Curvature regularization for resolution-independent images

Dickinson College Technical Report, May 2013

John MacCormick · Andrew Fitzgibbon

This technical report is an extended version of a conference paper [7].

Abstract A resolution-independent image models the true intensity function underlying a standard image of discrete pixels. Previous work on resolution-independent images demonstrated their efficacy, primarily by employing regularizers that penalize discontinuity. This paper extends the approach by permitting the *curvature* of resolution-independent images to be regularized. The main theoretical contribution is a generalization of the well-known elastica energy for regularizing curvature. Experiments demonstrate that (i) incorporating curvature improves the quality of resolution-independent images, and (ii) the resulting images compare favorably with another state-of-the-art curvature regularization technique.

Keywords curvature; elastica; regularization

1 Introduction and related work

Viola *et al.* [22,23] introduced the notion of a *resolution-independent* latent image to model the true intensity function underlying a standard image of discrete pixels. Figure 1 gives an example of the approach: the true intensity function is approximated by a piecewise linear function u, whose linear patches are defined on a triangle mesh. The crucial feature is that the mesh's vertices are positioned with arbitrary precision, which frees the model from any notion of discrete pixels. The vertex positions and patch intensities are determined by minimizing an energy that includes a regularizer term, which models the prior expectations of resolution-independent images in general.

John MacCormick
Department of Computer Science, Dickinson College, USA

Andrew Fitzgibbon
Microsoft Research, Cambridge, UK

Previous work on resolution-independent images employed a regularizer based primarily on the *discontinuities* in u. The main contribution in this paper is to extend the regularizer to incorporate the *curvature* of u. Starting from the well-known *elastica* energy [2], we derive explicit expressions for computing the elastica energy on the smooth and non-smooth regions of the image domain. The non-smooth region includes *steps* and *corners* (defined rigorously later), leading to separate step energy and corner energy terms in the energy functional. The paper also includes practical experiments demonstrating the benefits of the approach and a favorable comparison with another state-of-the-art curvature regularizer.

There is a considerable amount of related work on curvature regularization, including a long history of variational and level set methods (e.g. [10,15,19]), methods derived from the elastica energy (e.g. [2,11]), and other approaches such as total curvature [4] and Gaussian derivatives [6]. The work of Schoenemann *et al.* [18] and Strandmark and Kahl [20] is most closely related to the present paper. These approaches regularize curvature based on a fixed [18] or adaptive [20] mesh, employing linear programming relaxations for optimization. However, the meshes are restricted to a fixed small set of edge angles, so lines not at those orientations must be jagged. In our work, all angles are equally treated (ignoring floating point issues). The curvature term contrasts with this paper in that it applies to binary images and to corners with exactly two prongs (as defined in Section 4.2); the approach here permits resolution-independent images with arbitrary intensities and multi-pronged corners. Hence, we believe the novel theoretical contribution of the paper is twofold: first, the well-known approach of regularizing curvature by minimizing an elastica energy is reformulated so that it can be applied explicitly to resolution-independent images (Sections 3 and 4); second, this reformulation leads to a corner energy that has not, to our knowledge, been studied previously (Section 4.2).

2 The set of resolution-independent images

At the core of our approach is the concept of a *resolution-independent image*. Informally, a resolution-independent image is produced by an idealized camera with infinite resolution, infinite color depth, infinite depth of field, and zero noise. So the resulting image is defined on a connected subset Ω of $\mathbb{R}^2$, rather than on a discrete set of pixels as with any real camera. Similarly, the image takes values in a continuous range (say, $[0, 1]$), as opposed to a discrete set of greylevels. (For simplicity, we consider only monochrome images throughout the paper.) However, it is not the case that all functions $u : \Omega \to [0, 1]$ are resolution-independent images. This is because we also adopt an idealized model of the physical world: the world consists of piecewise smoothly-varying objects, and the color of each object is also piecewise smooth. As a result, resolution-independent images are piecewise smooth too.

Thus, Ω is partitioned into a *differentiable* region D (where u is continuously differentiable) and its complement J, termed the *jump set* (where u or its derivative is discontinuous). As shown in Figure 2, it is useful to further partition J into *steps*, *ridges*, and *corners*—so $\Omega = D \cup J_{\text{step}} \cup J_{\text{ridge}} \cup J_{\text{corner}}$. Some mild assumptions described below guarantee that u is well-behaved near the jump set. In particular, $\lim_{x \to x_0} u(x)$ exists for any $x_0 \in J$, and is independent of the path used to approach x_0, provided the path remains in the differentiable region D.

Note that the concept of a resolution-independent image is not new. Apart from minor technical differences, our resolution-independent images are identical to the so-called *cartoon* functions $f(x, y)$ introduced over 20 years ago by Mumford and Shah [12]. Many subsequent works have followed their formulation. Notwithstanding the considerable mathematical similarities between the Mumford-Shah model and our own, there is a crucial difference in philosophy and methodology. The vast majority of work in the Mumford-Shah tradition is described within a theoretical framework seeking a piecewise smooth cartoon $f(x, y)$ that approximates an arbitrary "true" image $g(x, y)$, where both f and g are defined on a plane domain $\Omega \in \mathbb{R}^2$. But at implementation time, both f and g are typically discretized onto fixed lattices (which may or may not coincide with the grid of pixels of a real-world input image I). In contrast, the approach of this paper is to think of the desired $f(x, y)$ as the "true" image—a resolution-independent image taken by an idealized camera. In this approach, the function g does not appear explicitly, since the input image I is regarded as a discretely-sampled, noisy, and blurred version of f. Moreover, even in the numerical implementation, f is never discretized onto a fixed lattice. Instead, we use a completely adaptive piecewise-linear triangle mesh whose vertices can move during optimization, taking arbitrary locations in the image plane.

For a detailed discussion of the differences between the present approach and the Mumford-Shah tradition, please see the work of Viola *et al.* [23], which is the direct inspiration for the present paper. Indeed, the *latent images* of Viola *et al.* are identical to our resolution-independent images, except that some additional technical conditions are imposed here to ensure that the required curvature-related quantities are finite. We switch terminology from "latent" to "resolution-independent" to emphasize that the implementation places no limit on the resolution of the output (except for floating-point limitations, which are negligible for the applications considered here).

We now establish some more formal definitions and notation. Let $\Omega \subset \mathbb{R}^2$ be a bounded open set, and $u : \Omega \to [0, 1]$ a real-valued function defined on Ω. Then u is a *resolution-independent image* on Ω if there is a finite collection of disjoint, connected, open subsets D_i, each of which has a piecewise twice continuously differentiable boundary, such that:

1. Ω is the disjoint union of the D_i and their boundaries (or more precisely, that part of the D_i's boundaries inside Ω); and,
2. u is continuously differentiable on each D_i, and obeys a Lipschitz condition on each D_i; and
3. each of the (finitely many) smooth portions of boundary of the D_i also obeys a second order Lipschitz condition.

The Lipschitz conditions are stronger than necessary for the framework of this paper, but they are a convenient way to ensure that u behaves well and they have no practical impact on the type of problems addressed. Insisting that Ω be open is also a convenience, since it excludes the image boundary from further consideration.

It is worth noting here that the implementation in this paper models only a very restricted class of resolution-independent images: piecewise-linear triangle meshes. That is, each D_i is a triangle, and u is a linear function on each D_i. However, in Sections 3 and 4, we develop a theory of curvature regularization

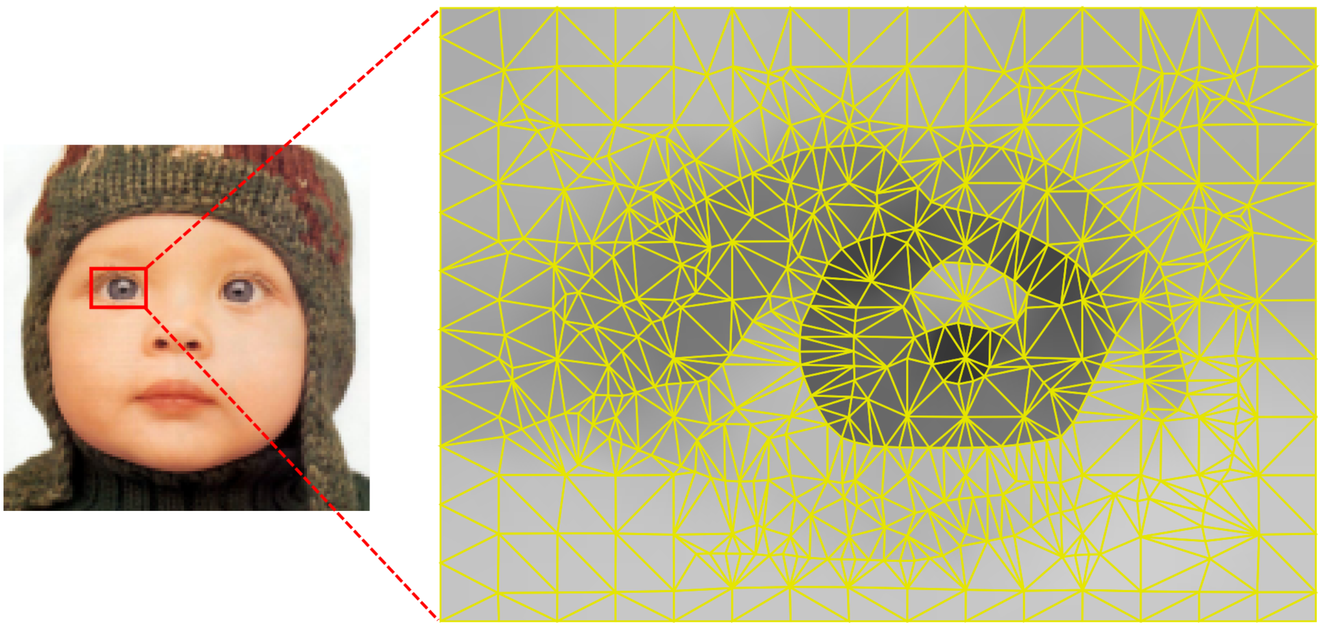


Fig. 1 In this paper, resolution-independent images are modeled as in Viola *et al.* [23], employing a piecewise linear intensity function defined on a triangle mesh whose vertices are positioned with arbitrary precision.

for the entire class of resolution-independent images. This general theory is then applied to piecewise-linear triangle meshes in Section 4.4.

Let us denote the set of all resolution-independent images on Ω by $\mathcal{I}(\Omega)$. Note that $\mathcal{I}(\Omega)$ is a proper subset of $\mathrm{BV}(\Omega)$, the set of functions of bounded variation on Ω. Intuitively, we can think of $\mathcal{I}(\Omega)$ as the set of bounded variation functions whose regions of discontinuity are "nice" (i.e. consisting of piecewise smooth curves) and whose gradient is bounded wherever it is defined.

We call the union of the boundaries of the D_i (but excluding the boundary of Ω) the *jump set* of u, denoted J. The terminology is motivated by the fact that any "jumps" in u (i.e. discontinuities in u or its derivative) must occur on J. An alternative definition is that $J = \Omega - D$, where $D = \cup_i D_i$ is the region where u is guaranteed to be continuously differentiable. (The notation D is intended to be a mnemonic for "differentiable.")

It follows immediately from the definition of a resolution-independent image that J consists of: (i) a finite set of *corner points* c_i; and (ii) a finite set of twice continuously differentiable curves γ_i which run between the corner points. We may assume that the γ_i are parameterized by arc length, and the second order Lipschitz condition means the curvature of the γ_i has an upper bound. The Lipschitz condition for u on D means that u has well-defined limits as it approaches the jump set. In particular, we can fix an orientation for each γ_i so that it makes sense to talk of the "left" and "right" sides of γ_i. Then for any point $\mathbf{x} = (x, y)$ lying on one of the γ_i, the limiting values $u^L(\mathbf{x})$ and $u^R(\mathbf{x})$, as we approach $\mathbf{x}$ from the left and right sides respectively, exist and are independent of the path used. Similarly, suppose the boundary of one of the regions D_i contains the corner point c_j. Then the limiting value of u as we approach c_j from within D_i exists and is independent of the path, provided only the approaching path stays within D_i.

It will be useful to partition the jump set of u into corners, steps, and ridges. For informal definitions, see Figure 2. Formally, we first define the *corner set* $J_{\text{corner}} = \cup_i c_i$ as the union of the endpoints of the γ_i. Next define the *step set*

$$J_{\text{step}} = \{\mathbf{x} \in J - J_{\text{corner}} \text{ s. t. } u \text{ is discontinuous at } \mathbf{x}\} \tag{1}$$

Finally, the *ridge set* is the remainder of the jump set: $J_{\text{ridge}} = J - J_{\text{corner}} - J_{\text{step}}$. Intuitively, ridges are locations where u is continuous but its derivative is discontinuous or undefined. As a matter of technical convenience, however, our definition of J_{ridge} also permits points where both u and its derivative are continuous.

To summarise notation, the domain Ω of a given resolution-independent image u can be expressed as the disjoint union of the continuously differentiable region D and the three types of jumps, as follows:

$$\begin{aligned}\Omega &= D \cup J \\ &= D \cup J_{\text{step}} \cup J_{\text{ridge}} \cup J_{\text{corner}}.\end{aligned} \tag{2}$$

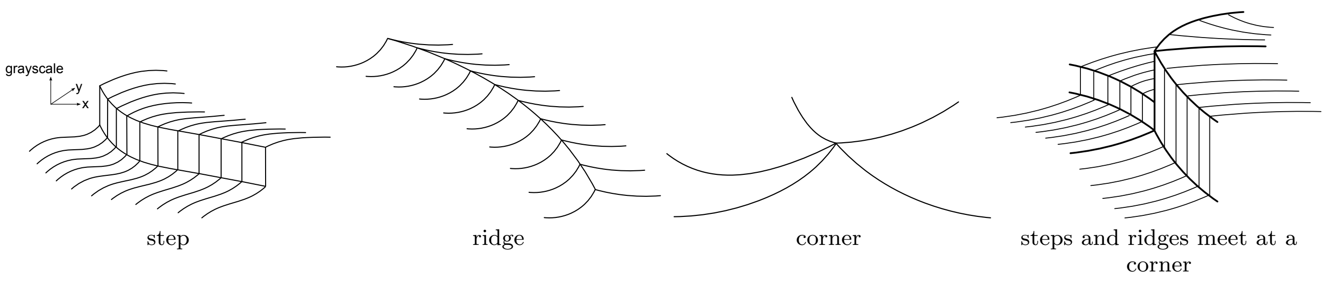


Fig. 2 Taxonomy of the jump set of a resolution-independent image. Each panel shows part of a 2D grayscale resolution-independent image, visualized as a surface. The image plane coincides with the horizontal x-y plane, and the grayscale intensity of the image is plotted on the vertical z axis, as indicated by the set of axes on the left.

3 Regularizers for resolution-independent images

We are interested in imposing a prior on resolution-independent images u. This will be done via a real-valued regularizer $E(u)$, with the usual interpretation of E as an energy functional, so that u-functions with low values of E have high prior probability.

The regularizer to be used in this work is a generalization of two commonly-used regularizers: the total variation energy and the elastica energy. To place the novel contribution of this paper in context, the total variation and elastica energies are reviewed in Sections 3.1 and 3.2 respectively. These sections are reformulations of standard, well-known material, albeit described in the non-standard context of resolution-independent images.

3.1 Total variation

One commonly-used regularizer for images is the *total variation*:

$$E_{\mathrm{TV}}(u) = \int_{\Omega} |\nabla u|. \tag{3}$$

At first glance, this expression appears problematic for discontinuous u, since ∇u is not defined at discontinuities. It is well known, however, that the total variation E_{TV} is defined—and finite—for all functions u of bounded variation, provided we interpret $\int |\nabla u|$ suitably (for example, as a distribution). There are rigorous measure-theoretic approaches to this [24], but the simplest approach is to smooth u with a small kernel of unit volume, and take a limit as the width of this kernel tends to zero. It is not hard to show that, for $u \in \mathcal{I}$, the contribution of a small element ds of the step set is $|u^L - u^R| ds$. By integrating these small elements ds over the step set J_{step}, we see that the contribution of J_{step} to the integral (3) is

$$\int_{J_{\mathrm{step}}} |u^L - u^R|. \tag{4}$$

Thus, for $u \in \mathcal{I}$, we can decompose the total variation regularizer as:

$$E_{\mathrm{TV}}(u) = \int_{D} |\nabla u| + \int_{J_{\mathrm{step}}} |u^L - u^R| \tag{5}$$

This justified by our previous calculation (4) of the contribution of J_{step}, and by noting that J_{ridge} and J_{corner} contribute nothing to the total variation.

The intuition behind using total variation as a regularizer should be obvious: it favors monotonicity, without specifying any preference between monotonic functions with the same boundary conditions. For example, consider a u that is a sequence of steps, each in the same direction (either all up or all down), starting at height 0 and finishing at height H. The heights of the individual steps can be different. This u receives the same energy as any smooth, monotonic interpolation from 0 to H.

3.2 Elastica energy

The *elastica energy* is a commonly-used regularizer for curvature in computer vision applications. The one-dimensional version of this energy, derived from the physical energy required to bend a thin pliable rod into a given smooth shape, was considered as early as 1744 by Euler ([3]; see this paper's appendix for details). For a smooth curve Γ parameterized by arc length s, it is given by

$$E_{\text{1D-elastica}}(\Gamma) = \int_{s} (a + b\kappa(\Gamma, s)^p)\, ds. \tag{6}$$

Here, $a, b, p \geq 0$ are constants and $\kappa(\Gamma, s)$ is the (unsigned) curvature of Γ at s, as defined in elementary geometry. Physics (and Euler) say that $p = 2$, but other values may give good results in computer vision applications.

Recall that κ is typically defined as the (possibly signed) magnitude of the second derivative of Γ, or equivalently as the reciprocal of the radius of curvature [5]. Throughout this paper, we adopt the convention that curvature is an unsigned quantity.

One-dimensional elastica energies have also been used in numerous computer vision applications [11,13]. Of more direct interest here is the generalization of elastica energy to two dimensions, as proposed by Masnou and Morel [9], and employed by many others (e.g. [18]). This two-dimensional elastica energy is given by

$$E_{\text{elastica}}(u) = \int_{\mathbf{x}\in\Omega} (a + b\kappa_{\text{LL}}(\mathbf{x})^p)|\nabla u(\mathbf{x})|\, d\mathbf{x}. \tag{7}$$

Here $\kappa_{\text{LL}}(\mathbf{x})$ is the (unsigned, 1D) curvature of the level line of u passing through $\mathbf{x} \in \Omega$. We will refer to κ_{LL} as the *level line curvature.* To define κ_{LL} more precisely, fix $\mathbf{x}_0 \in \Omega$ with $u(\mathbf{x}_0) = l$, and define the level set $\Gamma_l = \{\mathbf{x} \in \Omega | u(\mathbf{x}) = l\}$. Typically, the intersection of a sufficiently small neighborhood of $\mathbf{x}_0$ with Γ_l is a simple smooth curve $\gamma(s)$, parameterized as usual by arc length. We can then define $\kappa_{\text{LL}}(\mathbf{x}_0) = \kappa(\gamma, s_0)$, where s_0 has been chosen so that $\gamma(s_0) = \mathbf{x}_0$, and where κ is the (unsigned) 1D curvature from elementary geometry.

Chan *et al.* [2] provide a detailed and illuminating derivation of the 2D elastica energy (7) from the 1D pliable-rod definition (6). The basic idea is to integrate the 1D version over levels l; the extra weight of $|\nabla u(\mathbf{x})|$ in (7) then appears as the Jacobian when transforming from height and arc-length parameters (l, s) to image plane parameters $\mathbf{x} = (x, y)$.

As with the one-dimensional elastica energy, the two-dimensional energy (7) has an intuitive physical interpretation: it is the total amount of energy that would be expended to build the image out of thin, horizontal, pliable rods, assuming the energy of each individual rod is given by Equation (6) multiplied by the height spacing δl between rods. Note that for this physical analogy to be appropriate, the rods must be horizontal (so that they correspond to level sets), and they should be placed at equally-spaced heights separated by δl. As we will be repeatedly appealing to this physical interpretation of the elastica energy later, let us call it the *pliable rod analogy.*

There are three obvious problems with our definition of level line curvature for resolution-independent images. First, if u is flat (i.e. $\nabla u = 0$) in a neighborhood of $\mathbf{x}_0$, then there is no unique level curve γ passing through $\mathbf{x}_0$, and κ_{LL} is not defined at all. Second, if $\mathbf{x}_0$ is in the step set of u, the level curve may be degenerate (possibly consisting of a single point, for example), so κ_{LL} may not be defined. Third, if $\mathbf{x}_0$ is in the ridge set or corner set of u, the level curve may either be undefined or it may have a sharp corner at $\mathbf{x}_0$, meaning κ_{LL} is infinite. It turns out that the first two of these problems are easily rectified, but the third remains troublesome. Let us examine each in turn:

Problem 1 ($\nabla u = 0$): The value of κ_{LL} in regions where $\nabla u = 0$ is actually irrelevant, since throughout this document κ_{LL} (or, more generally, κ_{LL}^p) always appears multiplied by $|\nabla u|$. Equation (7) provides an example of this. For concreteness, we can arbitrarily define $\kappa_{\text{LL}} = 0$ in regions where $\nabla u = 0$, and this problem is solved. Alternatively, we can appeal to the pliable rod analogy discussed above. Because no rods are required to "build" a flat area of the image, the elastica energy of a flat area is zero.

Problem 2 ($x \in J_{\text{step}}$): This problem can be solved by again appealing to the pliable rod analogy. It turns out (see Section 3.3) that the contribution to (7) of a small element ds of the step set is $(a + b\hat{\kappa}^p|u^L - u^R|)ds$, where $\hat{\kappa}$ is the curvature of the step set itself. (Note that $\hat{\kappa}$ is defined and finite everywhere in J_{step}, by the definition of a resolution-independent image.) Hence, this problem is also solved.

Problem 3 ($x \in J_{\text{ridge}}, J_{\text{corner}}$): For points on a ridge, the pliable rod analogy fails, since the level sets at a ridge may have sharp corners. The assumed form (6) of the energy for our rods becomes infinite at a sharp corner (except in the particular case $p = 1$), as we can see from the following limit argument. We keep the angle of the corner constant, while decreasing the radius r of the circular arc used to approximate it. The length of the arc is proportional to r, while the curvature of the arc is, by definition, $1/r$. The energy therefore has a term proportional to r^{1-p}, which for $p > 1$ becomes infinite as $r \to 0$. The same problems of infinite energy apply to sharp corners where u has a step discontinuity.

We could, of course, solve this problem by permitting infinite energy values, and setting $E_{\text{elastica}}(u) = \infty$ for images u that have ridges or discontinuous corners. Indeed, this is what some previous authors employing the elastica energy (7) have implicitly done, with the consequence that ridges and discontinuous corners in u are forbidden (e.g. [2]). Whether or not it is reasonable to take this approach depends on the application. For example, if we are trying to perform inpainting using (7) as the regularizer, and the boundary of the region to be inpainted has no ridges, it is reasonable to attempt an inpainting with no ridges. But if the boundary does have ridges, it seems problematic to forbid ridges in the inpainted region. We propose a simple solution to the problem of ridges in Section 4.1, and corners in Section 4.2.

3.3 Computation of the step contribution

In the discussion of Problem 2 above, it was claimed that the contribution to the elastica energy of a small element ds of the step set is $(a+b\hat{\kappa}^p)|u^L-u^R|\,ds$, where $\hat{\kappa}$ is the curvature of the step set. We now give an argument supporting this claim, by appealing directly to the pliable rod analogy.

Consider a small portion ds of J_{step} shown in Figure 3(a), where the portion is small enough that we can approximate u^L and u^R as constant. To build this part of the image requires stacking horizontal rods directly on top of each other. Each individual rod has energy $\delta l(a+b\hat{\kappa}^p)\,ds$, by definition. The total height of the stack is just $|u^L-u^R|$, so the contribution of this stack is $(a+b\hat{\kappa}^p)|u^L-u^R|\,ds$. Integrating over all elements of the step set, this is equivalent to stating that the contribution of the entire step set to the elastica energy is

$$\int_{J_{\text{step}}} (a+b\hat{\kappa}^p)|u^L-u^R|\,ds. \tag{8}$$

Obviously, the above argument is based on physical intuition rather than mathematical rigor, which may trouble some readers. In this particular case, it is relatively easy to give a more rigorous calculation, based on smoothing u with a small unit-volume kernel, applying the definition of elastica energy (7) that is valid for smooth u, then taking the limit as the width of the kernel tends to zero. However, we prefer the approach based on physical intuition because it is easier to understand, and our final goal does not require mathematical rigor. We need to construct an energy whose minimization results in pleasing resolution-independent images; constructing that energy via plausible physical reasoning is a perfectly acceptable approach. Hence, in the remainder of the paper, we will appeal to physical intuition whenever necessary without attempting to inject additional rigor.

4 Curvature-related extensions of the elastica energy

This section describes the main theoretical contributions of the paper. It first gives details of how to compute the contribution to the elastica energy due to ridges (Sections 4.1) and corners (Sections 4.2). Section 4.3 then unifies the preceding calculations into a single generalized elastica energy. Finally, Section 4.4 describes the variant of the generalized elastica energy appropriate for the triangle meshes used in the present paper. To the best of our knowledge, all three subsections present primarily novel material.

4.1 Computation of the ridge contribution

As remarked above, building a ridge out of pliable rods requires sharp corners in the rods, and this can lead to infinite energies if we insist on an elastica energy of the form (7). Two easy solutions to suggest themselves. One solution is to take $p=1$. In this particular case, we can take a limit as the radius of curvature of a corner tends to zero, and the resulting value remains finite: the energy contribution of a sharp corner of exterior angle θ is easily seen to be $b\theta$. The other solution is to adopt a more general physical model of our rods: simply declare that the rods are made of some material that *can* be bent into a sharp corner using finite energy. For example, this energy could simply be $b\theta^p$ for some p, or the energy could also incorporate robustness by employing, say, $b\min(\tau,\theta^p)$ for some threshold τ. To keep our notation uncluttered, we will typically use $b\theta^p$ as the energy for a rod with a sharp corner of exterior angle θ, but the reader can bear in mind the many obvious generalizations that are possible.

Now that we have a viable physical model, it remains to compute the energy contribution of a small portion of ridge. For a sufficiently small portion, the left and right sides of the ridge can be modeled as planes, say by $z_1=a_1x+b_1y+c_1$ and $z_2=a_2x+b_2y+c_2$ respectively. The gradients of these planes are given by $\nabla z_1=(a_1,b_1)^\intercal$, $\nabla z_2=(a_2,b_2)^\intercal$. Each level set line in the neighborhood of this ridge consists of two line segments joined at a sharp corner. It's easy to check that the exterior angle at this corner is just the angle between the gradient vectors, denoted $\text{angle}(\nabla z_1,\nabla z_2)$, so the contribution of a single rod is

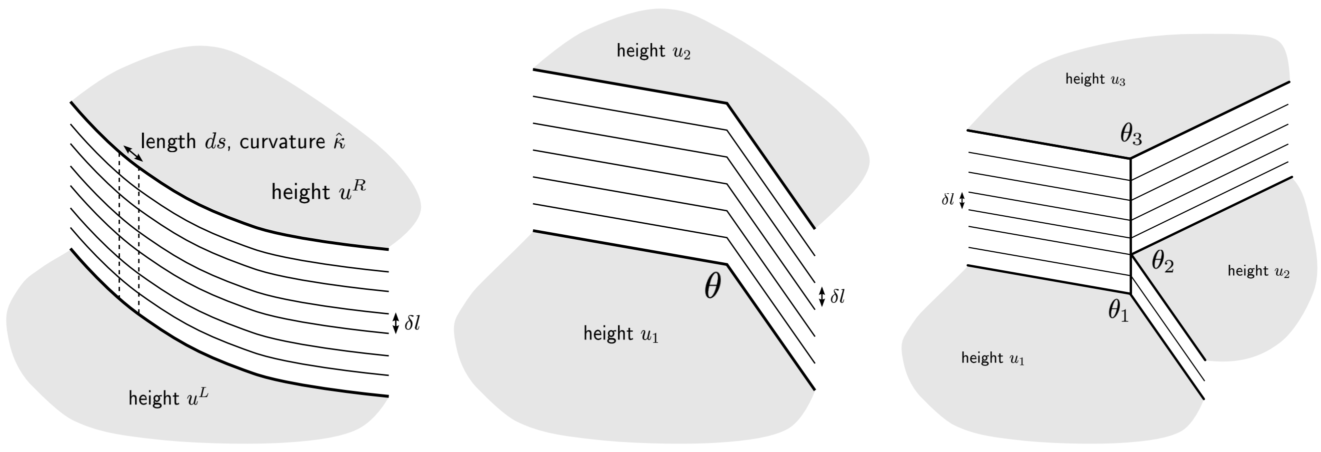
$$\delta l\,\text{angle}(\nabla z_1,\nabla z_2). \tag{9}$$

It's also easy to check that the magnitude of the gradient of the ridge itself (i.e. the intersection of the two planes) is $|a_1b_2-b_1a_2|/\left((b_1-b_2)^2+(a_1-a_2)^2\right)^{1/2}$, which can be rewritten in terms of gradients as

$$\frac{|\nabla z_1\times\nabla z_2|}{|\nabla z_1-\nabla z_2|}. \tag{10}$$

Here, $\times$ is the vector cross product and $|\bullet|$ is the Euclidean norm.

We now wish to add up the contributions of a large number of rods with energy (9), and change variables from level set parameter l to the ridge's arc length s. Some analysis of the number of rods with vertical separation δl needed to fill a macroscopic length of ridge shows that the slope of the ridge (10) is the correct Jacobian for this transformation. So we need to multiply (9) by (10) and integrate. Reverting to notation u^L,u^R (instead of planar patches z_1,z_2) for the



(a) step (b) 2-prong corner (c) 3-prong corner

Fig. 3 Computing the elastica energy of steps and corners.

resolution-independent image values on the left and right-hand sides of a ridge, we obtain the following expression for the energy of the ridge set:

$$E_{\text{ridge}} = \int_{J_{\text{ridge}}} b\,\text{angle}(\nabla u^L, \nabla u^R)^p \frac{|\nabla u^L \times \nabla u^R|}{|\nabla u^L - \nabla u^R|}\, ds. \tag{11}$$

Note that we have also re-introduced the coefficient b which weights all the curvature-related terms in our energy. Later, we will use a further generalization in which the final factor (the slope of the ridge) is raised to a power α. And it is also useful to make explicit the dependence on parameters b, p, α, writing

$$E_{\text{ridge}}(b, p, \alpha) = \int_{J_{\text{ridge}}} b\,\text{angle}(\nabla u^L, \nabla u^R)^p \left(\frac{|\nabla u^L \times \nabla u^R|}{|\nabla u^L - \nabla u^R|}\right)^{\alpha} ds. \tag{12}$$

4.2 Computation of the corner contribution

For the reasons given in Section 3.3, we use the pliable rod model to compute the energy of a sharp corner point, such as the one in the right-most panel of Figure 2. It follows from the Lipschitz conditions of Section 2 that every non-trivial corner lies at the endpoint of two or more *prongs*: smooth curves in the jump set.[1] The right-most panel of Figure 2 has five prongs, for example. The image intensity is, by definition, smooth between prongs. The Lipschitz conditions of Section 2 also guarantee that the intensity tends to sensible limits as we approach the prongs and the corner point itself. Thus, by considering a sufficiently small neighborhood of the corner, the intensity function can be approximated arbitrarily well by using prongs that are straight lines and an intensity that is constant between prongs.

[1] The use of the word "non-trivial" here requires some explanation. A *trivial* corner is isolated from the step set (although it may adjoin the ridge set), as in the third panel of Figure 2. Such corners may have no prongs, but they also have zero energy, so we need not consider them here.

Let us first examine the simplest possible case: a two-prong corner (see Figure 3(b)), whose prongs meet with interior angle θ, with constant intensity values u_1 and u_2 on the inside and outside of the corner respectively. How would we build this geometric shape using horizontal pliable rods? As shown in Figure 3(b), each rod must be bent through angle $|\pi - \theta|$, and the rods are stacked vertically (using our standard vertical spacing, δl).

Here we encounter the same apparent difficulty as with ridges: the construction requires sharp corners in the rods, but this leads to infinite energies if we insist on an elastica energy of the form (6). Fortunately, the same solution applies. We assume the rods are made of some material that *can* be bent into a sharp corner using finite energy. For example, this energy could be $b|\pi - \theta|^p$ for some p, or the energy could also incorporate robustness by employing, say, $b \min(\tau, |\pi - \theta|^p)$ for some threshold τ. Later experiments use the non-robust version, which performs well for our applications.

Adopting this (literally) more flexible definition of a rod, the energy of each rod is $b|\pi - \theta|^p\, \delta l$, and the total height of the stack of rods is $|u_2 - u_1|$. Integrating over l, we obtain the total corner energy for a two-pronged corner as $b|\pi - \theta|^p |u_2 - u_1|$.

Now let us turn to the general case of a multi-pronged corner, with N prongs. (See Figure 3(c) for a 3-prong example.) For concreteness, label the prongs from 1 to N in an anti-clockwise direction. As before, we may assume the prongs are straight lines and the image takes on constant values $u_1, u_2, \ldots, u_N$ on the *wedges* between each consecutive pair of prongs. So u_1 is the value of the image on wedge 1 between prongs 1 and 2, and so on up to u_N, which is the value on wedge N between prong N and prong 1. The angle of wedge i is θ_i. In what follows, subscripts are computed modulo N. In particular, u_{N+1} means the same thing as u_1, and similarly for θ_{N+1}.

We can think of this simplified geometry as a circular pie cut into wedges, where each wedge happens to be of a different height and angle. We need to calculate the energy required to build this multi-level pie out of pliable rods, as in Figure 3(c). One simple approach uses recursion: find the lowest wedge, and build up the sides of the wedge to the height of the lowest adjacent wedge. At this point, the lowest wedge has effectively been removed from the structure, and the problem has been reduced to building a new pie that possesses one fewer wedge. The recursion can bottom out at two wedges, which is the two-prong case considered above. Alternatively, we can make our final formula (15) a little more elegant by bottoming out at one wedge, which is a degenerate "corner" of zero energy.

More formally, let i^* be the index of the lowest wedge, so

$$i^* = \operatorname*{arg\,min}_{i\in\{1,\dots,N\}} u_i. \tag{13}$$

Let j^* be the index of the lowest wedge adjacent to i^*, so

$$j^* = \operatorname*{arg\,min}_{i\in\{i^*-1,i^*+1\}} u_i. \tag{14}$$

Let $C = (u_i, \theta_i)_{i=1}^N$ denote the N-prong corner, and $E_{\mathrm{cnr}}(C)$ the desired elastica energy of this corner. Write $\hat{C}$ for the $(N-1)$-prong corner that results from filling in wedge i^* up to height u_{j^*}. Then compute $E_{\mathrm{cnr}}(C)$ recursively according to

$$E_{\mathrm{cnr}}(C) = \begin{cases} 0 & \text{if } N = 1, \\ E_{\mathrm{cnr}}(\hat{C}) + b|\pi - \theta_{i^*}|^p |u_{i^*} - u_{j^*}| & \text{if } N > 1. \end{cases} \tag{15}$$

Later, we will consider a generalization of this formula in which $|\cdot|$ is replaced by a robust function $\rho_\tau(\cdot) \equiv \min(\tau, |\cdot|)$ and raised to a power α. We also make explicit the dependence on parameters b, p by writing

$$E_{\mathrm{cnr}}(C; b, p, \alpha) = \begin{cases} 0 & \text{if } N = 1, \\ E_{\mathrm{cnr}}(\hat{C}; b, p, \alpha, \tau) \\ \quad + b|\pi - \theta_{i^*}|^p \rho_\tau(u_{i^*} - u_{j^*})^\alpha & \text{if } N > 1. \end{cases} \tag{16}$$

4.3 A generalized elastica energy for all resolution-independent images

Recall that the domain Ω of a resolution-independent image u decomposes into four different regions according to (2): differentiable (D), step (J_{step}), ridge (J_{ridge}), and corner (J_{corner}) regions. Therefore, the elastica energy (7) decomposes into integrals over these four regions, and the previous subsections computed expressions for each integral. Combining these—specifically, substituting (8), (12), and (16) into (7)—gives a more explicit expression for the elastica energy:

$$\begin{aligned} E_{\mathrm{elastica}}(u) = & \\ & \int_{\mathbf{x}\in D} (a + b\kappa_{\mathrm{LL}}(\mathbf{x})^p)|\nabla u(\mathbf{x})|\, d\mathbf{x} \\ & + \int_{J_{\mathrm{step}}} (a + b\hat{\kappa}^p)|u^L - u^R|\, ds \\ & + E_{\mathrm{ridge}}(b, p, \alpha) \\ & + \sum_{\text{corners } C} E_{\mathrm{cnr}}(C; b, p, \alpha) \end{aligned} \tag{17}$$

It is convenient to generalize this expression by allowing different exponents α_i for the gradient factor in each term, and also allowing different coefficients λ_i and curvature exponents p_i for each term. This gives an elastica energy E_G (where "G" stands for " generalized") defined by

$$\begin{aligned} E_G(u) = & \lambda_1 \int_{\mathbf{x}\in D} (a + b\kappa_{\mathrm{LL}}(\mathbf{x})^{p_1})|\nabla u(\mathbf{x})|^{\alpha_1}\, d\mathbf{x} \\ & + \lambda_2 \int_{J_{\mathrm{step}}} (a + b\hat{\kappa}^{p_2})|u^L - u^R|^{\alpha_2}\, ds \\ & + \lambda_3\, E_{\mathrm{ridge}}(b, p_3, \alpha_3) \\ & + \lambda_4 \sum_{\text{corners } C} E_{\mathrm{cnr}}(C; b, p_4, \alpha_4) \end{aligned} \tag{18}$$

The generalization to arbitrary α_i, λ_i, p_i is not justified by any physical or theoretical reasoning. Rather, we appeal to the fact that we are seeking a regularizer that works well in practice. The generalization is justified if, by generalizing a physically realistic expression to one that is not physically realistic, we can obtain better performance when analyzing real images. As we shall soon see, numerous previous authors have taken exactly the same approach. But it should be noted that the natural (i.e. physically realistic, according to the pliable rod model) values for the α_i, λ_i are all 1, and for the p_i the natural value is 2.

Let us now examine how the generalized elastica energy (18) relates to previous work. By taking $\lambda_1 = \lambda_2 = \alpha_1 = \alpha_2 = 1$, and $\lambda_3 = \lambda_4 = b = 0$, we recover the total variation, up to a constant multiplier. By taking

$\lambda_3 = \lambda_4 = b = 0$, we obtain an expression similar[2] to the regularizer used by Viola *et al.* [23]—which, as previously noted, is the direct inspiration for the present work. Hence, the high-level claim that the present work adds a notion of curvature to Viola *et al.* can now be made more explicit: this paper incorporates the corner energy, by permitting $\lambda_4 \neq 0$ in (18). But there is an extra wrinkle here. This paper does not implement the ridge energy, which is equivalent to setting $\lambda_3 = 0$. Hence, a more accurate description of this paper's contribution is that it permits a non-zero corner energy to be taken into account.

4.4 Elastica energy for images on a triangle mesh

We are particularly interested in computing the generalized elastica energy E_G for images that are piecewise linear on a triangle mesh. These images have zero curvature on the interiors of all the triangles, so $\kappa_{\mathrm{LL}} \equiv 0$ on D. Moreover, the mesh edges (which are all straight lines between triangle vertices) have zero curvature too, so $\hat{\kappa} \equiv 0$ on J_{step}. It is easy to see that this renders irrelevant the values of p_1, p_2, a, b in (18). And as mentioned in the previous subsection, this paper takes $\lambda_3 = 0$. These observations result in a simplified form of the elastica energy for triangle meshes, E_T (where the "T" stands for "triangle"):

$$\begin{aligned} E_{\mathrm{T}}(u) =&\lambda_1 \int_{\mathbf{x}\in D} |\nabla u(\mathbf{x})|^{\alpha_1}\, d\mathbf{x} \\ &+\lambda_2 \int_{J_{\mathrm{step}}} |u^L - u^R|^{\alpha_2}\, ds \\ &+\lambda_3\, E_{\mathrm{ridge}}(b, p_3, \alpha_3) \\ &+\lambda_4 \sum_{\text{corners } C} E_{\mathrm{cnr}}(C; b, p_4, \alpha_4, \tau) \end{aligned} \tag{19}$$

Previous work [22] has shown some benefits from taking $\alpha_1 = 2, p_i = 1$. The experiments in this paper also adopt these settings, and set all other constants (λ_i, α_i) to their physically realistic value (1.0), except where stated otherwise. The robustness parameter τ is set to 10% of the dynamic range in the input image.

5 Algorithmic details

The algorithm used here for computing resolution-independent images is modeled closely on Viola's work [22, 23], where the reader can find many additional details. In this paper, we provide only a high-level overview of the technique, and highlight several differences to the Viola algorithm.

The discussion so far focused exclusively on the regularization of our resolution-independent image u, but we also need a data term that expresses the affinity between u and some input image I. This input I is a standard, discrete set of grayscale pixel values denoted I_i. We assume pixel i of I was formed by blurring the true (continuous) intensity function with some kernel κ_i. This leads to a data term $\mathcal{D}(u, I)$ of the form

$$\mathcal{D}(u, I) = \lambda_0 \sum_i \| I_i - \int_{\Omega} \kappa_i(x) u(x)\, dx \|. \tag{20}$$

Here, λ_0 is the *data gain* expressing the relative importance of the data and regularization terms. Experiments in this paper take $\lambda_0 = 10$ (unless stated otherwise), a value that was determined by trial and error to yield reasonable performance on a variety of inputs. For the norm $\|\cdot\|$, we use the square of the standard Euclidean norm. Ideally, the kernel functions κ_i would be estimated from the point spread function of the camera used to capture I, but this lies outside the scope of the present paper. We take the pragmatic and simple choice of setting κ_i to be a 2D square box function, equal to 1 on the unit square centered at pixel i and zero elsewhere.

The computation of a resolution-independent image $\hat{u}$ is achieved by minimizing the total energy $\mathcal{E}(u, I)$, which combines the triangle mesh energy (19) and data term (20):

$$\hat{u} = \arg\min_u \mathcal{E}(u, I) = \arg\min_u (E_T(u) + \mathcal{D}(u, I)). \tag{21}$$

Recall that u is a piecewise linear triangle mesh, parameterized by: (i) the 2D locations of each vertex in the mesh (two real parameters per vertex); and (ii) the height and slope of each triangle in the mesh (three real parameters per triangle). The average density of the mesh is application-dependent. In experiments reported here, the typical distance between neighboring vertices is 1–3 pixels. Even on modest-sized images, this leads to tens of thousands of triangles and vertices, and hundreds of thousands of parameters. For example, the 256×256 input of the segmentation result in Figure 9 leads to a mesh with over 42,000 triangles, 21,000 vertices and a resulting total of 170,000 parameters.

All experiments in this paper perform the minimization (21) over these parameters by first initializing the mesh to a reasonable estimate, then applying an off-the-shelf nonlinear optimizer. Specifically, the initialization is done by using a regular grid of vertices spaced

[2] Viola *et al.* actually *did* implement a form of ridge energy different to the one considered here. Ignoring that ridge energy, (18) with $\lambda_3 = \lambda_4 = b = 0$ is identical to the regularizer of Viola *et al.*

1.5 pixels apart, augmented by further vertices placed at subpixel locations identified as edgels by a Canny edge detector. The intensity values are initialized by assigning each triangle the constant intensity obtained by integrating I over the triangle.

Minimization is performed in Matlab via Schmidt's `minFunc`[3], using the LBFGS algorithm [14] with default options. Note that LBFGS is a quasi-Newton method, requiring the objective function's derivative but not its Hessian. The derivative of non-corner terms is taken from Viola [22]; the derivative of the corner energy (16), although tedious to implement and debug, requires only elementary geometry and calculus.

The experiments described here require hundreds or thousands of iterations to reach convergence (as defined by the default `minFunc` criteria). The approach is thus rather computationally expensive. The computational cost of the experiments reported here, all employing Matlab implementations on a 2012 desktop PC with an Intel Core2 Q9400 CPU, range from several CPU-core-minutes (for the results of Figure 5) to nearly 50 CPU-core-hours (for the results of Figure 9).

As previously stated, the above approach follows Viola in many respects. There are two important differences, however. First, we perform joint optimization over all parameters simultaneously. This contrasts with Viola's approach, which alternates between optimizations over the vertex location variables and the intensity height/slope variables, and also employs so-called *N/Z flip* moves, which make global changes to the mesh structure.

Second, we take a simpler approach to the problem of degenerate triangles—triangles that become excessively narrow slivers as the optimization proceeds. If the mesh contains one or more problematic slivers, we remove a vertex from each sliver, and re-triangulate the resulting hole using a constrained Delauney triangulation [16]. Each new triangle's intensities must then be initialized based on the nearest undisturbed triangle, and the entire minimization restarted. In principle, this could lead to extremely slow convergence. In practice, however, we find that running sliver-removal just once before beginning any minimization is typically sufficient.

Figure 7, discussed in more detail below, demonstrates that our *joint* minimization approach has similar performance to the more elaborate *alternating* approach of previous work. Moreover, the joint approach is simpler to implement and appears to encounter fewer problems with degenerate triangles. (Note that this discussion compares optimization approaches only. So in this experiment, both the joint and alternating approaches incorporated the corner energy, and therefore required the corner energy derivative also.)

[3] Mark Schmidt, `http://www.di.ens.fr/~mschmidt/Software/minFunc.html`, 2012

6 Results

6.1 Qualitative assessment of incorporating curvature

Figure 4 demonstrates the main qualitative result of this paper: incorporating curvature into the energy functional leads to modest improvements in the quality of resolution-independent images. In terms of the standard taxonomy of computer vision tasks, Figure 4 is best described as an example of super-resolution, showing various upsampled outputs (c)–(h) based on the input (b). However, we wish to avoid for now analyzing metrics for any one task such as super-resolution, instead concentrating on the more abstract and fundamental goal of constructing resolution-independent images. Specifically, we address the question: do we obtain better resolution-independent images by incorporating curvature? To answer this question, we run our experiments twice with identical settings—except for the parameter λ_4 of (19) which is switched from 0 (corresponding to "without curvature") to 1 (corresponding to "with curvature") between experimental runs.

Let us now examine Figure 4 in more detail. Panel (a) shows the original image of a person holding a computer chip, from which this example is derived. Panel (b) is a 60×46 detail from the original, used as input for the remaining panels. Panels (c) and (d) are renderings of the resolution-independent images produced by running the minimization (21) described in the previous section: panel (c) is "without curvature" and (d) is "with curvature." To the human eye, these two outputs appear extremely similar, with excellent reconstructions in some regions (e.g. the *M*, the two *7*s, and the *C*) and imperfect ones in others (e.g. the *P* and the *3* have their interiors incorrectly filled).

But as shown in panels (e)–(h), which zoom in on some particular regions of interest, there are some subtle but important differences between the two outputs. (Note that these panels represent extreme super-resolution, showing regions that are 11×7 pixels in the input image.) Specifically, panels (e) and (f) show the letter *C* derived from the input, without and with curvature respectively. In both cases, the curved shape of the *C* has been recovered surprisingly well, albeit imperfectly. (We can see artifacts of the triangle mesh, such as the sharp corner points at the lower right end of *C* in both outputs.) More importantly, the output computed with curvature shows some improvement over the without-curvature output: in panel (f), the outline of the C represents a smoother curve, and the grayscale values in the interior of the C are also smoother. Panels (g) and (h) show a portion of a straight specular edge. Again, the extreme super-resolution performs well in both cases, recovering boundaries that are nearly straight despite the blocky, staircase-like input. And we again see artifacts of the triangle mesh in both outputs: a few triangles with incorrectly-inferred grayscale values protrude from the main strip of high intensity. But the more important point is that panel (h), computed with curvature, produces a straighter boundary for the high-intensity strip, when compared with panel (g) (which was computed without curvature).

Although we have shown outputs for only one image here, the results are typical. It is reasonable to conclude that incorporating curvature produces modest improvements in the detailed structure of resolution-independent images.

6.2 Quantitative assessment of incorporating curvature

The previous subsection provided a qualitative assessment of the improvements from incorporating curvature, based on the subjective assessment of whether highly-magnified portions of an image (such as Figure 4(e)–(h)) look reasonable. In this subsection, we confirm the previous qualitative results with a quantitative assessment based on peak signal-to-noise ratio (PSNR). The experiment involves the task of simultaneous denoising and upsampling, as shown in Figure 5. The ground truth image in panel (a) is a 64×64 detail of the well-known "peppers" image. Panel (b) is a blurred, noisy version of the ground truth. It was created by first averaging 4×4 blocks of (a), then adding Gaussian noise with standard deviation equal to 5% of the image's dynamic range. This results in a 16×16 image to be used as input to the algorithm for estimating resolution-independent images (Section 5). As with the previous experiment, outputs were produced without and with curvature energy, shown in panels (c) and (d) respectively. A subjective assessment seems to confirm the previous experiment, since the with-curvature result appears to have smoother object boundaries, and smoother grayscale values within objects.

Because we have the ground truth for this experiment, we can also assess these outputs quantitatively, by computing their PSNR with respect to the ground truth. PSNR is a well-known concept, but we give a precise definition here for completeness. To compute PSNR, the resolution-independent image is first rendered at the resolution of the ground truth. Denote the ground truth pixel values by $I_j \in [0,1]$ and the rendered output's pixel values by $I'_j \in [0,1]$. Let the number of pixels in each image be $N = 64 \times 64$. Then the PSNR is defined by

$$\text{PSNR} = -10 \log_{10} \left(\sum_j (I_j - I'_j)^2 / N \right). \qquad (22)$$

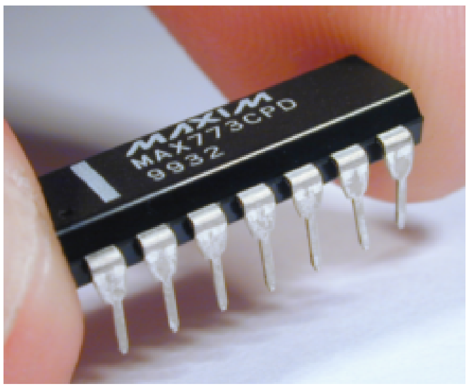
Figure 6 shows the results. This graph also demonstrates the sensitivity of the computation to the data gain parameter, λ_0, in Equation (20). The data gain is varied on the horizontal axis, with the corresponding PSNRs for the computations with and without curvature shown on the vertical axis. A higher PSNR corresponds to a higher-quality reconstruction, so it is clear that the with-curvature results are superior to the without-curvature results for each value of the data gain.

A further experiment demonstrated that these results are typical, on average. A 64×64 patch was selected uniformly at random from each of the first 40 images of the Berkeley Segmentation Dataset [8], and the above experiment was run with identical settings on all 40 patches. The mean improvement in PSNR after switching on curvature energy was 0.17 dB. Figure 8 shows the input patches for this experiment, together with the individual PSNR improvement for each patch. Note that incorporating curvature does not always result in significant improvement, and sometimes actually reduces PSNR. But the influence is strongly positive on average. For example, if we define PSNR changes of less than 0.15 dB as insignificant, then these 40 patches contain 10 significant improvements and only two significant reductions.

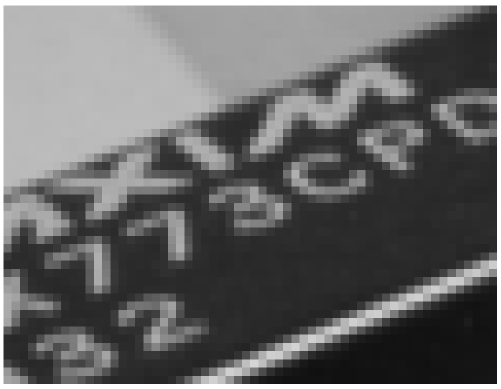
As discussed at the end of Section 5, this paper employs a simple joint optimization approach, contrasting with the more elaborate alternating approach of previous work. Figure 7 shows the computational expense of these two approaches for the experiment described above (i.e. simultaneously denoising and upsampling the "peppers" image). It is clear that the energy minimization proceeds at roughly the same rate for both approaches, but the alternating approach is less smooth since it encounters degenerate triangles more often. The resulting retriangulation can also lead to an increase in the energy value.

6.3 Comparison with alternative curvature regularization

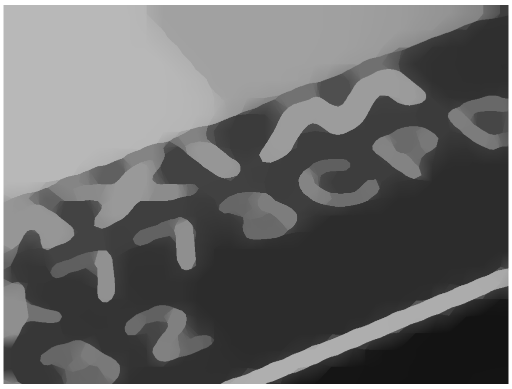
The previous two experiments focused solely on determining whether incorporating curvature regularization into the resolution-independent image framework improves the quality of results. However, curvature regularization is employed in many other paradigms, and it



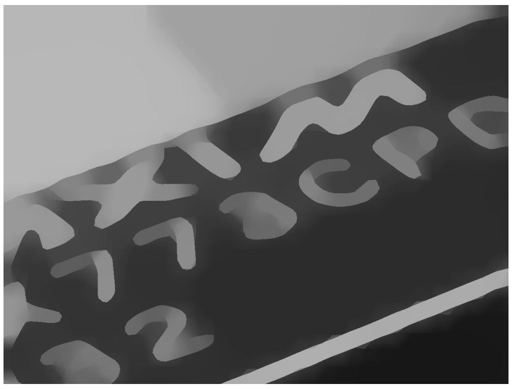
(a) 244 × 200 (uncropped) input image



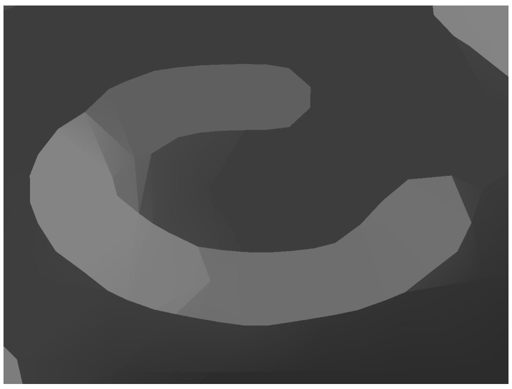
(b) 60 × 46 cropped input image



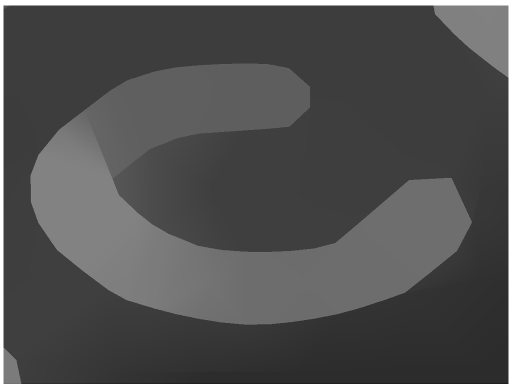
(c) output without curvature energy



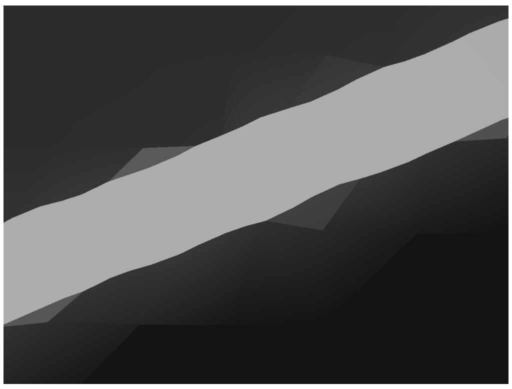
(d) output with curvature energy



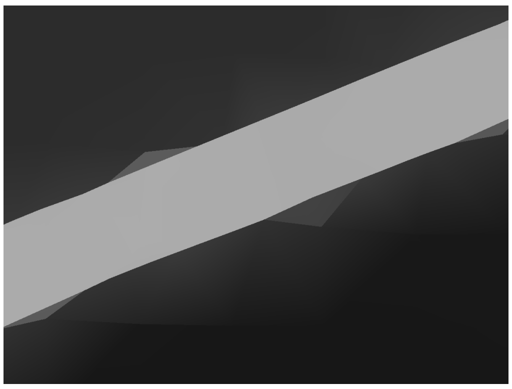
(e) an 11 × 7 region of (c) i.e. without curvature energy



(f) an 11 × 7 region of (d) i.e. with curvature energy

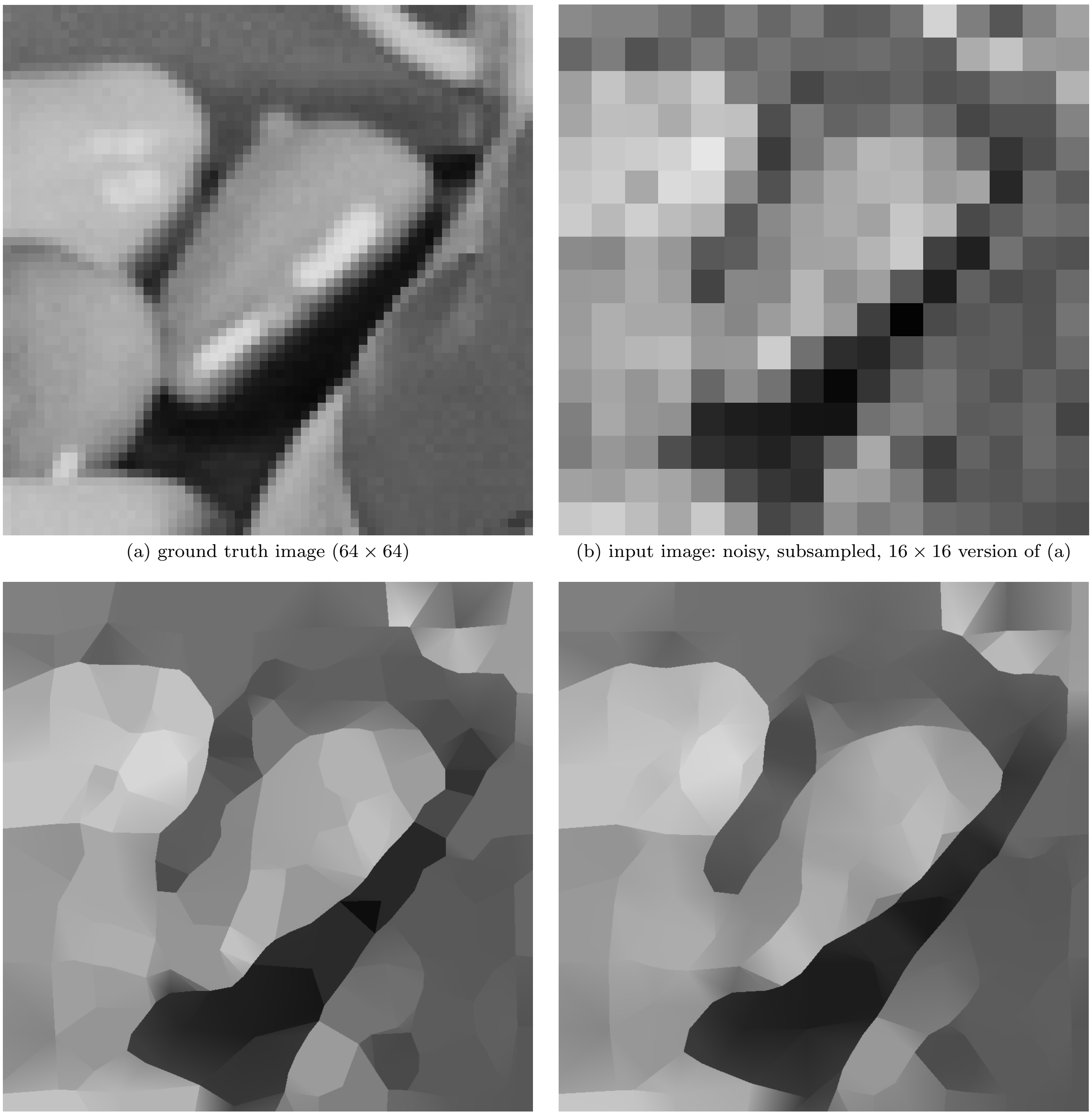


(g) an 11 × 7 region of (c) i.e. without curvature energy



(h) an 11 × 7 region of (d) i.e. with curvature energy

Fig. 4 Comparison of resolution-independent images computed with and without curvature energy.



(a) ground truth image (64 × 64)

(b) input image: noisy, subsampled, 16 × 16 version of (a)

(c) output without curvature energy

(d) output with curvature energy

Fig. 5 Simultaneous denoising and super-resolution.

is reasonable to ask how the approach of this paper compares to other state-of-the-art curvature regularization techniques. A comprehensive comparison with other curvature regularization techniques is beyond the scope of this paper, since our main objective is to enhance the theory and practice of resolution-independent images. Here, we provide here a comparison with one recent state-of-the-art approach: the technique of Strandmark and Kahl [20]. The Strandmark-Kahl (SK) approach minimizes a particular choice of elastica energy over an adaptive mesh, but the precise form of the energy and the methodology for adapting the mesh are quite different to the present paper.

In addition, the SK approach is targeted at regularizing curvature in *binary* output images, which can therefore be regarded as foreground-background seg-

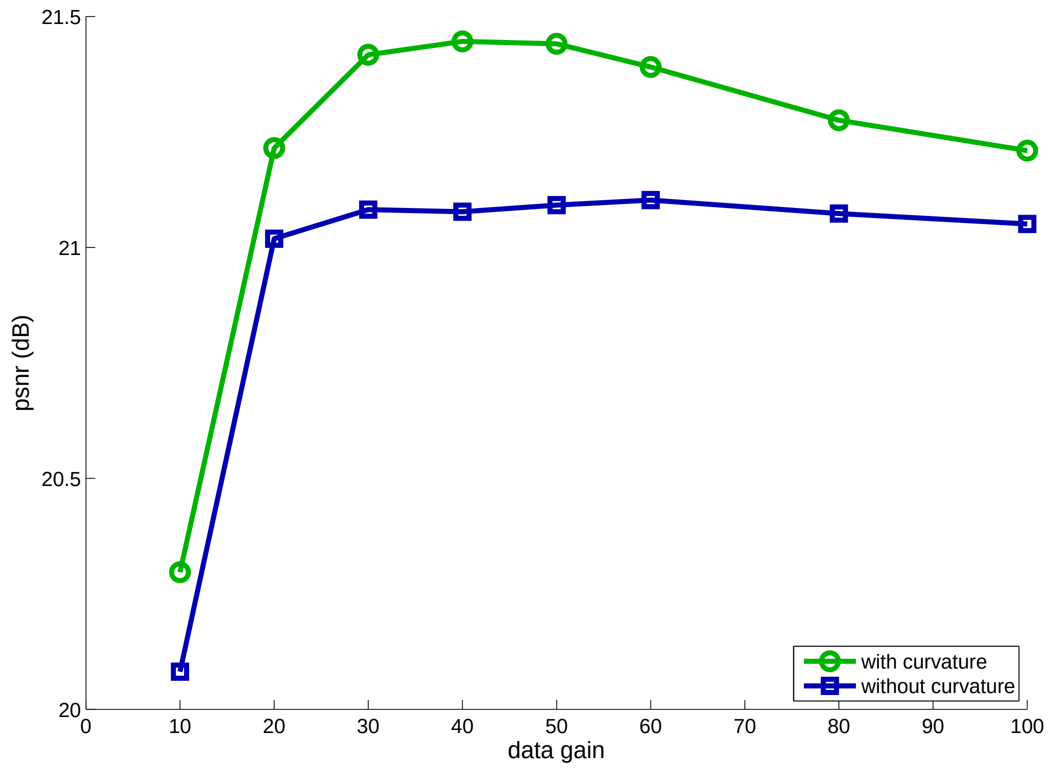


Fig. 6 Estimating a resolution-independent image with curvature energy produces superior PSNR.

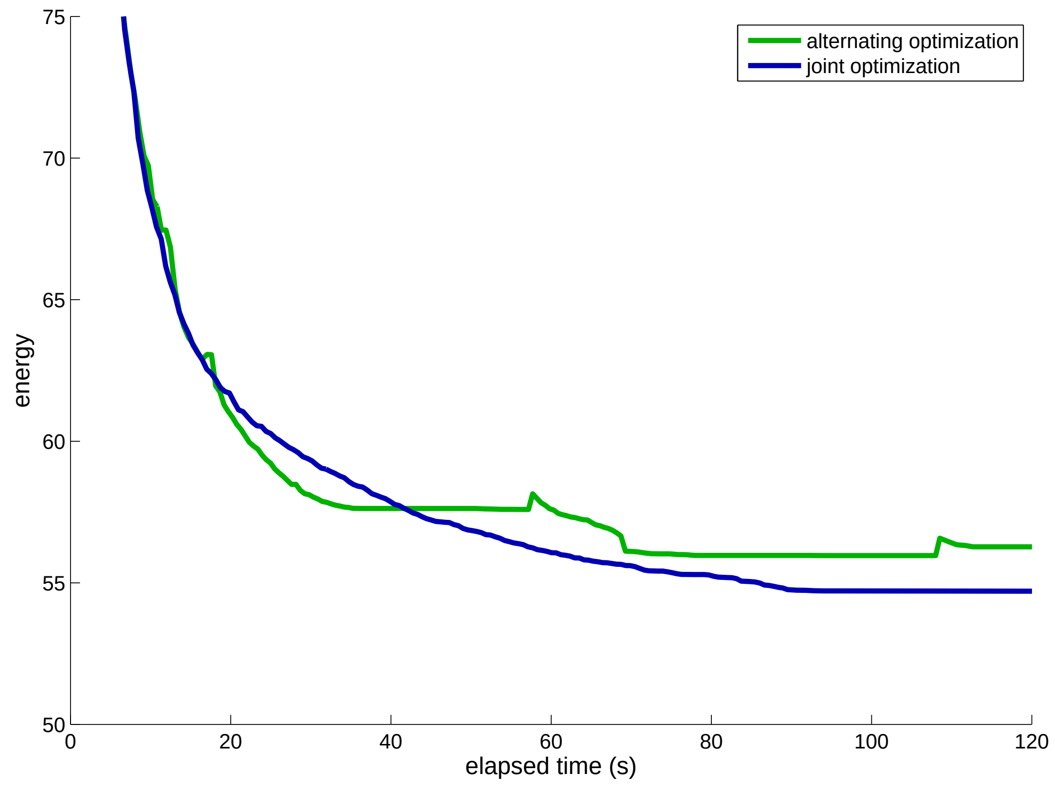


Fig. 7 The simple joint optimization approach of this paper has similar computational expense to the more elaborate alternating approach of previous work.

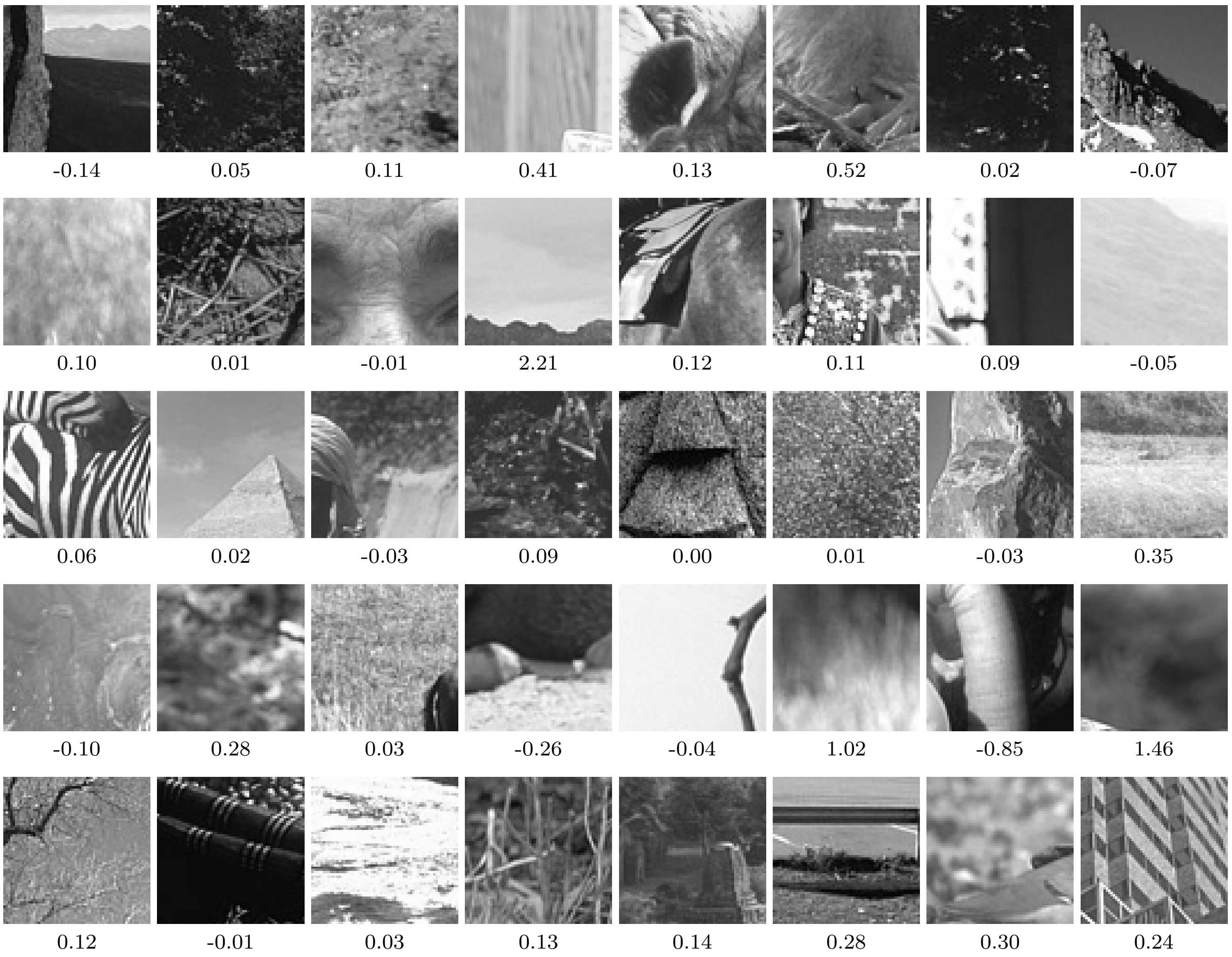


Fig. 8 Randomly-selected 64×64 patches from the first 40 images of the Berkeley Segmentation Dataset [8]. Below each image is the improvement in PSNR when curvature is used to compute the resolution-independent image.

mentations. So that our results can be compared directly with SK, we obtain a binary image by thresholding a rendering of the resolution-independent output. More precisely, given an input image I, we run the minimization algorithm of Section 5 to obtain a resolution-independent image u. Each triangle in the mesh of u is assigned a new constant intensity of 1 or 0 according to whether the triangle's mean intensity is above a given threshold. The resulting u' is a binary, resolution-independent image, and can be rendered at any desired resolution for comparison with other algorithms. The threshold is selected manually, after inspecting a histogram of the output's grayscale values.

Figure 9 shows the results of this comparison on the well-known "cameraman" image. Each image in the right-hand column is a detail of the corresponding image in the left-hand column. In the top row, panels (a) and (b), we have the 256×256 grayscale input image of the cameraman. The second row, panels (c) and (d), shows the resolution-independent image computed from the input. The third row, panels (e) and (f), shows the binary resolution-independent image computed from (c) as described above. The final row, (g) and (h), shows the best SK output on this image. (Here, "best" simply means the most visually pleasing result appearing in SK [20]; the image was kindly provided by the first author of that paper.)

Comparing the third and fourth rows of Figure 9, we see that the approach of this paper has segmented many more thin, elongated regions than SK. But this difference is of little interest—without specifying a particular application and associated error metric, one cannot say whether it is preferable to include elongated regions in the foreground or not. Of much more interest is the qualitative nature of the cameraman's boundary. We see that this paper's approach yields boundaries that are considerably more smooth and visually appealing than the SK output. This is particularly noticeable on the coat, right arm, and legs.

Note that the pixelation artifacts in panel (h) are not directly relevant to this discussion. When comparing panels (f) and (h), the relevant difference is the bumps in the boundary of (h) that have a scale of several pixels. The lack of pixelation in (f) is, of course, an advantage of the resolution-independent approach in general, but has no direct bearing on the curvature regularization being investigated here.

7 Conclusion

The key contribution of the paper was the derivation of a novel corner energy (16), used to regularize curvature in resolution-independent images modeled by piecewise linear triangle meshes. Experiments showed qualitative and quantitative improvements in the accuracy of resolution-independent images inferred using the new curvature regularizer. The technique also compared favorably with a state-of-the-art approach for binary segmentation. The clearest opportunity for future work is to incorporate an energy term for the ridge set. It may also be possible to reduce the computational expense of the approach by employing different techniques for mesh generation (e.g. [1,17,21]).

Acknowledgments.

The first author is grateful to Microsoft Research, UK (where this work was conducted) for support as a Visiting Researcher. We are indebted to Chris Francese for assistance with translating Euler's Latin.

References

1. Adams, M.D.: An improved content-adaptive mesh-generation method for image representation. In: Proc. ICIP (2010)
2. Chan, T.F., Kang, S.H., Shen, J.: Euler's elastica and curvature-based inpainting. SIAM Journal on Applied Mathematics 63(2), 564–592 (2002)
3. Euler, L.: Methodus inveniendi lineas curvas maximi minimive proprietate gaudentes. Bousquet (1744)
4. Goldluecke, B., Cremers, D.: Introducing total curvature for image processing. In: Proc. ICCV. pp. 1267–1274 (2011)
5. Kline, M.: Calculus: An Intuitive and Physical Approach. Dover, 2 edn. (1998)
6. Koenderink, J.J., van Doorn, A.J.: Surface shape and curvature scales. Image and Vision Computing 10(8), 557–564 (1992)
7. MacCormick, J., Fitzgibbon, A.: Curvature regularization for resolution-independent images. In: Proc. 9th Intl. Conf. Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR'13). Springer (2013)
8. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. ICCV. pp. 416–423 (2001)
9. Masnou, S., Morel, J.M.: Level lines based disocclusion. In: Proc. ICIP. pp. 259–263 (1998)
10. Mitiche, A., Ben Ayed, I.: Variational and Level Set Methods in Image Segmentation. Springer (2011)
11. Mumford, D.: Elastica and computer vision. In: Bajaj, C. (ed.) Algebraic Geometry and Its Applications, pp. 491–506. Springer (1994)
12. Mumford, D., Shah, J.: Optimal approximations by piecewise smooth functions and associated variational problems. Communications on Pure and Applied Mathematics 42(5), 577–685 (1989), `http://dx.doi.org/10.1002/cpa.3160420503`
13. Nitzberg, M., Mumford, D.: The 2.1-D sketch. In: Proc. ICCV. pp. 138–144 (dec 1990)
14. Nocedal, J., Wright., S.J.: Numerical Optimization. Springer Verlag (1999)
15. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer (2003)
16. Paul Chew, L.: Constrained delaunay triangulations. Algorithmica 4, 97–108 (1989)
17. Sarkis, M., Diepold, K.: Content adaptive mesh representation of images using binary space partitions. IEEE Trans. Image Processing 18(5), 1069–1079 (2009)
18. Schoenemann, T., Kahl, F., Cremers, D.: Curvature regularity for region-based image segmentation and inpainting: A linear programming relaxation. In: Proc. ICCV. pp. 17–23 (2009)
19. Sethian, J.A.: Level Set Methods and Fast Marching Methods. Cambridge University Press, 2nd edn. (1999)
20. Strandmark, P., Kahl, F.: Curvature regularization for curves and surfaces in a global optimization framework. In: Proc. EMMCVPR. pp. 205–218 (2011)
21. Vasilescu, M., Terzopoulos, D.: Adaptive meshes and shells: Irregular triangulation, discontinuities, and hierarchical subdivision. In: Proc. CVPR. pp. 829–832 (1992)
22. Viola, F.: Resolution-independent image models. Ph.D. thesis, University of Cambridge (2011)
23. Viola, F., Cipolla, R., Fitzgibbon, A.: A unifying resolution-independent formulation for early vision. In: Proc. CVPR (2012)
24. Ziemer, W.P.: Weakly Differentiable Functions: Sobolev Spaces and Functions of Bounded Variation. Springer (1989)

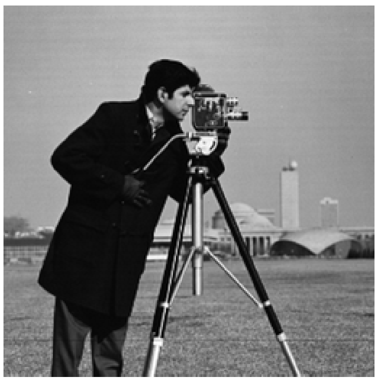
Appendix: Euler's definition of elastica

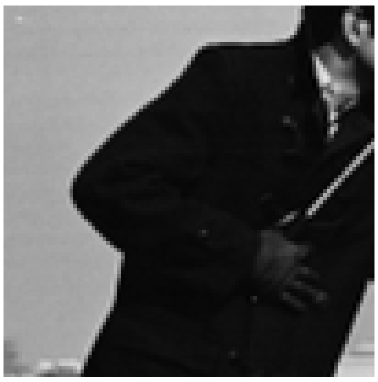
Numerous papers cite Euler's work on elastica, but it is surprisingly difficult to track down the relevant excerpt. It appears in Euler's 1744 publication [3], *Methodus inveniendi lineas curvas* . . . , Appendix I ("Additamentum I"), paragraph 2 (p247):

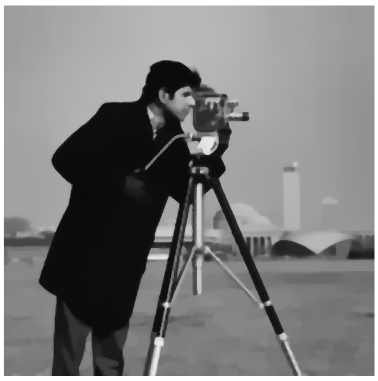
> . . . ut inter omnes curvus eiusdem longitudinis, qua non solum per puncta A & B transeant, sed etiam in his punctis a rectis positione datis tangantur, definiatur ea in qua sit valor huius expressionis $\int \frac{ds}{RR}$ minimus.

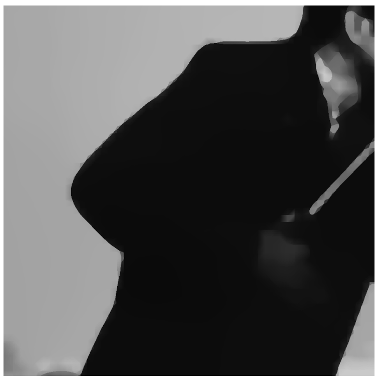
This can be translated as:

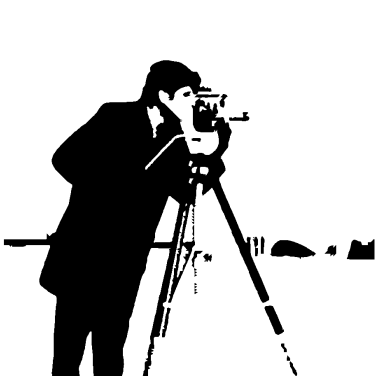
> . . . that, of all curves of the same length, which not only pass through points A and B, but also are touched at these points by given tangents, it is defined by that in which the value of the expression $\int \frac{ds}{R^2}$ is the smallest.

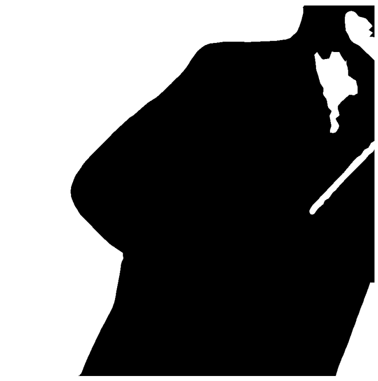

(a) input image (256 × 256)

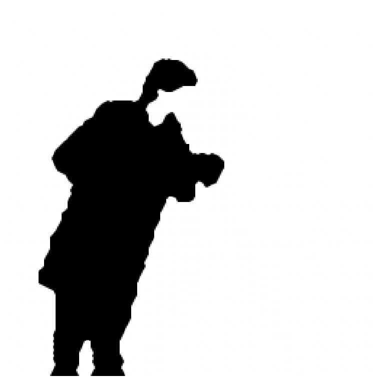

(b) detail of (a)


(c) resolution-independent image


(d) detail of (c)


(e) binarized version of (c)


(f) detail of (e)


(g) output of SK [20]

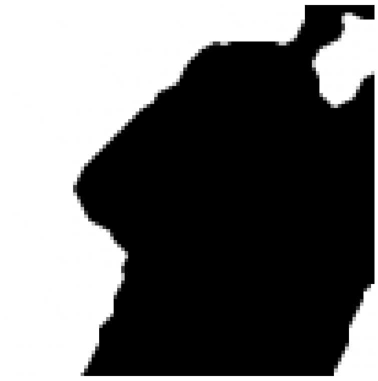

(h) detail of (g)

Fig. 9 Resolution-independent segmentation.