6 Interpolation

The interpolation is an extension of the previous method [7] to stencils of any shape. The interpolation is constructed locally about each pixel over a small window, then these local reconstructions are combined to obtain the global interpolation.



Local reconstruction with oriented functions $\rho^n_{\mathcal{S}}$. Left: Reconstruction with a line-shaped stencil. Center: With a parabola-shaped stencil. Right: With the circle stencil.

For every pixel k in the input image v, let

$$u_k(x) = v_k + \sum_{n\in\mathcal{N}} c_n\, \rho^n_{\mathcal{S}^\star(k)}(x-n), \tag{19}$$

where $\mathcal{N} \subset \mathbb{Z}^2$ is a neighborhood of the origin, the c_n are coefficients to be determined, and $\rho^n_{\mathcal{S}}$ is a function oriented in the direction

$$\theta^n_{\mathcal{S}} = \angle \int_{[-\frac{1}{2},+\frac{1}{2}]^2} \nabla\varphi^{\perp}_{\mathcal{S}}\big(x-n\big)\,dx \tag{20}$$

with anisotropy

$$\mu_{\mathcal{S}} = \min_{n\in\mathcal{N}} \Big| \int_{[-\frac{1}{2},+\frac{1}{2}]^2} \nabla\varphi_{\mathcal{S}}\big(x-n\big)\,dx \Big| \tag{21}$$

so that it follows the contours modeled by the best-fitting stencil $\mathcal{S}^\star(k)$ at point $k+n$. The larger the anisotropy parameter, the more strongly the function points in the direction $\theta^n_{\mathcal{S}}$. If $\mu_{\mathcal{S}}$ is zero, then $\rho^n_{\mathcal{S}}$ should prefer no direction. The essential changes compared to the previous work [7] are that the $\rho^n_{\mathcal{S}}$ may have a different orientations for different points n in the window and the use of the anisotropy parameter.

The c_n coefficients are selected so that the discretization model is satisfied locally, $(h * u_k)(m) = v_{k+m}$ for $m \in \mathcal{N}$. This implies the c_n satisfy

$$\sum_n (A_{\mathcal{S}^\star(k)})_{m,n} c_n = v_{k+m} - v_k, \quad n \in \mathcal{N}, \tag{22}$$

where $A_{\mathcal{S}}$ is a matrix with elements $(A_{\mathcal{S}})_{m,n} = (h * \rho^n_{\mathcal{S}})(m-n)$.

The global interpolation is obtained by combining the u_k,

$$u(x) = \sum_{k\in\mathbb{Z}^2} w(x-k)u_k(x-k), \tag{23}$$

where w is a window function satisfying $\sum_k w(x-k) = 1$ for all $x \in \mathbb{R}^2$ and $w(k) = 0$ for $k \in \mathbb{Z}^2\backslash\mathcal{N}$.

It is convenient for computation to define the functions

$$\tilde{\rho}^n_{\mathcal{S}}(x) = w(x) \sum_{m\in\mathcal{N}} (A^{-1}_{\mathcal{S}})_{m,n}\, \rho^m_{\mathcal{S}}(x-m), \tag{24}$$

then the interpolation can be expressed directly in terms of v as

$$u(x) = \sum_{k\in\mathbb{Z}^2} \Big[w(x-k)v_k + \sum_{n\in\mathcal{N}\backslash\{0\}} (v_{k+n} - v_k)\, \tilde{\rho}^n_{\mathcal{S}^\star(k)}(x-k) \Big]. \tag{25}$$

6.1 Iterative Refinement (Prefiltering)

Although each u_k locally satisfies the discretization model exactly, the global interpolation u only satisfies sample$(h * u) = v$ approximately. To improve the accuracy in satisfying the discretization model, we use iterative refinement as previously [7].

Iterative refinement is an algorithm for finding a solution of a linear system $Ax = y$ using an approximate right inverse B:

$$x^0 = 0, \quad \begin{cases} r^i = y - Ax^i, \\ x^{i+1} = x^i + Br^i. \end{cases} \tag{26}$$

Equivalently, iterative refinement can be expressed in terms of modifying the right-hand side y:

$$y^0 = y, \quad y^{i+1} = y^i + (y - ABy^i). \tag{27}$$

If A is invertible, then the error in x^i is $e^i := A^{-1}(y - Ax^i) = A^{-1}r^i$ and the iteration adds the correction Br^i to approximate e^i. More generally, iterative refinement can be applied to an underdetermined system where AB is close to identity. The iteration converges if $(I - AB)$ is a contraction,

$$x^{i+1} = By^i = B\sum_{j=0}^{i} (I - AB)^j y \quad \overset{i\to\infty}{\longrightarrow} \quad B(AB)^{-1}y = A^{-1}y. \tag{28}$$

Let $\mathcal{R}$ denote the operator

$$(\mathcal{R}v)(x) := \sum_{k\in\mathbb{Z}^2} w(x-k)\Big[v_k + \sum_{n\in\mathcal{N}} (v_{k+n} - v_k)\tilde{\rho}^n_{\mathcal{S}^\star(k)}(x-k)\Big]. \tag{29}$$

The interpolation is nonlinear because $\mathcal{R}$ depends on the best-fitting stencils $\mathcal{S}^\star(k)$. However, regarding the stencils as fixed parameters, $\mathcal{R}$ is a linear operator on v. The operation sample$(h * \cdot)$ has the role of A and $\mathcal{R}$ has the role of the approximate right inverse B. In the previous method [7], the deconvolution accuracy is improved by the iteration

$$u^0 = 0, \quad \begin{cases} r^i = v - \text{sample}(h * u^i), \\ u^{i+1} = u^i + \mathcal{R}r^i. \end{cases} \tag{30}$$

It is equivalent but computationally advantageous to iterate as modifying v,

$$v^0 = v, \quad v^{i+1} = v^i + \big(v - \text{sample}(h * \mathcal{R}v^i)\big). \tag{31}$$

In this form, v is prefiltered and the final interpolation is evaluated directly as $u = \mathcal{R}v^i$. The convenience of this approach is that the grid used to approximate the interpolation and convolution $h * \mathcal{R}v^i$ need not be the same as the grid used for the final interpolation. It is computationally efficient and only a small loss in accuracy to compute v^i on a coarser grid than the grid used for the final interpolation. By using this prefiltering approach to iterative refinement, the proposed method is faster than the previous method [7], especially for large interpolation factors.

To ensure convergence, suppose that $C < 1$ exists such that for any v and k

$$|v_k - (h * \mathcal{R}v)(k)| \le C|v_k|, \tag{32}$$

then v^i converges in ℓ^∞ and u^i converges uniformly to u that satisfies $\text{sample}(h * u) = v$ exactly. Unfortunately, we have no proof of a practical condition that guarantees $C < 1$. Numerical experiments suggest that the iteration converges reliably with the suggested parameters and h with standard deviation less than 0.8.

6.2 Parameters

The following parameters are fixed in the experiments: h is a Gaussian with standard deviation 0.5, $\mathcal{N} = \{-1, 0, 1\} \times \{-1, 0, 1\}$, w is the cubic B-spline

$$w(x, y) = B(x)B(y), \quad B(t) = \big(1 - |t| + \tfrac{1}{6}|t|^3 - \tfrac{1}{3}\big|1 - |t|\big|^3\big)^+,$$

$\rho^n_{\mathcal{S}}$ is an oriented Gaussian

$$\rho^n_{\mathcal{S}}(x, y) = \exp\Big(-\frac{\tau^2}{2\sigma_\tau^2} - \frac{\nu^2}{2\sigma_\nu^2}\Big), \quad \begin{pmatrix}\tau\\ \nu\end{pmatrix} = \begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}$$

with $\sigma_\tau = 1.2$ and $\sigma_\nu = \sigma_\tau(1 - \mu^4/2)$, and θ and μ are the orientation and anisotropy parameters for $\mathcal{S}$ at n, and three iterations of iterative refinement are applied (two passes of prefiltering and one final interpolation).

7 Algorithm

7.1 Constructing the Stencils

The first step is to construct the contour stencils. For each type of geometric feature, the signed distance function φ needs to be determined or numerically approximated. Distance functions for several elementary curves are given in the previous section. For each φ, the corresponding stencil is obtained by averaging over cells,

$$\mathcal{S}(x) = \frac{1}{h^2}\int_{\Omega_{i,j}} \nabla\psi(x)^\perp\, dx, \quad x \in \Omega_{i,j},$$

where $\psi = h * \varphi$ and h is the point spread function. This two-dimensional integral over the cell is simplified by applying the fundamental theorem of calculus,

$$\mathcal{S}_1(x) = -\frac{1}{h^2}\Big[\int_0^h \psi(i + t, j + h)\, dt - \int_0^h \psi(i + t, j)\, dt\Big], \tag{33}$$

$$\mathcal{S}_2(x) = \frac{1}{h^2}\Big[\int_0^h \psi(i + h, j + t)\, dt - \int_0^h \psi(i, j + t)\, dt\Big]. \tag{34}$$

For numerical implementation, φ and h can first be evaluated on a fine grid and convolved to approximate ψ. The one-dimensional integrals along the cell boundaries can then be performed by trapezoidal rule to obtain the stencil $\mathcal{S}$. Finally, $\mathcal{S}$ is divided by $\|\mathcal{S}\|_1$ so that all stencils have the same average magnitude.

7.2 Interpolation Precomputations

Some quantities can be precomputed to speed up the interpolation. For each stencil, the orientation and anisotropy parameters are computed,

$$\theta_{\mathcal{S}}^n = \angle \int_{[-\frac{1}{2},+\frac{1}{2}]^2} \nabla\varphi_{\mathcal{S}}^{\perp}\big(x-n\big)\,dx, \tag{35}$$

$$\mu_{\mathcal{S}} = \min_{n\in\mathcal{N}} \Big| \int_{[-\frac{1}{2},+\frac{1}{2}]^2} \nabla\varphi_{\mathcal{S}}\big(x-n\big)\,dx \Big|, \tag{36}$$

where $n \in \mathcal{N} = \{-1,0,1\}\times\{-1,0,1\}$. Note that the orientation $\theta_{\mathcal{S}}^n$ is in general different for each point in $\mathcal{N}$ but the anisotropy parameter is the same. The integrals can be approximated by the same approach used in the stencil construction.

Next, the interpolation matrix is computed

$$(A_{\mathcal{S}})_{m,n} = (h * \rho_{\mathcal{S}}^n)(m-n), \quad m,n \in \mathcal{N}, \tag{37}$$

where the $\rho_{\mathcal{S}}^n$ are determined by the orientation and anisotropy parameters determined in the previous step. For a Gaussian PSF h, the convolution can be evaluated in closed form,

$$A_{m,n} = \frac{\sigma_\tau\sigma_\nu}{\sqrt{\sigma_h^2+\sigma_\tau^2}\sqrt{\sigma_h^2+\sigma_\nu^2}} \exp\left(-\frac{d_1^2}{2(\sigma_h^2+\sigma_\tau^2)} - \frac{d_2^2}{2(\sigma_h^2+\sigma_\nu^2)}\right) \tag{38}$$

with $\binom{d_1}{d_2} = \left(\begin{smallmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{smallmatrix}\right)(m-n)$, or for general h the convolution can be approximated by numerical quadrature. The inverse of the interpolation matrix is computed and stored for later use in the interpolation.

For scaling by an integer scale factor, samples of the functions $\tilde{\rho}_{\mathcal{S}}^n$ and the window function w can be precomputed as well,

$$\tilde{\rho}_{\mathcal{S}}^n(x) = w(x) \sum_{m\in\mathcal{N}} (A_{\mathcal{S}}^{-1})_{m,n}\, \rho_{\mathcal{S}}^m(x-m). \tag{39}$$

Notice that the stencil set construction and interpolation precomputations can all be done prior to knowing the input image. So for a fixed h and scale factor, these only need to be computed once to interpolate any number of images.

7.3 Stencil Selection

Before the interpolation itself, the best-fitting stencil is determined for each pixel k of the input image,

$$\mathcal{S}^\star(k) = \operatorname*{arg\,min}_{\mathcal{S}\in\Sigma} V\big(\mathcal{S}, v(\cdot-k)\big),$$

where

$$V(\mathcal{S},v) := \int_\Omega |\mathcal{S}\cdot\nabla\mathcal{I}v|\,dx = \sum_{i=-N}^{N-1}\sum_{j=-N}^{N-1} V_{i,j}(\mathcal{S},v), \tag{40}$$

$$V_{i,j}(\mathcal{S},v) := \begin{cases} \frac{h}{2}\big(\big|\alpha v_{i+1,j+1} - (\alpha-\beta)v_{i,j+1} - \beta v_{i,j}\big| & \alpha\beta \geq 0,\\ \quad + \big|\beta v_{i+1,j+1} + (\alpha-\beta)v_{i+1,j} - \alpha v_{i,j}\big|\big) & \\ \frac{h}{2}\big(\big|\alpha v_{i,j+1} - (\alpha+\beta)v_{i+1,j+1} + \beta v_{i+1,j}\big| & \alpha\beta \leq 0,\\ \quad + \big|\beta v_{i,j+1} - (\alpha+\beta)v_{i,j} + \alpha v_{i+1,j}\big|\big) & \end{cases} \tag{41}$$

and $(\alpha, \beta)^T$ is the value of $\mathcal{S}$ in $\Omega_{i,j}$. The TV estimates of the best and second best stencils are compared. If the difference is above a threshold

$$\frac{V(\mathcal{S}^{\star 2}, v) - V(\mathcal{S}^{\star}, v)}{4\sqrt{2}} \geq 10^{-4}, \tag{42}$$

then the best-fitting stencil is accepted. Otherwise, the circle stencil is selected as an isotropic model of the contours at this point.

To speed up the computation of the TV estimates, the $V_{i,j}$ can first be computed for each cell of the image at uniformly spaced orientations,

$$V_{i,j}\big(\tbinom{\cos\theta}{\sin\theta}, v\big), \quad \theta = j\tfrac{\pi}{64}, j = 0, \ldots, 63.$$

The $V_{i,j}(\mathcal{S}, v)$ are then approximated by using the closest orientation

$$V_{i,j}(\mathcal{S}, v) \approx |\mathcal{S}_{i,j}| \cdot V_{i,j}\big(\tbinom{\cos\theta}{\sin\theta}, v\big), \quad |\theta - \angle\mathcal{S}_{i,j}| \leq \tfrac{\pi}{128}. \tag{43}$$

The advantage of this approach is that the cell TV estimates are reused many times in evaluating the TV for different stencils and between the overlapping neighborhoods of nearby pixels.

7.4 Interpolation

Here we describe how to evaluate operator $\mathcal{R}$ for a single pass of the interpolation. For scaling by an integer scale factor, the interpolation is computed as

$$u(x) = \sum_{k\in\mathbb{Z}^2} \Bigg[w(x-k)v_k + \sum_{n\in\mathcal{N}\setminus\{0\}} (v_{k+n} - v_k)\, \tilde{\rho}^n_{\mathcal{S}^\star(k)}(x-k) \Bigg]. \tag{44}$$

Since w has compact support, only a few terms are nonzero for each x. Interpolation is very efficient in this case because w and $\tilde{\rho}^n_{\mathcal{S}}$ can be precomputed, so the computation amounts to multiply-add operations.

For scaling by a non-integer scale factor or interpolation at arbitrary points, the coefficients c^k_n are computed for each local reconstruction u_k,

$$c^k_m = \sum_{n\in\mathcal{N}} (A^{-1}_{\mathcal{S}^\star(k)})_{m,n}(v_{k+n} - v_k), \quad m \in \mathcal{N}. \tag{45}$$

Then the global interpolation is computed as

$$u(x) = \sum_{k\in\mathbb{Z}^2} w(x-k)\Big[v_k + \sum_{m\in\mathcal{N}} c^k_m\, \rho^m_{\mathcal{S}^\star(k)}(x-m-k)\Big]. \tag{46}$$

The bottleneck operation in this formula are the many evaluations of the oriented Gaussian functions $\rho^m_{\mathcal{S}}$. A substantial speed improvement is gained by using look-up tables for the exp and trigonometry functions in these evaluations.

7.5 Iterative Refinement (Prefiltering)

Iterative refinement can be applied so that the interpolation more accurately approximates $v_{m,n} = (h*u)(m,n)$. This refinement sharpens the interpolation, which usually improves the perceived image quality. Iterative refinement is implemented by prefiltering the input image,

$$v^0 = v, \quad v^{i+1} = v^i + \big(v - \mathrm{sample}(h * \mathcal{R}v^i)\big). \tag{47}$$

For a typical PSF h, only two or three passes of iterative refinement are needed for convergence. Note that the best-fitting stencils are selected once and are fixed for each evaluation of $\mathcal{R}$. To approximate the interpolation and convolution $h * \mathcal{R}v^i$, the interpolation is performed on a grid with twice the resolution of v, and the convolution is approximated by discrete convolution.

The final interpolation is obtained by interpolating the prefiltered image $u = \mathcal{R}v^i$ at the desired points.

7.6 Complexity

The complexity of the interpolation is the same as with the previous method [7]. Each evaluation of $\mathcal{R}$ for factor-d scaling costs $\mathcal{O}(|\mathcal{N}|^2 d^2)$ operations per input pixel. For an integer scale-factor with the settings used in the examples, the cost is $864d^2$ operations per input pixel.

8 Examples

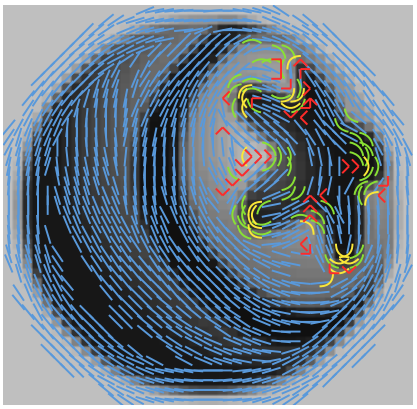
In the examples, h is a Gaussian with standard deviation 0.5. In practice, the standard deviation of h should estimate the blurriness of the actual PSF used to sample the input image. It is preferable to underestimate this value rather than overestimate: if it is smaller than the true standard deviation of the PSF, the result is blurrier, while using a standard deviation slightly too large creates ripple artifacts, which are visually more destructive. In the online demo, the default value for the standard deviation σ_h is 0.35, which reasonably models the blurriness of typical images.

The first two examples demonstrate the contour estimation. At every pixel in the image, the best-fitting stencil is determined, and a small line is drawn to indicate the estimated contour $C_{\mathcal{S}^\star(k)}$ modeled by the stencil. The stencil set includes lines, parabolas, corners, and the circle, so the method is able to distinguish these different geometric features. The lines are colored according to the type of feature.

In the first image, straight lines are selected as the contour model for most of the image and parabolas and some corners are used along the curving edge of the star.

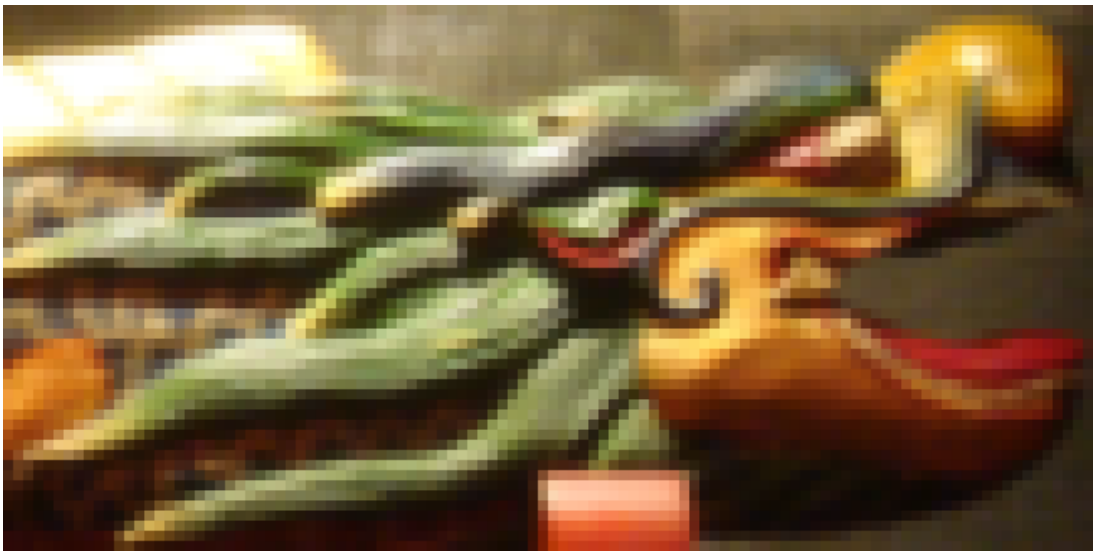
Input

Estimated Contours

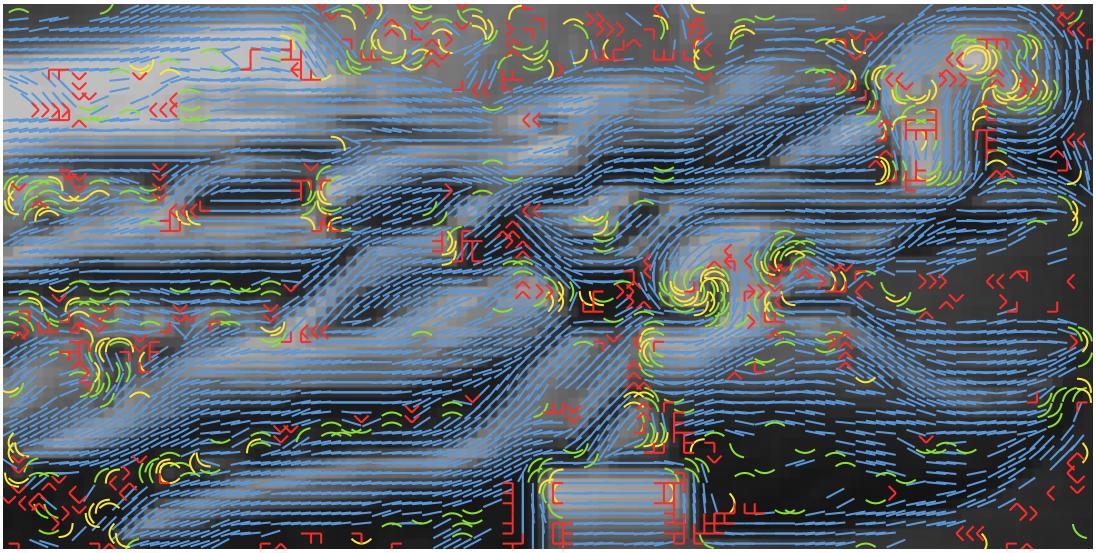


The next image demonstrates contour estimation on a natural image. Again, straight lines are selected over the low curvature parts of the image and parabolas and corners are selected in the sharper curves. Contour estimation is chaotic in the background of the image, which is textured but nearly constant (for example, along the top center and in lower right corner of the image). The estimation includes many corners in these regions, suggesting high curvature.

Input



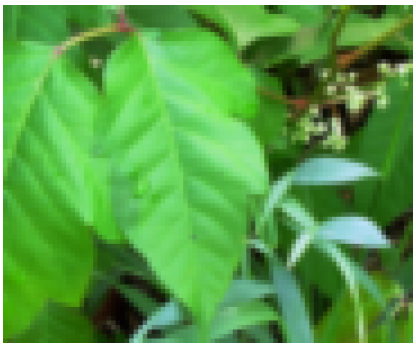
Estimated Contours



Next we examine the effects of iterative refinement for factor-4 image scaling. The interpolation u^1 is created from the input data directly, $u^1 = \mathcal{R}v$. Interpolation u^3 is created after applying two passes of prefiltering to the input, $u^3 = \mathcal{R}v^2$, which visibly sharpens the image and improves the PSNR and MSSIM (Appendix A explains how these metrics are computed).

Exact

Input



u^1 (initial interpolation)

PSNR 27.58, MSSIM 0.8434, CPU Time 0.073s

u^3 (two passes of prefiltering)

PSNR 28.11, MSSIM 0.8571, CPU Time 0.102s

The proposed method is an improvement over the previous method [7]. The most visible difference is improved quality on corners. The estimated contours below visualize how the proposed method adapts to the fine geometry of the window frame and the crenellated overhang.

Exact

Input

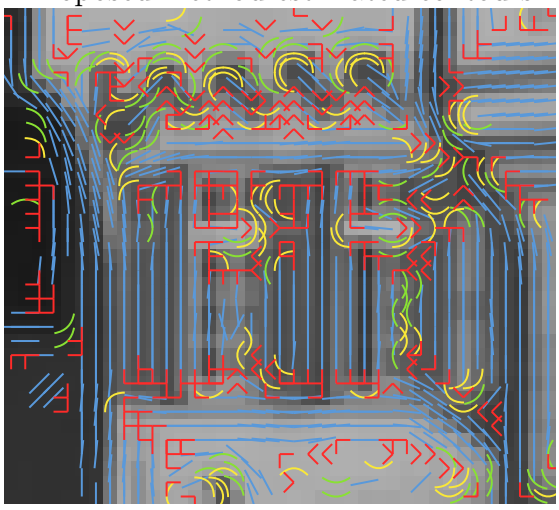
Previous method [7]

PSNR 18.52, MSSIM 0.5351

Proposed method

PSNR 18.79, MSSIM 0.5476

Proposed method estimated contours



The previous method has a tendency to join contours that are not actually connected, giving textured regions an unnatural painted quality. The proposed method reduces this effect somewhat. There is a subtle improvement in the fur in the next image.

Exact

Input

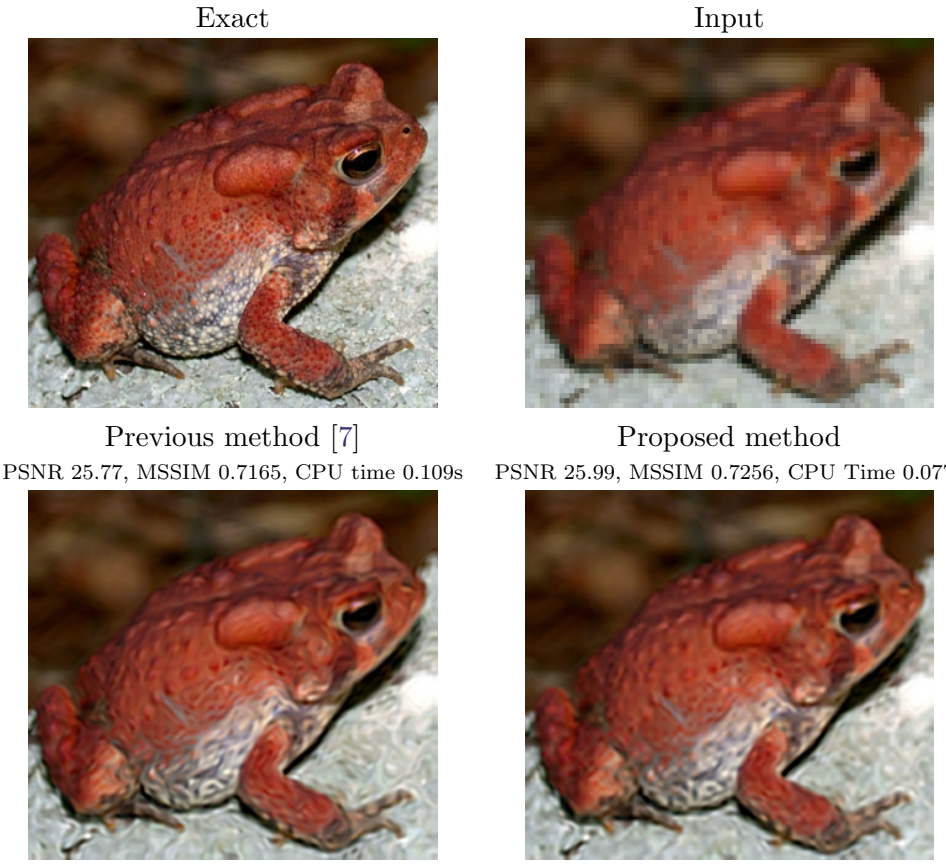
Previous method [7]

PSNR 22.47, MSSIM 0.6051, CPU time 0.115s

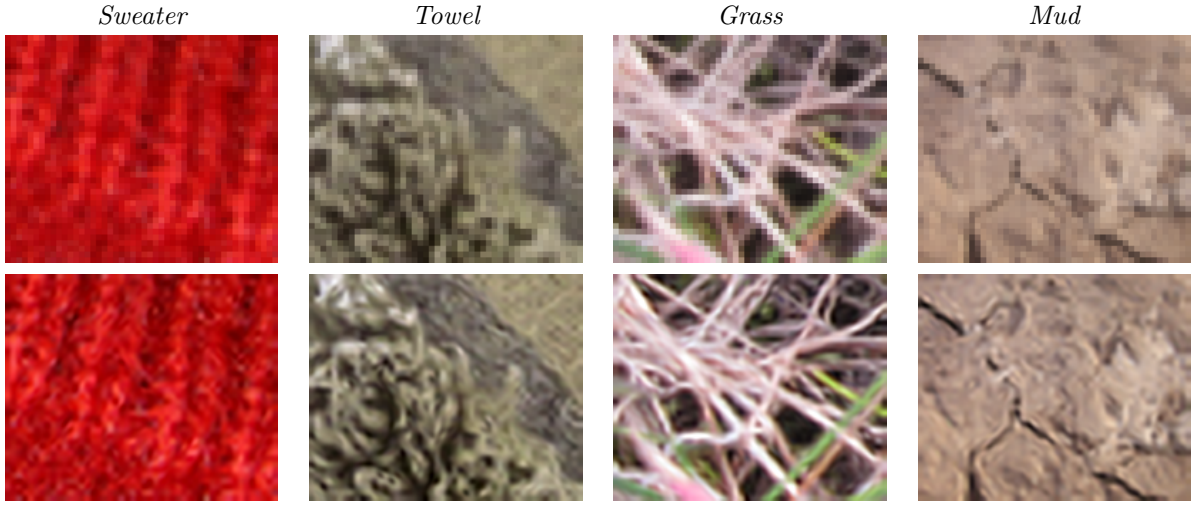
Proposed method

PSNR 22.69, MSSIM 0.6161, CPU Time 0.091s

As another example, the painted effect is reduced on the rough textures of this image.



Since the method tries to preserve the structure of the image contours, it can partially recover oriented textures where the contour lines flow smoothly.



A closely related problem to natural image interpolation is pixel art scaling. Pixel art images are created at the pixel level, usually at low resolution with a limited number of colors. Edge adaptivity is especially important for pixel art scaling to prevent jagged edge artifacts. This example compares several methods for factor-4 pixel art scaling. Pixel art is better modeled as point-value samples rather than weighted averages, so the proposed method is applied with PSF $h = \delta$ for this example.

For reference, bilinear interpolation is included to demonstrate that it is not satisfactory for pixel art scaling. Super 2× SaI [2] and HQ4X [3] are methods specialized for pixel art scaling, producing sharp cartoon-like results. Super 2× SaI is only defined for factor-2 scaling, so the method was applied twice for this interpolation. AQua-2 [4] and the proposed method perform reasonably since they are edge adaptive, though they are blurrier than Super 2× SaI and HQ4X. However, notice that none of the methods plausibly interpolate the dithered gradient in the background.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Award No. DMS-1004694. Work partially supported by the Office of Naval Research under grant N00014-97-1-0839 and by the European Research Council, advanced grant "Twelve labours."

Image Credits

 Cameraman standard test image

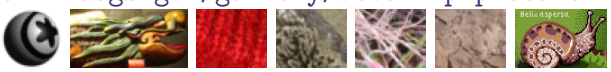
 Jan Miller, U.S. Fish and Wildlife Service, Public Domain (http://www.fws.gov/digitalmedia/cdm4/item_viewer.php?CISOROOT=/natdiglib&CISOPTR=6846)

 Kodak Image Suite, image 8 (http://r0k.us/graphics/kodak/)

 Gary Kramer, U.S. Fish and Wildlife Service, Public Domain (http://www.fws.gov/digitalmedia/

cdm4/item_viewer.php?CISOROOT=/natdiglib&CISOPTR=203)

 John D. Willson, USGS Amphibian Research and Monitoring Initiative, Public Domain (http://armi.usgs.gov/gallery/detail.php?search=Genus&subsearch=Bufo&id=323)

 Pascal Getreuer, CC-BY

References

[1] H. Federer. "Curvature measures." *Transactions of the American Mathematical Society*, 93, pp. 418–491, 1959. http://dx.doi.org/10.2307/1993504

[2] D. L. K. Fa. "2xSaI : The advanced 2x Scale and Interpolation engine." 2001. http://www.xs4all.nl/~vdnoort/emulation/2xsai/

[3] M. Stepin. "HQ4x magnication lter." 2003. http://www.hiend3d.com/hq4x.html

[4] D. D. Muresan, "Fast edge directed polynomial interpolation." in *International Conference on Image Processing* (2), pp. 990–993, 2005. http://dx.doi.org/10.1109/ICIP.2005.1530224

[5] P. Getreuer. "Contour stencils for edge-adaptive image interpolation." *Proceedings of SPIE*, vol. 7257, 2009. http://dx.doi.org/10.1117/12.806014

[6] P. Getreuer. "Image zooming with contour stencils." *Proceedings of SPIE*, vol. 7246, 2009. http://dx.doi.org/10.1117/12.805934

[7] P. Getreuer. "Image Interpolation with Contour Stencils." *Image Processing On Line*, 2011. http://dx.doi.org/10.5201/ipol.2011.g_iics

[8] P. Getreuer. "Contour Stencils: Total Variation along Curves for Adaptive Image Interpolation." *SIAM Journal on Imaging Sciences*, vol. 4, no. 3, pp. 954–979, 2011. http://dx.doi.org/10.1137/100802785

A Image Quality Metrics

A.1 ℓ^p Metrics

Let A and B be two color images to be compared, each with N pixels. We consider the images as vectors in $\mathbb{R}^{3N}$ with each pixel represented by red, green, blue intensities in $\{0, 1, \ldots, 255\}$. Several standard metrics can then be defined in terms of ℓ^p norms.

$$\begin{aligned}
&\text{Maximum absolute difference} && := \|A - B\|_\infty \\
&\text{Mean squared error (MSE)} && := \tfrac{1}{3N}\|A - B\|_2^2 \\
&\text{Root mean squared error (RMSE)} && := \sqrt{\text{MSE}} \\
& && = \tfrac{1}{\sqrt{3N}}\|A - B\|_2 \\
&\text{Peak signal-to-noise ratio (PSNR)} && := 10\log_{10}\tfrac{255^2}{\text{MSE}} \\
& && = 20\log_{10}\tfrac{255}{\text{RMSE}}
\end{aligned}$$

For the first three metrics, a smaller value implies a smaller discrepancy between A and B. For PSNR, a larger value implies a smaller discrepancy, with $\text{PSNR} = \infty$ when $A = B$.

A.2 MSSIM

The mean structural similarity (MSSIM) index[1] is a somewhat more complicated metric designed to agree better with perceptual image quality.

We first describe MSSIM on grayscale images. Let w be a Gaussian filter with standard deviation 1.5 pixels, and compute the following local statistics:

$$\begin{aligned} \mu_A &= w * A, & \mu_B &= w * B, \\ \sigma_A^2 &= w * (A - \mu_A)^2, & \sigma_B^2 &= w * (B - \mu_B)^2, \\ \sigma_{AB} &= w * \big((A - \mu_A)(B - \mu_B)\big). \end{aligned}$$

At every pixel, the structural similarity (SSIM) index is calculated as

$$\mathrm{SSIM} := \frac{(2\mu_A\mu_B + C_1)(2\sigma_{AB} + C_2)}{(\mu_A^2 + \mu_B^2 + C_1)(\sigma_A^2 + \sigma_B^2 + C_2)}$$

where $C_1 = (0.01 \cdot 255)^2$ and $C_2 = (0.03 \cdot 255)^2$. The mean SSIM (MSSIM) is the average SSIM value over the image.

For color images, we compute the MSSIM over each channel and take the average,

$$\mathrm{MSSIM} := \tfrac{1}{3}(\mathrm{MSSIM}_{\mathrm{red}} + \mathrm{MSSIM}_{\mathrm{green}} + \mathrm{MSSIM}_{\mathrm{blue}}).$$

The MSSIM index is always between 0 and 1. A larger value implies smaller discrepancy.

A.3 Computation Time

The computation time shown in the demo is computed using the UNIX `gettimeofday()` function to obtain the system time in units of nanoseconds. Note that the computation is affected by other tasks running simultaneously on the server, so the reported computation time is only a rough estimate.

[1] http://www.ece.uwaterloo.ca/~z70wang/research/ssim