A Unifying Resolution-Independent Formulation for Early Vision[*]

Fabio Viola
University of Cambridge
fabio@thefoundry.co.uk

Roberto Cipolla
University of Cambridge
cipolla@eng.cam.ac.uk

Andrew Fitzgibbon
Microsoft Research Cambridge
awf@microsoft.com

Abstract

We present a model for early vision tasks such as denoising, super-resolution, deblurring, and demosaicing. The model provides a resolution-independent representation of discrete images which admits a truly rotationally invariant prior. The model generalizes several existing approaches: variational methods, finite element methods, and discrete random fields. The primary contribution is a novel energy functional which has not previously been written down, which combines the discrete measurements from pixels with a continuous-domain world viewed through continous-domain point-spread functions. The value of the functional is that simple priors (such as total variation and generalizations) on the continous-domain world become realistic priors on the sampled images. We show that despite its apparent complexity, optimization of this model depends on just a few computational primitives, which although tedious to derive, can now be reused in many domains. We define a set of optimization algorithms which greatly overcome the apparent complexity of this model, and make possible its practical application. New experimental results include infinite-resolution upsampling, and a method for obtaining "subpixel superpixels".

1. Introduction

This paper argues for a new formulation of image reconstruction problems. The essential claim, motivated in figure 1, is that while image *data* is provided as spatially discrete samples, the interpretation and reconstruction of images should be in terms of functions defined over a spatially continuous world. The contributions of our work are:

1. a new energy functional (6) which generalizes a large class of existing models and allows solutions to be compared independent of the basis functions over which they are defined;
2. a set of algorithms for optimization of this functional over finite-element meshes with moving vertices, and a subpixel-accurate image partition ("superpixels") aligned with object boundaries;
3. because we are truly optimizing over functions, rather than fixed discretizations, we begin to find that *simple priors on the continuous model can replace complex priors on pixels*. Figure 2 illustrates how a simple total variation (TV)-like prior in 1D can perform as well as large patch priors. In 2D, a simple TV-like prior is inherently rotationally invariant, so that metrication artifacts are trivially avoided (fig 3);
4. a set of precomputed integrals and derivatives which will enable other researchers to easily adopt these techniques.

[*]This is the CVPR 2012 version with supplementary material included.

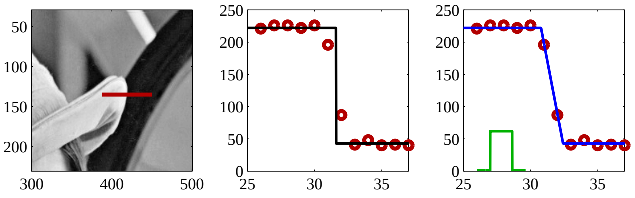


Figure 1. **The need for a spatially continuous world model**. The red circles are samples from a slice across the sharpest edge on a well known image. The image edge subtends about 2.5 pixels. Despite the hat's presumably slightly furry real-world boundary, it is almost certainly not blurred to the tune of 5mm, which is what would be needed to cause this level of image blur. An ideal step edge (black) is a poor model of the image edge, implying that discontinuity-preserving priors are poor models of this image. However, if the discontinuity-preserving prior is instead applied on a *latent* image of the spatially continuous world, and viewed through a blur kernel (green), the blue curve results, which is a good fit to the image data. Thus even when not attempting upsampling, the combination of "infinite resolution" world edges and an explicit blur kernel is a better model than discontinuity-preserving priors on the image.

To discuss related work, let us follow a standard example: combined denoising and (non-blind) upsampling. The input is a digital image $\mathcal{I}$, represented as a set of n pixels

$$\mathcal{I} = \{(I_i, \mathbf{x}_i)\}_{i=1}^{n}, \tag{1}$$

where pixel i comprises a location $\mathbf{x}_i$ and a scalar I_i, which ultimately represents some physical quantity such as a count of the number of photons which have fallen into a particular sensor bin in a given capture period. Define the vector of image samples $\mathbf{I} = [I_1, ..., I_n]^\top$. The task is to recover a latent image $\mathcal{U}$ such that the downsampled latent image is close to $\mathbf{I}$ and such that $\mathcal{U}$ is likely under some prior on natural images.

In the **discrete** approach [14, 5], the latent image is a vector $\mathbf{u} \in \mathbb{R}^N$ for $N > n$. It is found by minimizing an energy similar to

$$E(\mathbf{u}) = \left\|\mathbf{I} - \mathtt{F}\mathbf{u}\right\|_\rho + \sum_j \left\|\mathtt{D}_j \mathbf{u}\right\|_{\rho'} \tag{2}$$

where the i^{th} row of $\mathtt{F}$ encodes the blur kernel or point-spread function at pixel i, and $\mathtt{D}_j$ encodes the regularizer or prior term, with nonzero entries in $\mathtt{D}_j$ corresponding to the elements of the j^{th} *clique* or *neighbourhood*. The $\|\cdot\|_\rho$ norms are used to signify any nondescending function such as absolute value or a robust estimator. From this formulation we immediately see that the form of the prior term, encoded in the $\mathtt{D}_j$, is strongly dependent on the output resolution. However, methods using large patch dictionaries are known to produce excellent results albeit at sometimes tremendous computational cost [10]. Multiresolution computation is also complex [23], as the priors at different resolutions must be kept consistent. On the other hand, discrete formulations benefit from efficient algorithms which can make use of the grid structure on GPUs.

The **continuous** approach represents the latent image by a function $u(\mathbf{x})$ defined on a continuous image domain $\Omega \subset \mathbb{R}^2$, found by minimizing the functional

$$\mathcal{E}^{\text{cd}}(\underset{\sim}{u}) = \iint_\Omega \left\|I(\mathbf{x}) - u(\mathbf{x})\right\|_\rho \mathrm{d}\mathbf{x} + \mathcal{R}(\underset{\sim}{u}) \tag{3}$$

where $\mathcal{R}$ is a regularizer functional, to be discussed in §1.2, and where the undertilde notation refers to a function as an object. We note immediately that the blur kernel does not appear, and that the image is assumed to be supplied as a continuous function $I(\mathbf{x})$ which must somehow be assembled from the discrete samples $\mathcal{I}$. Ignoring these deficiencies, approaches to minimizing $\mathcal{E}^{\text{cd}}$ divide into two subclasses: variational approaches, and finite element methods.

Finite element approaches [19, 18, 17] explicitly parametrize $\underset{\sim}{u}$ as a linear combination of M basis functions:

$$u(\mathbf{x}) = \sum_{m=1}^{M} u_m \psi_m(\mathbf{x}) \tag{4}$$

so that the function $\underset{\sim}{u}$ is parameterized by the M-dimensional vector $\mathbf{u} = [u_1, ..., u_M]^\top$, and the integrals in $\mathcal{E}^{\text{cd}}$ become plain functions of $\mathbf{u}$ which can be optimized

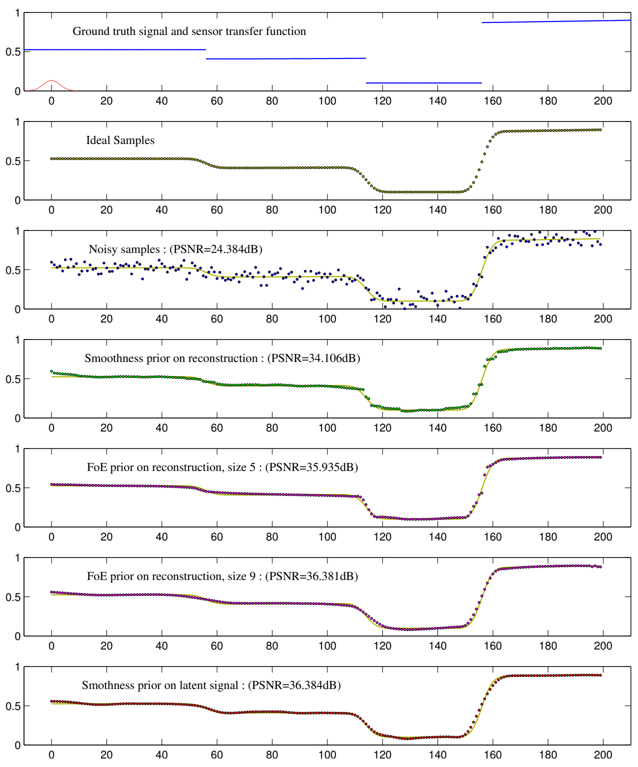


Figure 2. **Denoising as an upsampling problem.** Rows 1-3: we obtain noisy samples of a piecewise constant world through a point-spread function. Row 4: Simple TV-like prior on the discrete image at input resolution cannot model the PSF-blurred discontinuities. Row 5-6: A field of experts (FoE) prior yields a good reconstruction only for large filter sizes. Row 7: Simple TV-like prior on the resolution-independent latent image is equivalent to 9-wide FoE.

using standard methods. Finite element approaches are easily modified to allow multiresolution computation and have been applied on massive datasets [8]. While *refinement* of FE meshes is well known, few systems allow the user to *move vertices*, as in our work. Szeliski [18] shows how multiresolution approaches can be generalized using hierarchical preconditioners, but cannot deal with blur kernels other than the Dirac delta function.

Variational approaches [12, 22] apply variational calculus to obtain the Euler-Lagrange equations, a set of differential equations in $\underset{\sim}{u}$. This has the apparent advantage that in the common case where $\mathcal{R}$ involves ∇u, the energy is rotationally invariant. However, these methods then discretize the differential equations in order to compute a solution, with the same disadvantages as the discrete approach above. Indeed all variational methods we have encountered can be recast as a finite element formulation whose parametrization yields the same discretization. It is well known that there are transformations between discrete and continuous approaches [16] but such transformations, depending on the continuous function $I(\mathbf{x})$ lack the connection to the raw image samples that we now propose.

1.1. The new model

Our representation of the world is the continuous function $\underset{\sim}{u}$. Pixel measurements are generated through the action of a point-spread function. For pixel i, the kernel[1] is written $\kappa_i(\mathbf{x})$ and is, like $\underset{\sim}{u}$, a function from $\Omega \to \mathbb{R}^+$. A noise-free image sample $\mathring{I}_i$ is generated from the integral

$$\mathring{I}_i = \iint_\Omega \kappa_i(\mathbf{x}) u(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{5}$$

and the noisy pixel $I_i = \mathring{I}_i + \eta_i$ where η_i is a random variable representing imaging noise. The energy functional is then

$$\mathcal{E}(\underset{\sim}{u}) = \sum_{i=1}^{n} \left\| I_i - \iint_\Omega \kappa_i(\mathbf{x}) u(\mathbf{x}) \mathrm{d}\mathbf{x} \right\|_\rho + \mathcal{R}(\underset{\sim}{u}) \tag{6}$$

where $\mathcal{R}$ is the regularizer, detailed below. The energy combines discrete and continuous components: a sum over the n discrete pixel samples, and a continuous integral over spatial locations $\mathbf{x}$, and it is our claim that optimization of this discrete/continuous energy has previously been proposed only in strictly special cases. Work that comes close to minimizing (6) includes [20], who compute an integral over a piecewise-continuous grid representation of u, and [13], who compute the integral over a quadrilateral. Similarly, discrete superresolution techniques [3, §5.4], describe an area-sampling approximation to the integral, but none refine the underlying grid, or allow arbitrary boundary positions.

1.2. The regularizer

The regularizer $\mathcal{R}$ that we use is a generalization of the Mumford-Shah functional, and is the sum of a continuous HyperLaplacian [9] prior in the smooth areas of $\underset{\sim}{u}$, and a TV-like term on the discontinuities. If we define $\mathsf{J}(\underset{\sim}{u})$, the *jump set* of $\underset{\sim}{u}$, as the set of locations in Ω where ∇u is not finite, and we assume that $\mathsf{J}(\underset{\sim}{u})$ is a simple curve almost everywhere, then the regularizer is defined as

$$\begin{aligned}\mathcal{R}(\underset{\sim}{u}) = \lambda_1 & \iint_{\Omega \setminus \mathsf{J}(\underset{\sim}{u})} \|\nabla u(\mathbf{x})\|_p^\alpha \, \mathrm{d}\mathbf{x} + \\ & \lambda_2 \int_{\mathsf{J}(\underset{\sim}{u})} \left| u^l(\mathbf{x}) - u^r(\mathbf{x}) \right|^\beta \mathrm{d}\mathbf{x} + \\ & \qquad \lambda_3 \int_{\mathsf{J}(\underset{\sim}{u})} \left\| \nabla u^l(\mathbf{x}) - \nabla u^r(\mathbf{x}) \right\|_{p'}^{\beta'} \mathrm{d}\mathbf{x}\end{aligned} \tag{7}$$

[1] A common simplification is to assume a spatially invariant kernel so that $\kappa_i(\mathbf{x}) = \kappa_0(\mathbf{x} - \mathbf{x}_i)$, yielding an interpretation of (5) as convolution $\underset{\sim}{u}' = \underset{\sim}{\kappa}_0 * \underset{\sim}{u}$ followed by sampling $\mathring{I}_i = u'(\mathbf{x}_i)$, but this offers no real advantage in our framework, so we use the general model throughout.

where $u^l(\mathbf{x})$ at a discontinuity point $\mathbf{x}$ is the limit value of u as we approach $\mathbf{x}$ from the left along the normal to $\mathsf{J}(\underset{\sim}{u})$, and similarly u^r from the right. The second term is thus an integral along the discontinuity boundary of a function of the difference in intensities across the boundary. Because of the work of Ambrosio et al. [1, 2], we are assured that this prior is "sensible" in the sense of having a minimizer. Note that $\mathsf{J}(\underset{\sim}{u})$ is a function of $\underset{\sim}{u}$, so that the functional $\mathcal{R}$ is correctly expressed as a functional over $\underset{\sim}{u}$ alone. Figure 1 discusses the applicability of this "infinite resolution" prior on real-world images.

1.3. Finite element representation

Under the above definition (4) of $\underset{\sim}{u}$, the image formation integral (5) becomes linear in the parameter vector $\mathbf{u} = [u_1, ..., u_M]$, as follows:

$$\mathring{I}_i = \iint_\Omega \kappa_i(\mathbf{x}) \sum_{m=1}^{M} u_m \psi_m(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{8}$$

$$= \sum_{m=1}^{M} u_m \iint_\Omega \kappa_i(\mathbf{x}) \psi_m(\mathbf{x}) \tag{9}$$

$$=: \sum_{m=1}^{M} u_m f_{im}, \tag{10}$$

so we obtain a simple form for the energy (6):

$$E(\mathbf{u}) = \sum_i \left\| I_i - \sum_m u_m f_{im} \right\|_\rho + R(\mathbf{u}) \tag{11}$$

where R is the evaluation of the regularizer at $\underset{\sim}{u}(\mathbf{u})$. If $\|\cdot\|_\rho$ were just the squared Frobenius norm, and we were to ignore $R(\cdot)$, this would simply yield a linear least squares problem: by assembling the coefficients f_{im} into an $n \times m$ render matrix $\mathbf{F}$, we would obtain

$$E(\mathbf{u}) = \|\mathbf{I} - \mathbf{F}\mathbf{u}\|^2 \tag{12}$$

which is minimized using any linear solver. When the norm is not Frobenius, any of a number of standard methods can be used [9].

It is now straightforward to relate the new formulation and the standard discrete formulation. Defining as basis functions the box function around each pixel centre, that is $\psi_m(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_m\|_\infty < \frac{1}{2}$ means that the u_m simply parameterize pixel height (assuming a simple "box" PSF κ). Noticing that only the second term of $\mathcal{R}$ is nonzero anywhere, setting $\beta = 1$ gives a 4-connected total variation regularizer where a pixel of height u with North and East neighbours u^N, u^E contributes $|u_N - u| + |u_E - u|$ to the energy. What is more interesting is to see that an interpretation in terms of more complex regularizers can be found. For example, it is known that this simple prior suffers from metrication artefacts, and is sometimes replaced

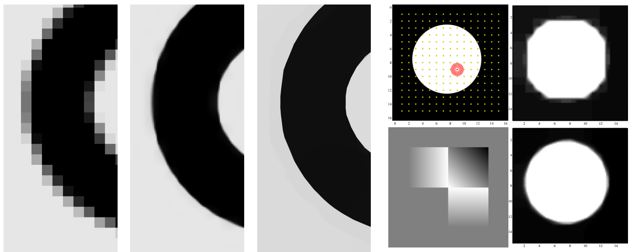


Figure 3. **Rotation invariance. Left**: (a) Low resolution input (after Fattal [5]). (b) 8x upsampling [5] showing different behaviour on vertical and slanted edges. (c) 128x upsampling using our method[3]. **Right**: (Top left) Disc, point-spread function and sample points; (Top right) Reconstruction showing metrication artifacts; (Lower left) "Three pin" basis function; (Lower right) "Three pin" (i.e. anisotropic TV) reconstruction.

by an "isotropic TV" term $\sqrt{|u_N - u|^2 + |u_E - u|^2}$ [4]. Figure 3 illustrates an unusual "3-pinned" planar basis function which yields exactly the isotropic TV energy.

So far, however, we have merely derived the new formulation and hinted as to how it can be related to a variety of existing models. The next sections show how to gain the real power of the method, by defining a second-order optimizer over a triangle mesh partition of the image, and introducing algorithms to optimize mesh vertices as well as the basis function weights.

2. The mesh model

Our model is defined by a set of V vertices, represented by a $2 \times V$ matrix $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_V]$, a set of edges where each edge is a 4-tuple of indices $e_j = (\nu_j^0, \nu_j^1, t_j^l, t_j^r)$, indexing the edge's start and end vertices, and left and right triangles respectively. The line segment corresponding to the edge, as a subset of $\mathbb{R}^2$, is written e_j. The number of triangles is T, triangles are indexed by t, and the triangle as a subset of $\mathbb{R}^2$ is written T_t. We define the characteristic function of each triangle $\chi_t(\mathbf{x})$ as

$$\chi_t(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \mathsf{T}_t \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

and again express $u(\mathbf{x})$ as a linear combination of basis functions modulated by χ_t, parameterized per triangle by a vector $\mathbf{u}_t = [u_{t1}, ..., u_{tK}]^\top \in \mathbb{R}^K$,

$$u(\mathbf{x}) = \sum_{t=1}^{T} \sum_{k=1}^{K} u_{tk} \phi_k(\mathbf{x}) \chi_t(\mathbf{x}) \tag{14}$$

[3]One might think that when comparing to existing algorithms, we should use the same upsampling factors. Prima facie this is true, but as our algorithm in fact upsamples to "infinite resolution", we believe we should compare the N-fold upsampling of existing methods to our resolution-independent representation. The problem is that the latter can't be rendered reliably in PDF, so in the end we chose to simply render a bitmap at 128x, as a "near-infinite" scale factor. No existing algorithm can get to that resolution in a reasonable time, so we ended up comparing between 8x and 128x.

For typical applications, a piecewise linear model is appropriate, so $K = 3$ and

$$\phi_1(x, y) = 1 \tag{15}$$
$$\phi_2(x, y) = x \tag{16}$$
$$\phi_3(x, y) = y. \tag{17}$$

This allows u to have discontinuities on every edge in the mesh, so is a richer model than would be obtained by linearly interpolating per-vertex values. In certain applications, such as image vectorization, it might be useful to use a different set of basis functions as discussed in §6.

The elements of the render matrix f_{im} are now integrals of the form in (9), i.e. of the blur kernel multiplied by $\psi_m = \phi_k \chi_t$. Note that we associate indices m with pairs tk, e.g. via $m = (t-1)K + k$. Our model of the blur kernel is a sum of constant functions defined within arbitrary convex polygons, which allows the integral to be computed very simply. In the the simplest case, illustrated in figure 4, the kernel for pixel i is defined as constant inside a polygon Q_i, i.e.

$$\kappa_i(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \mathsf{Q}_i \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

so that the integral to evaluate is

$$f_{itk} = \iint_\Omega \kappa_i(\mathbf{x}) \phi_k(\mathbf{x}) \chi_t(\mathbf{x}) d\mathbf{x} = \iint_{\mathsf{T}_t \cap \mathsf{Q}_i} \phi_k(\mathbf{x}) d\mathbf{x}, \tag{19}$$

the integral of the basis function ϕ_k inside the convex polygon $\mathsf{P} = \mathsf{T}_t \cap \mathsf{Q}_i$, whose vertices are easily computed using convex polygon intersection [11]. For ϕ_1 the result is simply the area of P. For other k we can use Green's theorem to greatly simplify evaluation of the region integral, e.g. for $k = 2$, we have $\phi_2(x, y) = x$, and

$$\iint_\mathsf{P} \phi_2(\mathbf{x}) d\mathbf{x} = \iint_\mathsf{P} x \, dxdy = \int_{\partial \mathsf{P}} \frac{1}{2} x^2 dy \tag{20}$$

which may be broken into a sum of contributions along each segment of P. More details are supplied in the supplementary material.

2.1. Computing the regularizer term

The second item to be computed is the regularizer (7). Again given $\mathbf{V}$, the integral becomes a sum over triangles

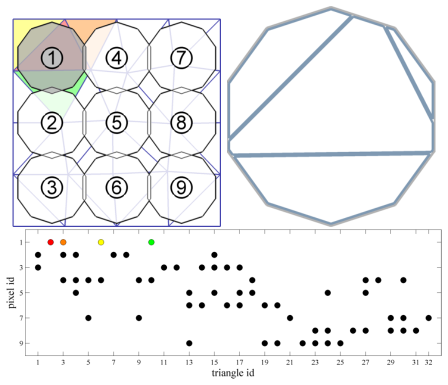


Figure 4. **The imaging model.** (a) Example mesh and blur kernels for a 3×3 pixel image. The blur kernel $\kappa_i(\mathbf{x})$ for each pixel i is 1 inside the decagon Q_i and 0 outside. (b) The integral for pixel 1 is $\iint \kappa_1(\mathbf{x})\mathbf{u}_t^\top \phi(\mathbf{x})\mathrm{d}\mathbf{x}$, computed as the sum over all triangles t which intersect Q_i. The integral is easily computed by convex polygon clipping, then applying Green's theorem on the returned boundary (20). Note that the pixels at the image edge need not be specially treated: triangles which intersect any pixel contribute to the integral, and those which intersect no pixel will be filled by the prior. Second row: the render matrix $\mathbf{F}$ for this example.

and edges:

$$R(\mathbf{U}) = \lambda_1 \sum_{t=1}^{T} \iint_{\mathsf{T}_t} \left\| \sum_k u_{tk} \nabla\phi_k(\mathbf{x}) \right\|_p^\alpha \mathrm{d}\mathbf{x} + \\ + \lambda_2 \sum_{j=1}^{n_\text{edges}} \int_{\mathsf{e}_j} \left| \sum_k (u_{t_j^l,k} - u_{t_j^r,k})\phi_k(\mathbf{x}) \right|^\beta \mathrm{d}\mathbf{x} \\ + \lambda_3 \sum_{j=1}^{n_\text{edges}} \int_{\mathsf{e}_j} \| \sum_k (u_{t_j^l,k} - u_{t_j^r,k}) \nabla\phi_k(\mathbf{x}) \|_{p'}^{\beta'} \mathrm{d}\mathbf{x} \quad (21)$$

For linear elements the first integrand is independent of $\mathbf{x}$ which means the first summation becomes

$$\sum_t \#\mathsf{T}_t \left\| \begin{pmatrix} u_{t2} \\ u_{t3} \end{pmatrix} \right\|_p^\alpha, \quad \text{where } \#\mathsf{T}_t \text{ is the area of } \mathsf{T}_t \quad (22)$$

which although in general a nonlinear function of $\mathbf{U}$, is one for which derivatives are straightforward to compute. The special case where $p = \alpha = 2$, is particularly simple, giving a quadratic form in $\mathbf{U}$.

The second term is rather involved, but can be computed in closed form (see supplementary material). Differences of elements of $\mathbf{U}$ appear raised to powers of $\beta + 1$, and again derivatives are straightforward. Similarly, the start and end vertices of the edge appear, raised to powers of $\beta + 1$. We write the result in terms of overloaded functions R^disc as

$$R^\text{disc}(\mathbf{U}, \mathbf{V}) = \sum_{j=1}^{n_\text{edges}} R^\text{disc}(\mathbf{u}_{t_j^l} - \mathbf{u}_{t_j^r}, \mathbf{v}_{\nu_j^0}, \mathbf{v}_{\nu_j^1}) \quad (23)$$

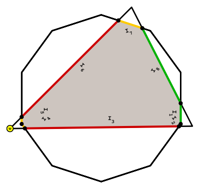


Figure 5. **Derivatives with respect to vertex positions.** We must compute the derivative with respect to $\mathbf{v}$ of an integral computed over the grey intersection polygon P. The integral is computed over line segments, giving three cases: on red segments, both endpoints are a function of $\mathbf{v}$, on yellow segments one endpoint, and on green none.

2.2. Derivatives with respect to vertex positions

To summarize the preceding sections, $E(\mathbf{U})$ is a relatively well behaved (and for some parameter settings, convex) function of $\mathbf{U}$. However, the real power of the mesh model is not seen until optimization over the mesh vertices is introduced. The strategy is ostensibly straightforward: for fixed $\mathbf{U}$, compute the derivatives of E with respect to vertex positions, and then perform a line search. Let us first write E with the dependencies on $\mathbf{V}$ made clear:

$$E(\mathbf{U}, \mathbf{V}) = \left\| \mathbf{I} - \mathbf{F}(\mathbf{V})\mathbf{U}_: \right\|_\rho + \\ \lambda_1 \sum_t \#\mathsf{T}_t(\mathbf{V}) \left\| \begin{pmatrix} u_{t2} \\ u_{t3} \end{pmatrix} \right\|_p^\alpha + \lambda_2 R^\text{disc}(\mathbf{U}, \mathbf{V}) \quad (24)$$

Computing derivatives with respect to $\mathbf{V}$ is easy for R^disc, and for the triangle areas $\#\mathsf{T}_t$ as the vertices simply appear analytically in those expressions. The more interesting terms are the derivatives of $\mathbf{F}$ where the computational steps comprise first a polygon clipping, and then a moment computation (19). The strategy is illustrated in figure 5. For example, to compute $\frac{\mathrm{d}f_{itk}}{\mathrm{d}\mathbf{v}_\nu}$, the intersection polygon $\mathsf{T}_t \cap \mathsf{Q}_i$ is computed, and for each line segment in the intersection, it is determined whether perturbation of $\mathbf{v}_\nu$ affects zero, one, or both endpoints, and the derivative of that segment's contribution is accumulated.

3. Optimization

Our optimization strategy interleaves a number of high-level processes:

- Optimization over $\mathbf{U}$ using quasi-Newton method.
- Optimization over $\mathbf{V}$ using analytic block-diagonal Hessian. This requires some interesting strategies to make progress in the presence of "sliver" triangles, and is described in the supplementary material.
- Edge optimization via global "N/Z" flips.
- Mesh refinement

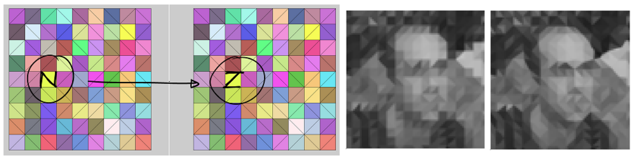


Figure 6. **Global "N/Z" flips.** The triangle mesh is grouped into triangle pairs, and within each pair, a boolean flag indicates whether to perform an edge swap. Global optimization of the boolean flags can discover useful image structure.

The main optimization loop is an alternation over $\mathbf{U}$ and $\mathbf{V}$, with NZ flips and refinement when the alternation convergence rate drops. Each process optimizes the same global energy, so that there is no difficulty in interleaving them arbitrarily. The optimization may also begin with a piecewise constant model, which tends to position the image edges well, and then on convergence initialize a piecewise linear model, which improves reconstruction in smooth areas.

3.1. Global "N/Z" flips

Local mesh moves are effective at positioning object boundaries when edges are close to correctly oriented. However when meshes are initialized at a coarse scale, such alignment is difficult to ensure. The N/Z flip process takes a triangle mesh, and greedily creates pairs of triangles which share an edge, with each triangle in at most one pair. Some triangles may not be paired without causing problems. In each pair, an alternative configuration is considered, whereby the edge is swapped to traverse the other two vertices. The data term of the energy is easily computed for the new pair, as is the prior contribution on the internal edge. The prior contribution on the external edges can be computed as a function of the boolean flag in an adjacent pair, leading to a quadratic boolean optimization problem, which, although not submodular, can be solved via a variety of modern techniques [15]. Figure 6 illustrates the process.

4. Extensions to the basic model

The model as described deals only with grayscale images, and does not deal with varying noise levels or missing data.

Colour is most simply handled for reconstruction problems by performing reconstruction on the luminance image, and interpolating the colour channels via some ad hoc method. However, we are interested in obtaining a representation which treats colour correctly, having each triangle parameterize a colour triple rather than a scalar. This can be implemented using vector-valued basis functions, so that $\phi_k(\mathbf{x})$ maps to $\mathbb{R}^3$ rather than $\mathbb{R}$, and adapting the discontinuity term in the prior (7), for example by replacing the absolute value by an appropriate norm. If that norm is the 1-norm, the model behaves as the sum of three separate models, which means that boundaries are not necessarily coherent across channels. An alternative method of linking the channels is to parameterize the latent image colour as YCrCb, and impose the prior only on the Y channel.

At the sensor side, colour appears in two ways: the image samples I_i may be true colour (e.g. from a 3-CCD camera), in which case I_i might be considered an element of a colorspace $\mathcal{C}$, for example RGB, and be treated as a vector in $\mathbb{R}^3$; alternatively the image samples may be through a Bayer filter. When each pixel sample is one of R,G,B, the mixing coefficients appear in the data term. If I_i is the sample count behind a red filter, its energy contribution is

$$\left\| I_i - \gamma_4^r - (\gamma_1^r \mathbf{F}_Y + \gamma_2^r \mathbf{F}_{\text{Cr}} + \gamma_3^r \mathbf{F}_{\text{Cb}}) \mathbf{U}_{:} \right\|_\rho \tag{25}$$

where $\boldsymbol{\gamma}^r$ is the first row of the YCbCr-to-RGB conversion matrix.

The above exposition has talked mostly about blur kernels which are constant inside a specified polygon Q, so we stress again that kernels which are the sum of such kernels may also be implemented without any new derivations. On the other hand, if piecewise linear or other kernels were required, one would need to be able to compute the integrals of $\kappa(\mathbf{x})\phi_k(\mathbf{x})$ over polygonal regions. For any polynomial kernel this is easy; for something like a Gaussian, it is more difficult, but of course real-world PSFs are not Gaussian.

5. Applications

Images in this section are duplicated in the supplementary material.

The most obvious application of this approach is in super-resolution. The approach is simple: compute the latent image, essentially an infinite-resolution upsampling, and then re-render using a render matrix corresponding to a narrow PSF at the target resolution. It is a strength of our model that enormous upsampling ratios can be obtained easily, while some example-based methods [7] require repeated application of the pixel-based method, with a loss in quality.

An important parameter is the selection of the blur kernel, which encapsulates the prior knowledge one has about the low-resolution image source. For the experiments in figure 7 we used the known downsampling kernel. For the more interesting case in 8, we modelled the PSF at each pixel as a linear combination of two fixed PSFs. The fixed PSFs were estimated by selecting one edge on the chip, and one on the blurry background, and then choosing the PSF width which optimized reconstruction error using a fixed model with a small number of triangles aligned to the selected edge.

As a denoising experiment, we took the heavy noise example from [10] and fitted a latent image to it. While one might expect the algorithm to perform only slightly better than TV, it is, in terms of PSNR, among the better performers (figure 9). Visually, it is probably fairest to say that the

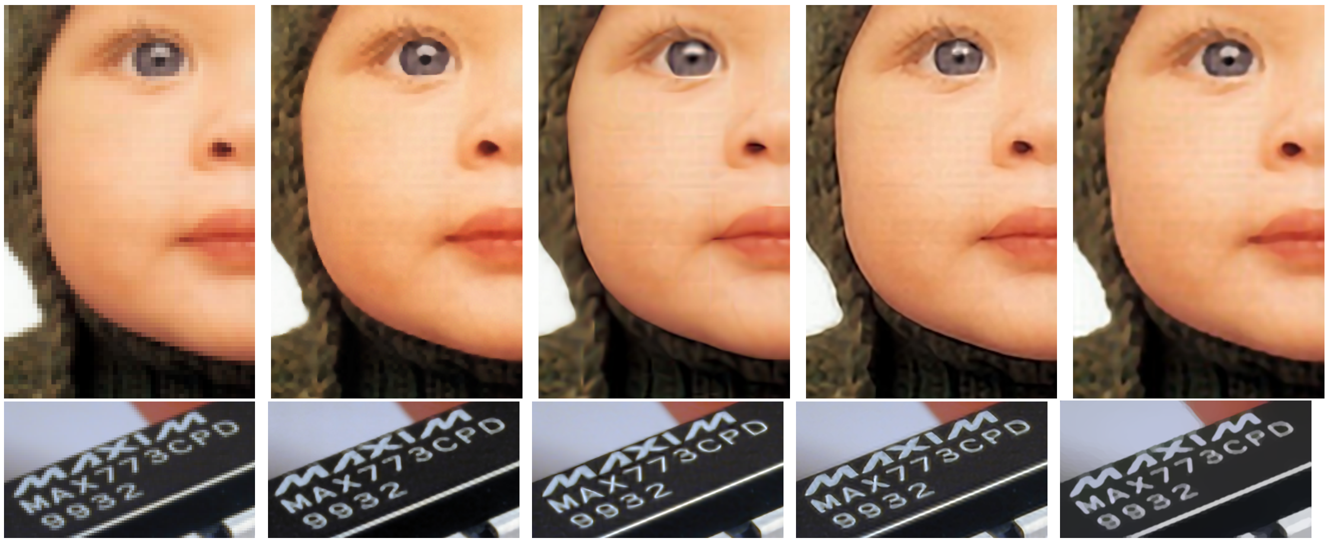


Figure 7. **Upsampling** (a) Input data. (b) Fattal [5]. (c) Freedman [6]. (d) Glasner [7]. (e) Ours. On natural images, performance is visually similar to Fattal [5]. On man-made images, performance arguably exceeds the other algorithms.

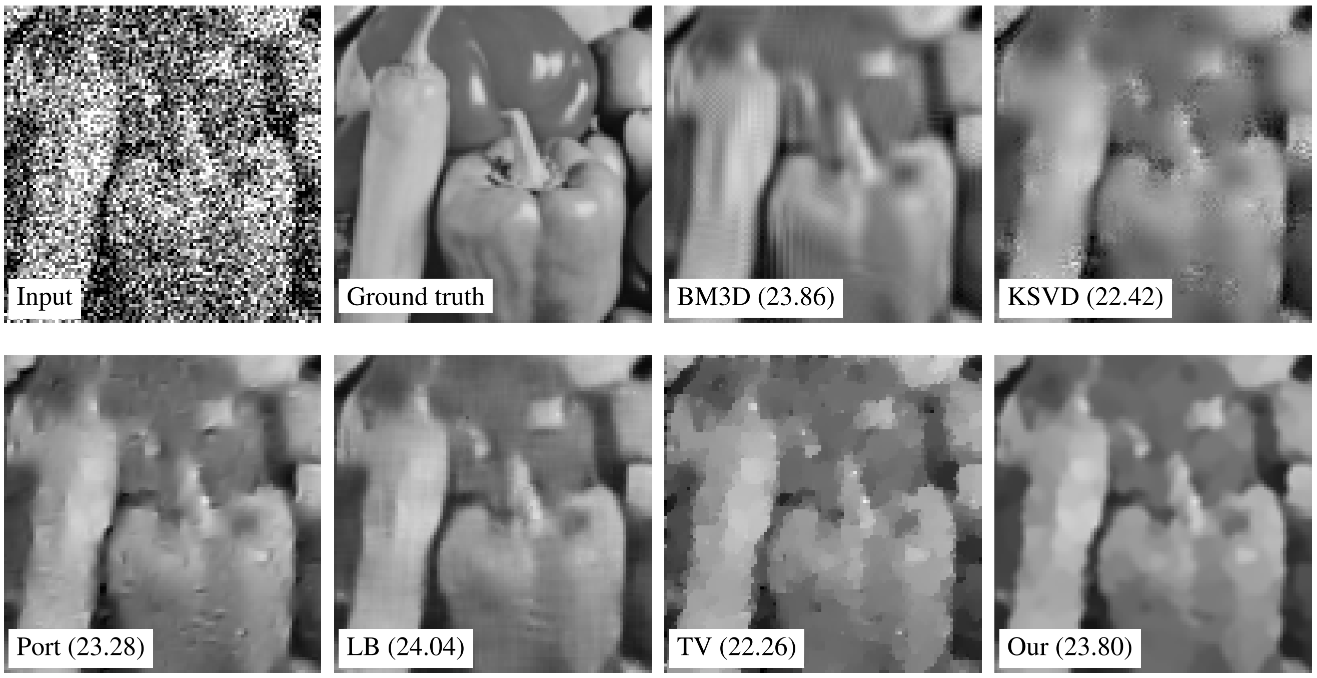


Figure 9. **Denoising.** Algorithms with PSNR. Our method, with TV-like prior, is competitive in PSNR with BM3D, and has "different" artifacts.

artifacts are less pleasant than LB [10], but the improvement over TV appears clear.

For many image processing tasks, the ultimate consumer is willing to interact with the process to produce better results. We simulated such a scenario in figure 10, where the user draws lines on an initial reconstruction, and polygons on those lines are constrained to have a given colour. This is easily implemented as a constrained optimization.

6. Conclusions

We have presented, we believe for the first time, a near-complete account of image modelling using a resolution-independent mesh representation. In particular, the ability to adapt mesh vertex positions on an arbitrary subpixel grid has not previously been demonstrated in the context of image processing.

Currently most examples require several minutes of

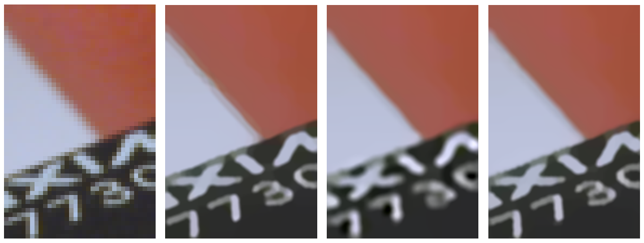


Figure 8. **Spatially varying PSF.** (a) Input data. (b) Reconstruction with narrow PSF. (c) Reconstruction with broad PSF. (d) Reconstruction with PSF width determined as a linear combination of (b),(c).

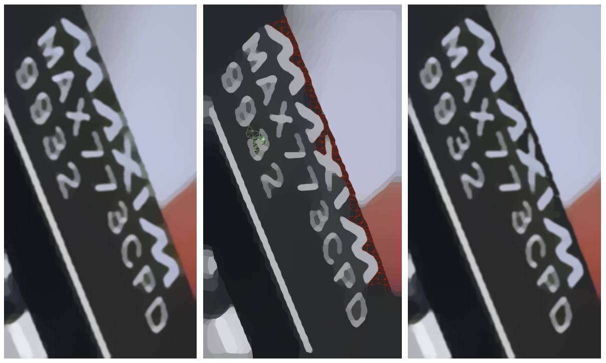


Figure 10. **User interaction.** (a) Reconstruction without user edits. (b) Constrained polygons. (c) Reconstruction with user interaction

CPU, but we do not believe this is representative of ultimate performance. Our implementation has been optimized for speed at an algorithmic level by use of second order optimizers, and by limiting the number of polygon clippings required, but has not been micro-optimized. We would hope to be not worse than a small factor of grid-graph performance for most tasks.

Although we have not demonstrated it here, we expect the model to be of particular use in modelling intrinsically sub-pixel entities such as matte boundaries (using a sub-pixel binary matte rather than the approximation in alpha matting), optical flow, and depth discontinuities in stereo.

As mentioned above, the prior is currently limited to a purely geometric prior on real-world discontinuities. It would be interesting to see if texture priors could be introduced using more wavelet-like basis functions. For example, instead of $[1, x, y]$ one might consider $[1, \sin(x), \cos(x), \sin(y), \cos(y), \sin(2x), \sin(2y), ...]$ and then impose different prior weights on the different bases.

A. Integrals

At a number of points we need to compute the integral of some function over a polygonal region P. For example, for the basis function set

$$\boldsymbol{\phi}(x, y) = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{26}$$

we must compute

$$\iint_{\mathsf{P}} x \, \mathrm{d}x\mathrm{d}y \tag{27}$$

$$\iint_{\mathsf{P}} y \, \mathrm{d}x\mathrm{d}y \tag{28}$$

$$\iint_{\mathsf{P}} 1 \, \mathrm{d}x\mathrm{d}y \tag{29}$$

Taking the first of these, we apply Green's theorem (stated in Box A). To do so, we should find a vector function $\mathbf{f}(x, y)$ such that $\operatorname{div} \mathbf{f} = x$. Of course, many such functions exist; an obvious choice is

$$\mathbf{f}(x, y) = \begin{pmatrix} \frac{1}{2}x^2 \\ 0 \end{pmatrix}. \tag{30}$$

Thus (27) becomes an integral along the polygon P, which itself becomes a sum over the linear segments of the polygon. Let one such segment be $(x_1, y_1) \to (x_2, y_2)$. Parameterizing $\mathbf{p}(t) = (x_1, y_1) + t(\Delta_x, \Delta_y)$, with $\Delta_x = x_2 - x_1, \Delta_y = y_2 - y_1$ gives

$$\mathbf{f}(\mathbf{p}(t)) = \begin{pmatrix} \frac{1}{2}(x_1 + t\Delta_x)^2 \\ 0 \end{pmatrix} \qquad \mathbf{n}(t) = \begin{pmatrix} -\Delta_y \\ \Delta_x \end{pmatrix}$$

so that the right hand side of (34) is

$$\int_0^1 \frac{1}{2}(x_1 + t\Delta_x)^2 \Delta_y \mathrm{d}t = \Delta_y(x_1 x_2 + \frac{1}{3}\Delta_x^2) \tag{31}$$

giving the expansion of (27) as

$$\iint_{\mathsf{P}} x = \sum_{i=0}^{n_{\mathsf{P}}} (y_{i+1} - y_i)(x_i x_{i+1} + \frac{1}{3}(x_{i+1} - x_i)^2) \tag{32}$$

where n_{P} is the number of segments in P, and point 0 is assumed identified with point n_{P}.

B. Rdisc

The derivation of R^{disc} from (7) is as follows.

$$\int_{\mathsf{e}_j} \left| u^l(\mathbf{x}) - u^r(\mathbf{x}) \right|^\beta \tag{35}$$

$$= \int_{\mathsf{e}_j} \left| \mathbf{u}_{t_j^l} \cdot \boldsymbol{\phi}(\mathbf{x}) - \mathbf{u}_{t_j^r} \cdot \boldsymbol{\phi}(\mathbf{x}) \right|^\beta \tag{36}$$

$$= \int_{\mathsf{e}_j} \left| (\mathbf{u}_{t_j^l} - \mathbf{u}_{t_j^r}) \cdot \boldsymbol{\phi}(\mathbf{x}) \right|^\beta \tag{37}$$

Box 1: Green's theorem

We consider only the special case where the boundary of P can be represented by a parametric curve. Let that curve be

$$\partial P = \{\mathbf{p}(t) \mid 0 \le t < 1\} \tag{33}$$

where $\mathbf{p}(t) := [p_x(t), p_y(t)]$.
Given a vector function $\mathbf{f} : \mathbb{R}^2 \mapsto \mathbb{R}^2$, Green's theorem states that

$$\iint_{\mathsf{P}} (\operatorname{div} \mathbf{f})\, \mathrm{d}A = \int_0^1 \mathbf{f}(\mathbf{p}(t))^\top \mathbf{n}(t) \mathrm{d}t \tag{34}$$

where $\operatorname{div} \mathbf{f} = \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y}$ and the curve normal $\mathbf{n}(t)$ is defined as $\mathbf{n}(t) = [-\frac{\mathrm{d}}{\mathrm{d}t}p_y(t), \frac{\mathrm{d}}{\mathrm{d}t}p_x(t)]^\top$

Let $\mathbf{\Delta}_j = \mathbf{u}_{t^l_j} - \mathbf{u}_{t^r_j}$, and parameterize the integral along e_j as $\mathbf{x}(t) = t\mathbf{v}_1 + (1-t)\mathbf{v}_2$. Then, for linear elements, $\boldsymbol{\phi}(\mathbf{x}) = [1; \mathbf{x}] =: \bar{\mathbf{x}}$, and the integral is

$$R^{\text{disc}}(\mathbf{\Delta}, \bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2) = \tag{38}$$

$$= \int_0^1 |t\mathbf{\Delta}.\bar{\mathbf{v}}_1 + (1-t)\mathbf{\Delta}.\bar{\mathbf{v}}_2|^\beta \ \mathrm{d}t \tag{39}$$

$$=: \int_0^1 |tA + (1-t)B|^\beta \ \mathrm{d}t \tag{40}$$

$$= \frac{1}{(B-A)(\beta+1)} \left[|B|^\beta B - |A|^\beta A\right] \tag{41}$$

$$= \frac{\left[|\mathbf{\Delta}.\bar{\mathbf{v}}_2|^\beta \mathbf{\Delta}.\bar{\mathbf{v}}_2 - |\mathbf{\Delta}.\bar{\mathbf{v}}_1|^\beta \mathbf{\Delta}.\bar{\mathbf{v}}_1\right]}{\mathbf{\Delta}.(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1)(\beta+1)} \tag{42}$$

$$= \frac{\mathbf{\Delta}.(|\mathbf{\Delta}.\bar{\mathbf{v}}_2|^\beta \bar{\mathbf{v}}_2 - |\mathbf{\Delta}.\bar{\mathbf{v}}_1|^\beta \bar{\mathbf{v}}_1)}{\mathbf{\Delta}.(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1)(\beta+1)} \tag{43}$$

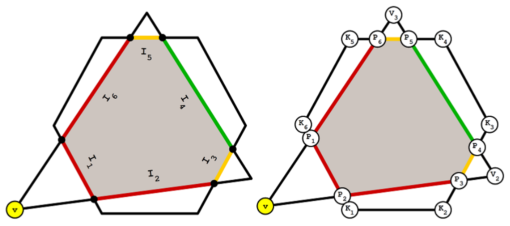
Note that if $\mathbf{\Delta}.(\bar{\mathbf{v}}_2 - \bar{\mathbf{v}}_1) = 0$, the first line is simply $\int_0^1 |\mathbf{\Delta}.\bar{\mathbf{v}}_2|^\beta \,\mathrm{d}t = |\mathbf{\Delta}.\bar{\mathbf{v}}_2|^\beta = |\mathbf{\Delta}.\bar{\mathbf{v}}_1|^\beta$.

C. Derivatives

Although the paper is accompanied by code to perform the various derivative calculations, is is valuable to detail them here, largely to show that although somewhat tedious, they are not much more demanding than conventional approaches. The first key term is the derivative of $\mathbf{F}(\mathbf{V})$, which contains individual terms

$$\frac{\partial f_{itk}}{\partial \mathbf{v}_j} = \frac{\partial}{\partial \mathbf{v}_j} \iint_{\mathsf{T}_t \cap \mathsf{Q}_i} \phi_k(\mathbf{x}) \mathrm{d}\mathbf{x}, \tag{44}$$

This will be nonzero only for triangles t which intersect Q_i. For triangles which do intersect, the derivative calculation for each vertex is identical up to index bookkeeping, so is illustrated for one vertex in figure 11.



Define:

```
function ∩(A, B, C, X) = intersect(line(A, B), line(C, X)), see (4.3)
function ∇∩(A, B, C, X) = $\frac{\partial}{\partial X}$ ∩ (A, B, C, X)
function IntFun(P, Q) = integral function, like (4.11), (4.12), (4.13)
function GradIntFun(P, Q) = [$\frac{\partial}{\partial P}$IntFun; $\frac{\partial}{\partial Q}$IntFun]
```

Compute:

function $f_{itk}(\mathbf{v} : \mathbb{R}^2) : \mathbb{R}$	**function** Grad$f_{itk}(\mathbf{v} : \mathbb{R}^2) : \mathbb{R}^2$
Vec2 $\mathtt{P}_1 = \cap(\mathtt{K}_6, \mathtt{K}_1, \mathtt{V}_3, \mathbf{v})$	Mat2x2 $\nabla\mathtt{P}_1 = \nabla\cap(\mathtt{K}_6, \mathtt{K}_1, \mathtt{V}_3, \mathbf{v})$
Vec2 $\mathtt{P}_2 = \cap(\mathtt{K}_6, \mathtt{K}_1, \mathtt{V}_2, \mathbf{v})$	Mat2x2 $\nabla\mathtt{P}_2 = \nabla\cap(\mathtt{K}_6, \mathtt{K}_1, \mathtt{V}_2, \mathbf{v})$
Vec2 $\mathtt{P}_3 = \cap(\mathtt{K}_2, \mathtt{K}_3, \mathtt{V}_2, \mathbf{v})$	Mat2x2 $\nabla\mathtt{P}_3 = \nabla\cap(\mathtt{K}_2, \mathtt{K}_3, \mathtt{V}_2, \mathbf{v})$
Vec2 $\mathtt{P}_4 = \cap(\mathtt{K}_2, \mathtt{K}_3, \mathtt{V}_2, \mathtt{V}_3)$	Mat2x2 $\nabla\mathtt{P}_4 = \mathbf{0}_{2\times 2}$
Vec2 $\mathtt{P}_5 = \cap(\mathtt{K}_4, \mathtt{K}_5, \mathtt{V}_2, \mathtt{V}_3)$	Mat2x2 $\nabla\mathtt{P}_5 = \mathbf{0}_{2\times 2}$
Vec2 $\mathtt{P}_6 = \cap(\mathtt{K}_4, \mathtt{K}_5, \mathtt{V}_3, \mathbf{v})$	Mat2x2 $\nabla\mathtt{P}_6 = \nabla\cap(\mathtt{K}_4, \mathtt{K}_5, \mathtt{V}_3, \mathbf{v})$
double $\mathtt{I}_1 = \text{IntFun}(\mathtt{P}_1, \mathtt{P}_2)$	Vec2 $\nabla\mathtt{I}_1 = [\nabla\mathtt{P}_1; \nabla\mathtt{P}_2]^\top \text{GradIntFun}(\mathtt{P}_1, \mathtt{P}_2)$
double $\mathtt{I}_2 = \text{IntFun}(\mathtt{P}_2, \mathtt{P}_3)$	Vec2 $\nabla\mathtt{I}_2 = [\nabla\mathtt{P}_2; \nabla\mathtt{P}_3]^\top \text{GradIntFun}(\mathtt{P}_2, \mathtt{P}_3)$
double $\mathtt{I}_3 = \text{IntFun}(\mathtt{P}_3, \mathtt{P}_4)$	Vec2 $\nabla\mathtt{I}_1 = [\nabla\mathtt{P}_3; \nabla\mathtt{P}_4]^\top \text{GradIntFun}(\mathtt{P}_3, \mathtt{P}_4)$
double $\mathtt{I}_4 = \text{IntFun}(\mathtt{P}_4, \mathtt{P}_5)$	Vec2 $\nabla\mathtt{I}_4 = [\nabla\mathtt{P}_4; \nabla\mathtt{P}_5]^\top \text{GradIntFun}(\mathtt{P}_4, \mathtt{P}_5)$
double $\mathtt{I}_5 = \text{IntFun}(\mathtt{P}_5, \mathtt{P}_6)$	Vec2 $\nabla\mathtt{I}_5 = [\nabla\mathtt{P}_5; \nabla\mathtt{P}_6]^\top \text{GradIntFun}(\mathtt{P}_5, \mathtt{P}_6)$
double $\mathtt{I}_6 = \text{IntFun}(\mathtt{P}_6, \mathtt{P}_1)$	Vec2 $\nabla\mathtt{I}_6 = [\nabla\mathtt{P}_6; \nabla\mathtt{P}_1]^\top \text{GradIntFun}(\mathtt{P}_6, \mathtt{P}_1)$
return $\sum_{i=1}^6 \mathtt{I}_i$	**return** $\sum_{i=1}^6 \nabla I_i$

Figure 4.2: **Computation of $\frac{\partial}{\partial \mathbf{v}} f_{i,t,k}(\mathbf{V})$.** The surface integral $f_{i,t,k}$ is computed as a sum of line integrals over the intersection polygon's edges. The derivatives of every line integral wrt $\mathbf{v}$ are computed using the chain rule, i.e. by factoring them into the product of the derivatives of the integral function IntFun wrt the segments' endpoints, and the derivatives of the endpoints wrt to $\mathbf{v}$. In the pseudocode, intersection polygon's vertices that are functions of $\mathbf{v}$, and their derivatives, are marked in red, making it easy to recognize the three cases discussed in section 4.2.1.

Figure 11. Computing the derivative in (44). Equation references are to [21].

References

[1] L. Ambrosio. Existence theory for a new class of variational problems. *Archive for Rational Mechanics and Analysis*, 111(4):291–322, 1990.

[2] L. Ambrosio, N. Fusco, and D. Pallara. *Functions of Bounded Variation and Free Discontinuity Problems*. Oxford University Press, 2000.

[3] D. Capel. *Image mosaicing and super-resolution*. Springer-Verlag, 2004.

[4] A. Chambolle. Image segmentation by variational methods: Mumford and Shah functional and the discrete approximations. *SIAM J App Math*, 55(3):827–863, 1995.

[5] R. Fattal. Image upsampling via imposed edge statistics. *ACM Trans. Graph.*, 26(3):95, 2007.

[6] G. Freedman and R. Fattal. Image and video upscaling from local self-examples. *ACM Trans. Graph.*, 30:12:1–12:11, 2011.

[7] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *Proc. ICCV*, 2009.

[8] M. Kazhdan and H. Hoppe. Streaming multigrid for gradient-domain operations on large images. *ACM Trans. Graph.*, 27(3):21ff, 2008.

[9] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *NIPS*, 2009.

[10] A. Levin and B. Nadler. Natural image denoising : Optimality and inherent bounds. *Proc. CVPR*, 2011.

[11] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1998.

[12] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. An algorithm for minimizing the Mumford-Shah functional. In *Proc. ICCV*, 2009.

[13] A. Rav-Acha, P. Kohli, C. Rother, and A. Fitzgibbon. Unwrap mosaics: A new representation for video editing. *ACM Trans. Graph.*, 27(3):17:1–17:11, 2008.

[14] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Proc. CVPR*, pages 860–867, 2005.

[15] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *Proc. CVPR*, 2007.

[16] K. Schelten and S. Roth. Connecting non-quadratic variational models and mrfs. In *Proc. CVPR*, 2011.

[17] T. Schoenemann, F. Kahl, and D. Cremers. Curvature regularity for region-based image segmentation and inpainting: A linear programming relaxation. In *Proc. ICCV*, 1999.

[18] R. Szeliski. Locally adapted hierarchical basis preconditioning. *ACM Trans. Graph.*, 25(3):1135–1143, 2006.

[19] D. Terzopoulos. Multilevel reconstruction of visual surfaces: Variational principles and finite element representations. In A. Rosenfeld, editor, *Multiresolution Image Processing and Analysis*, pages 237–310. 1984.

[20] M. Unger, T. Pock, M. Werlberger, and H. Bischof. A convex approach for variational super-resolution. In *DAGM-Symposium*, pages 313–322, 2010.

[21] F. Viola. *Resolution-independent Image Models*. PhD thesis, University of Cambridge, 2011.

[22] A. Vlasenko and C. Schnörr. Physically consistent and efficient variational denoising of image fluid flow estimates. *IEEE Trans Image Proc*, 19(3):586–595, 2010.

[23] A. S. Willsky. Multiresolution Markov models for signal and image processing. In *Proceedings of the IEEE*, pages 1396–1458, 2002.