Published in Image Processing On Line on 2011–08–01.
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_iics

Image Interpolation with Contour Stencils

Pascal Getreuer

CMLA, ENS Cachan, France
(pascal.getreuer@cmla.ens-cachan.fr)

Communicated by François Malgouyres *Demo edited by* Pascal Getreuer

Abstract

Image interpolation is the problem of increasing the resolution of an image. Linear methods must compromise between artifacts like jagged edges, blurring, and overshoot (halo) artifacts. More recent works consider nonlinear methods to improve interpolation of edges and textures. In this paper we apply contour stencils for estimating the image contours based on total variation along curves and then use this estimation to construct a fast edge-adaptive interpolation.

Source Code

The source code (ANSI C), its documentation, and the online demo are accessible at the IPOL web page of this article[1].

Keywords: image interpolation; image level-line analysis; total variation

1 Introduction

Image interpolation is the problem of increasing the resolution of an image. Linear methods have traditionally been preferred, for example, the popular bilinear and bicubic interpolations are linear methods. However, a linear method must compromise between artifacts like jagged edges, blurring, and overshoot (halo) artifacts. These artifacts cannot all be eliminated simultaneously while maintaining linearity.

More recent works consider nonlinear methods, especially to improve interpolation of edges and textures. An important aspect of nonlinear interpolation is accurate estimation of edge orientations. For this purpose we apply contour stencils, a new method for estimating the image contours based on total variation along curves. This estimation is then used to construct a fast edge-adaptive interpolation.

[1] https://doi.org/10.5201/ipol.2011.g_iics

2 Contour Stencils

The idea in contour stencils is to estimate the image contours by measuring the total variation of the image along curves. Define the *total variation* (TV) *along curve* C

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

where γ is a smooth parameterization of C. The quantity $||u||_{TV(C)}$ can be used to estimate the image contours. If $||u||_{TV(C)}$ is small, it suggests that C is close a contour. The contour stencils strategy is to estimate the image contours by testing the TV along a set of candidate curves, the curves with small $||u||_{TV(C)}$ are then identified as approximate contours.

Contour stencils are a discretization of TV along contours. As described in [4, 5], a "contour stencil" is a function $\mathcal{S} : \mathbb{Z}^2 \times \mathbb{Z}^2 \to \mathbb{R}^+$ describing weighted edges between the pixels of v. Stencil $\mathcal{S}$ is applied to v at pixel $k \in \mathbb{Z}^2$ as

$$(\mathcal{S} \star [v])(k) := \sum_{(m,n) \in \mathbb{Z}^2} \mathcal{S}(m, n) |v_{k+m} - v_{k+n}|.$$

Defining $[v](m, n) := |v_m - v_n|$, this quantity is (with an abuse of notation) a cross-correlation over $\mathbb{Z}^2 \times \mathbb{Z}^2$ evaluated at (k, k). The stencil edges are used to approximate a curve C so that the quantity $(\mathcal{S} \star [v])(k)$ approximates $||u||_{TV(C+k)}$ (where $C + k := \{x + k : x \in C\}$). The image contours are estimated by finding a stencil with small TV. The best-fitting stencil at pixel k is

$$\mathcal{S}^\star(k) = \arg\min_{\mathcal{S} \in \Sigma} (\mathcal{S} \star [v])(k),$$

where Σ is a set of candidate stencils. It is possible that the minimizer is not unique, for example in a locally constant region of the image. For simplicity, we do not treat this situation specially and always choose a minimizer even if it is not unique. This best-fitting stencil $\mathcal{S}^\star(k)$ provides a model of the image contours in the neighborhood of pixel k. The stencils used in this work are shown in Figure 1.

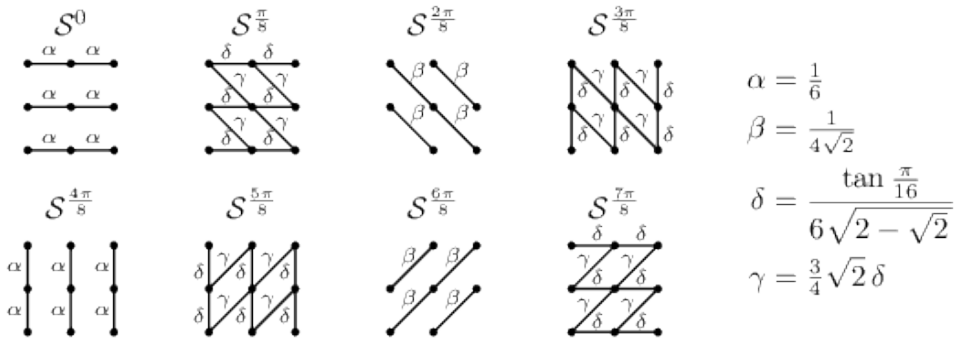

Figure 1: The proposed stencil set Σ. The edge weights $S(m, n)$ are denoted by superscript α, β, γ, δ.

For the set of candidate stencils Σ, we use 8 line-shaped stencils that were designed to distinguish between the functions

$$f^j(x) = x_1 \sin\frac{\pi}{8} j - x_2 \cos\frac{\pi}{8} j, \quad j = 0, \cdots, 7.$$

The edge weights α, β, γ, δ are selected so that on the function $f(x) = x_1 \sin\theta - x_2 \cos\theta$,

$$|\theta - \frac{\pi}{16} j| < \frac{\pi}{16} \quad \longrightarrow \quad \mathcal{S}^{\frac{\pi}{8} j} = \arg\min_{\mathcal{S} \in \Sigma} (\mathcal{S} \star [f]) \qquad \text{(see Figure 2)}.$$

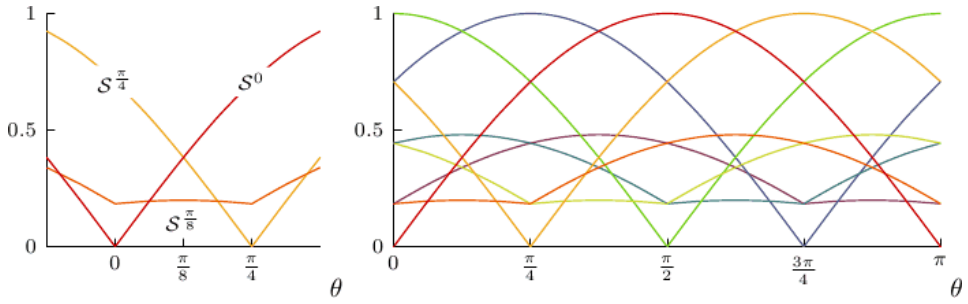

Figure 2: Normalized stencil total variations ($\mathcal{S}^{\frac{\pi}{8}j} \star [f]$) vs. θ. Left: The first three stencils, $j = 0, 1, 2$. Right: All eight stencils.

In this way, the stencils can fairly distinguish 8 different orientations. An estimate of the local contour orientation at point k is obtained by noting which stencil is the best-fitting stencil $\mathcal{S}^\star(k)$.

For a color image, the image is converted from RGB to a luma+chroma space

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 2 & 4 & 1 \\ -1 & -2 & 3 \\ 4 & -3 & -1 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

and the stencil TV is computed as the sum of $(\mathcal{S} \star [v])(k)$ applied to each color channel.

3 Interpolation

Given image v known on $\mathbb{Z}^2$, we seek to construct an image u on $\mathbb{R}^2$ such that

$$v(x) = (h * u)(x), \quad \text{for all } x \in \mathbb{Z}^2,$$

where h is the (assumed known) point spread function and $*$ denotes convolution.

The goal is to incorporate deconvolution yet maintain computational efficiency. To achieve this, the global operation of deconvolution is approximated as a local one, such that pixels only interact within a small window.

3.1 Local Reconstructions

For every pixel k in the input image, we begin by forming a local reconstruction

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

where $\mathcal{N} \subset \mathbb{Z}^2$ is a neighborhood of the origin and $\varphi_{\mathcal{S}^\star(k)}$ is a Gaussian oriented with the contour modeled by the best-fitting stencil $\mathcal{S}^\star(k)$ (see Figure 3).

The c_n are chosen such that u_k satisfies the discretization model locally,

$$(h * u_k)(m) = v_{k+m}, \quad m \in \mathcal{N}.$$

This condition implies that the c_n satisfy the linear system

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

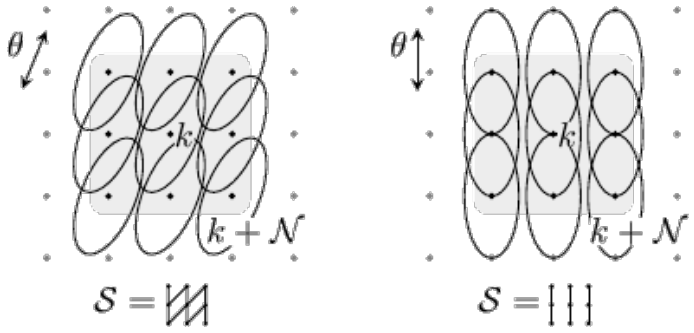

Figure 3: Local reconstruction u_k with oriented functions functions $\varphi_{\mathcal{S}^\star}$.

where $A_{\mathcal{S}}$ is a matrix with elements $(A_{\mathcal{S}})_{m,n} = (h * \varphi_{\mathcal{S}})(m-n)$. By defining the functions

$$\psi^n_{\mathcal{S}}(x) := \sum_{m\in\mathcal{N}} (A^{-1}_{\mathcal{S}})_{m,n}\varphi_{\mathcal{S}}(x-m),$$

u_k can be expressed directly in terms of the samples of v,

$$u_k(x) = v_k + \sum_{n\in\mathcal{N}} (v_{k+n} - v_k)\psi^n_{\mathcal{S}^\star(k)}(x).$$

3.2 Global Reconstruction

The u_k are combined with overlapping windows to produce the interpolated image,

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

The window should satisfy $\sum_k w(x-k) = 1$ for all $x \in \mathbb{R}^2$ and $w(k) = 0$ for $k \in \mathbb{Z}^2\backslash\mathcal{N}$.

3.3 Iterative Refinement

This global reconstruction satisfies the discretization model approximately, $\downarrow (h * u) \approx v$. The accuracy may be improved using the method of iterative refinement. Let $\mathcal{R}$ denote the global reconstruction formula

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

such that $u = \mathcal{R}v$ (where we consider $\mathcal{S}^\star$ as fixed parameters so that $\mathcal{R}$ is a linear operator). Then the deconvolution accuracy is improved by the iteration

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

Each iteration should reduce the residual in satisfying the discretization model,

$$r^i = v - \text{sample}(h * u^i).$$

The residual reduces quickly in practice, usually three or four iterations is sufficient for accurate results.

3.4 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,

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

- $\varphi_{\mathcal{S}}$ is an oriented Gaussian,

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

  with $\sigma_\tau = 1.2$ and $\sigma_\nu = 0.6$, and θ is the orientation modeled by $\mathcal{S}$,

and three iterations of iterative refinement are applied (one initial interpolation and two correction passes).

For sake of demonstration, the examples below use a PSF with a substantial amount of blur, $\sigma_h = 0.5$. The default value for σ_h is 0.35 in the online demo associated with this article[2], which better models the blurriness of typical images.

4 Algorithm

The interpolation is computationally efficient. We first consider the complexity without iterative refinement.

The matrices $A_{\mathcal{S}}^{-1}$ can be precomputed for each stencil $\mathcal{S} \in \Sigma$, allowing the c_n coefficients to be computed in $6|\mathcal{N}|^2 + 3|\mathcal{N}|$ operations per (color) input pixel. Furthermore, since w has compact support, u only depends on the small number of u_k where $w(x - k)$ is nonzero. Let W be a bound on the number of nonzero terms,

$$\#\{k \in \mathbb{Z}^2 : w(x - k) \neq 0\} \leq W, \quad \text{for all } x \in \mathbb{R}^2.$$

We suppose that W is $O(|\mathcal{N}|)$. Given the c_n, each evaluation of $u(x)$ costs $O(|\mathcal{N}|^2)$ operations. So for factor-d scaling, the total computational cost is $O(|\mathcal{N}|^2 d^2)$ operations per input pixel. For scaling by rational d, samples of w and $\psi_{\mathcal{S}}^n$ can also be precomputed, and scaling costs $6|\mathcal{N}|Wd^2$ operations per input pixel. For the settings used in the examples, this is $864d^2$ operations per input pixel.

With iterative refinement, the previous cost is multiplied by the number of steps and there is the additional cost of computing the residual. If h is quickly decaying, then it is accurately approximated by an FIR filter with $O(d^2)$ taps and the residual

$$r^i = v - \text{sample}(h * u^i)$$

can be computed in $O(d^2)$ operations per input pixel.

[2] https://doi.org/10.5201/ipol.2011.g_iics

5 Implementation

This software is distributed under the terms of the simplified BSD license. Please see the readme.html file or the online documentation[3] for details.

Implementation notes:

- Fixed-point arithmetic is used to accelerate the main computations.
- For efficiency in the correction passes of iterative refinement, the u_k for which $|v_k|$ is small are not added (so that they do not need to be computed),

$$u(x) = \sum_{k \in \mathbb{Z}^2 : |v_k| \geq T} w(x-k) u_k(x-k).$$

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.

6 Examples

In Figure 4 we perform an interpolation experiment to test the performance of the proposed interpolation strategy.

First, a high-resolution image u_0 is smoothed and downsampled by factor 4 to obtain a coarsened image $v = \downarrow (h * u_0)$ where h is a Gaussian with standard deviation 0.5 in units of input pixels, $\sigma_h = 0.5$. This amount of smoothing is somewhat weak anti-aliasing, so the input data is slightly aliased.

The value of σ_h should estimate the blurriness of the PSF used to sample the input image. It is better to underestimate σ_h rather than overestimate: if σ_h is smaller than the true standard deviation of the PSF, the result is merely blurrier, but using σ_h slightly too large creates ripple artifacts.

The method works well for $0 \leq \sigma_h \leq 0.7$. For σ_h above 0.7, the method produces visible ringing artifacts (even if the true PSF used to sample the input image has standard deviation σ_h). One could expect this effect, since there is no kind of regularization in the deconvolution. In the online demo, the default value for σ_h is 0.35, which reasonably models the blurriness of typical images.

Interpolation is then performed on v to produce u approximating the original image uo. The interpolation and the original image are compared with the peak signal-to-noise ratio (PSNR) and mean structural similarity (MSSIM) metrics (see Appendix to know how these are computed).

Table 6 shows the convergence of the residual $r^i = v - \downarrow (h * u)$ where the image intensity range is $[0, 1]$.

Iteration i	$\|\|r^i\|\|_\infty$
1	0.05409007
2	0.01677390
3	0.00661765

[3] https://doi.org/10.5201/ipol.2011.g_iics

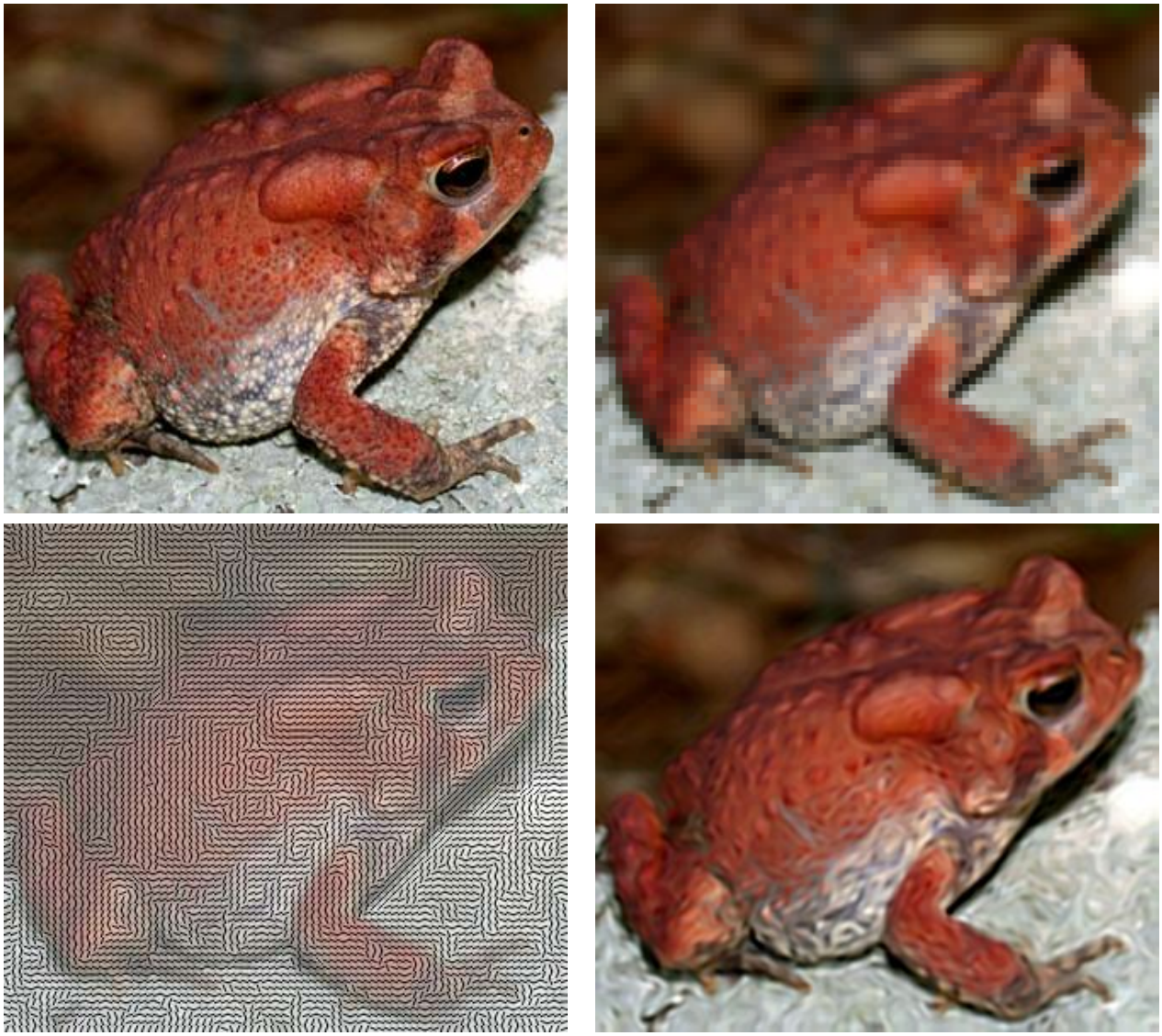

Figure 4: Interpolation experiment to test the performance of the proposed interpolation strategy. From left to right and from top to botton: original image (332×300), input image (83×75), estimated contour orientations, result of contour stencil interpolation (PSNR 25.77, MSSIM 0.7165, CPU time 0.109s).

6.1 Comparison with Other Methods

For comparison, the same experiment is performed with standard bicubic interpolation, Muresan's AQua-2 edge-directed interpolation[2], Genuine Fractals fractal zooming[4], Fourier zero-padding with deconvolution, Malgouyres' TV minimization [1], and Roussos and Maragos' tensor-driven diffusion [3]. The first three of these methods do not take advantage of knowledge about the point spread function, while the later three do (notice their sharper appearance). The results are shown in Figure 5.

The contour stencil interpolation has good quality similar to tensor-driven diffusion but with an order of magnitude lower computation time.

[4]onOne software. Genuine Fractals. www.ononesoftware.com

Figure 5: Interpolation results on the input image of Figure 4 for different methods. See text for details. From left to right and from top to botton: Bicubic (PSNR 24.36, MSSIM 0.6311, CPU time 0.012s), AQua-2 (PSNR 23.97, MSSIM 0.6062, CPU time 0.016s), Fractal Zooming (PSNR 24.50, MSSIM 0.6317), Fourier Zero-Padding with Deconvolution (PSNR 25.70, MSSIM 0.7104, CPU time 0.049s), TV Minimization (PSNR 25.87, MSSIM 0.7181, CPU Time 2.72s), Tensor-Driven Diffusion (PSNR 26.00, MSSIM 0.7297, CPU Time 5.11s).

6.2 Geometric Features

The experiment shown in Figure 6 on a synthetic image tests the method's ability to handle different geometric features.

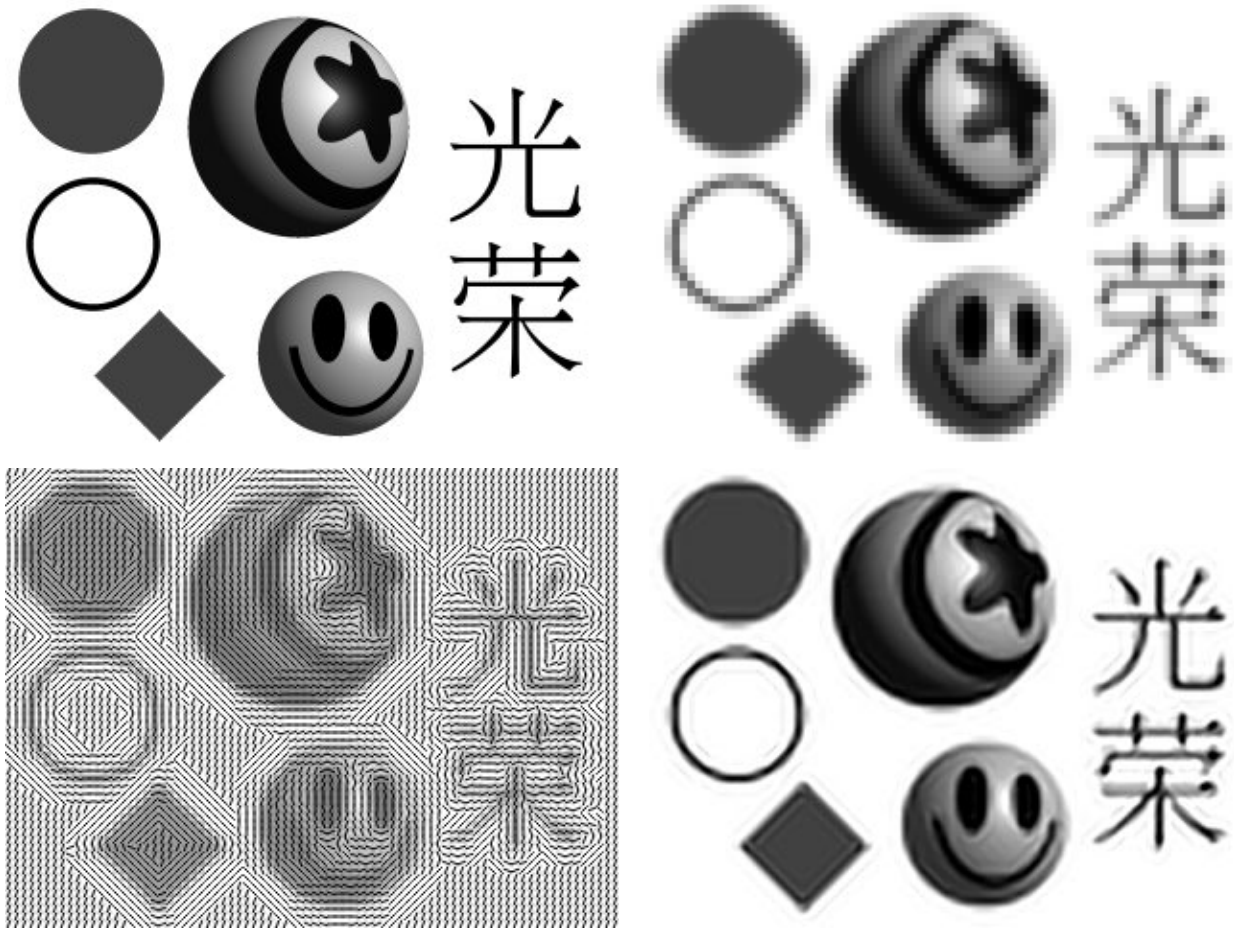

Figure 6: Interpolation experiment to test the method's ability to handle different geometric features. From left to right and from top to botton: original image (320×240), input image (80×60), estimated contour orientations, result of contour stencil interpolation (PSNR 21.23, MSSIM 0.8548, CPU time 0.078s).

6.3 Textures

Because the method is sensitive to the image contours, oriented textures like hair can be reconstructed to some extent. Interpolation of rough textures with turbulent contours is less successful. Some results on textured images are shown in Figure 7.

6.4 Noisy Images

A limitation of the method is the design assumption that noise in the input image is negligible. If noise is present, it is amplified by the deconvolution. The sensitivity to noise increases with the PSF standard deviation σ_h, which controls the deconvolution strength. Similarly, if σ_h is larger than the standard deviation of the true PSF that sampled the image, then the method produces significant oscillation artifacts because the deconvolution exaggerates the high frequencies. Figure 8 displays some interpolation results on noisy images.

Figure 7: Interpolation experiment to test the method's ability to handle textures. From top to botton: original images (left: 392×304; right: 332×304), input images (left: 98×76; right: 83×76), result of contour stencil interpolation (left: PSNR 33.24, MSSIM 0.7762, CPU time 0.129s; right: PSNR 22.47, MSSIM 0.6051, CPU time 0.115s).

Appendix. Image Quality Metrics.

6.5 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 $\{0, 1, \cdots, 255\}$. Several

Figure 8: Interpolation results on noisy images. The top row shows the input images, from left to right: clean input, JPEG compressed and quantized colors. The bottom row shows their interpolations, from left to right: PSNR 26.48, MSSIM 0.8196; PSNR 20.38, MSSIM 0.5244; PSNR 18.30, MSSIM 0.3393.

standard metrics can then be defined in terms of l^p norms.

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

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 PSNR $= \infty$ when $A = B$.

6.6 MSSIM

The mean structural similarity (MSSIM) index[5] is a somewhat more complicated metric designed to agree better with perceptual image quality.

[5] https://ece.uwaterloo.ca/~z70wang/research/ssim/

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

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

where $C1 = (0.01 \cdot 255)^2$ 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,

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

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

Acknowledgments

This material is based upon work supported by the National Science Foundation under Award No. DMS-1004694. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Work partially supported by the MISS project of Centre National d'Etudes Spatiales, the Office of Naval Research under grant N00014-97-1-0839 and by the European Research Council, advanced grant "Twelve labours".

Image Credits

 John D. Willson, USGS Amphibian Research and Monitoring Initiative[6]

 Pascal Getreuer

 Standard test image

 Gary Kramer, U.S. Fish and Wildlife Service[7]

References

[1] F. Malgouyres and F. Guichard, "Edge direction preserving image zooming: A mathematical and numerical analysis", SIAM Journal of Numerical Analysis, 39, pp. 1–37, 2002. http://dx.doi.org/10.1137/S0036142999362286

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

[6] http://armi.usgs.gov/gallery/detail.php?search=Genus&subsearch=Bufo&id=323
[7] https://www.fws.gov/home/wolfrecovery/

[3] A. Roussos and P. Maragos, "Reversible interpolation of vectorial images by an anisotropic diffusion-projection PDE", International Journal of Computer Vision, 84(2), pp. 130–145, 2008. http://dx.doi.org/10.1007/s11263-008-0132-x

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

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