Curvature Filters Efficiently Reduce Certain Variational Energies

Yuanhao Gong, *Student Member, IEEE*, and Ivo F. Sbalzarini

***Abstract*— In image processing, the rapid approximate solution of variational problems involving generic data-fitting terms is often of practical relevance, for example in real-time applications. Variational solvers based on diffusion schemes or the Euler-Lagrange equations are too slow and restricted in the types of data-fitting terms. Here, we present a filter-based approach to reduce variational energies that contain generic data-fitting terms, but are restricted to specific regularizations. Our approach is based on reducing the regularization part of the variational energy, while guaranteeing non-increasing total energy. This is applicable to regularization-dominated models, where the data-fitting energy initially increases, while the regularization energy initially decreases. We present fast discrete filters for regularizers based on Gaussian curvature, mean curvature, and total variation. These pixel-local filters can be used to rapidly reduce the energy of the full model. We prove the convergence of the resulting iterative scheme in a greedy sense, and we show several experiments to demonstrate applications in image-processing problems involving regularization-dominated variational models.**

***Index Terms*— Approximation, filter, gaussian curvature, half-window regression, mean curvature, regularization, total variation, variational model.**

I. Introduction

VARIATIONAL models play a central role in image processing, as many tasks can be formulated in this framework, from denoising [1], registration [2], and enhancement [3], [4] to distortion removal [5], super-resolution [4], [6], multi-spectral reconstruction [7], and segmentation [8], [9]. In every variational formulation, one aims to find a minimizing function $\hat{U} = \arg\min_U \mathcal{E}(U)$ to an energy functional

$$\mathcal{E}(U) = \mathcal{E}_{\Phi_0}(U, I) + \lambda \mathcal{E}_{\Phi_1}(U)\,. \tag{1}$$

The total energy $\mathcal{E} \geq 0$ is composed of two parts: the data-fitting energy $\mathcal{E}_{\Phi_0}(U, I) \geq 0$ measuring how well U fits the image data I (typically a norm or divergence [9]), and the regularization energy $\mathcal{E}_{\Phi_1}(U) \geq 0$ formalizing prior knowledge about U. These two contributions are weighted with the scalar regularization coefficient $\lambda > 0$. All terms are non-negative, as they relate to negative logarithms of probabilities in Bayesian inference.

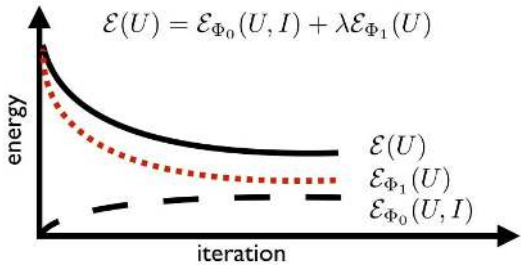


Fig. 1. In regularization-dominated variational models, the regularization energy initially dominates the reduction in the total energy, while the data-fitting energy initially increases when starting from the input image as initial condition.

Finding the globally optimal solution of such a variational problem can be challenging and is often not required in a particular practical application. Rather, in some applications it is important to obtain approximations rapidly. Given a variational model, the task then is to rapidly *reduce* the variational energy $\mathcal{E}$, without necessarily *solving* the model.

Here, we present a discrete filtering approach to efficiently compute reduced-energy images for regularization-dominated variational models with curvature or total-variation (TV) regularization. We define *regularization-dominated* variational models as those where the regularization term is the dominant initial contribution to the total energy. This is the case whenever the initial U, U_0, is identical to the input image, i.e., $U_0 = I$. Starting from the data themselves as initialization, $\mathcal{E}_{\Phi_0}(U_0 = I, I) = 0$ initially. The initial total energy hence only consists of the regularizer. Reducing the regularizer, the data-fitting term initially increases or remains constant, since it cannot become negative. Any reduction in $\mathcal{E}$ hence initially comes from the regularizer, as illustrated in Fig. 1

This motivates the design of fast filters that minimize the regularizer. Repeatedly iterating such filters over an image successively reduces the regularization energy. The data-fitting energy is evaluated in order to choose filter moves that guarantee non-increasing total energy. Information from the data-fitting term is hence appropriately considered. The filter iterations are stopped as soon as the total energy does no longer decrease. Since the filters are point-wise, and the data-fitting energy need not be monotonic, convergence need not be to a (local) minimum of the total energy. It is guaranteed, however, that the final $\mathcal{E}(U) \leq \mathcal{E}(U_0)$.

Manuscript received September 30, 2016; revised January 11, 2017; accepted January 20, 2017. Date of publication January 25, 2017; date of current version February 17, 2017. This work was supported in part by the Max Planck Society and in part by the German Federal Ministry of Education and Research under Grant 031A099. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Peter Tay.

Y. Gong and I. F. Sbalzarini are with the Chair of Scientific Computing for Systems Biology, Faculty of Computer Science, TU Dresden, 01187 Dresden, Germany, and also with the MOSAIC Group, Center for Systems Biology Dresden, Max Planck Institute of Molecular Cell Biology and Genetics, 01307 Dresden, Germany (email: gongyuanhao@gmail.com; ivos@mpi-cbg.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2658954

We present such filters for Gaussian curvature (GC) [2], [10]–[15], mean curvature (MC) [12], [16]–[19], and TV regularizers [20]. The TV regularizer formalizes the assumption that U is a piecewise constant function over the image domain. MC assumes U to define a minimal surface, and GC favors piecewise developable U. These three regularizers hence constitute a hierarchy of function spaces of increasing complexity. Since the data-fitting term only needs to be evaluated, the present filters work with generic data-fitting terms, as long as they are computable. The filters use half-window regression in order to preserve edges in the image and are pixel-local, meaning they only involve nearest-neighbor pixel operations.

The conceptual idea of the present filters is to use the pixel-local analytical solutions of the regularizer as a projection operation. Locally approximating the image by a constant (TV), minimal (MC), or developable (GC) surface reduces the regularization energy. All possible constant, minimal, or developable approximations in a local 3×3 pixel neighborhood can be enumerated. We then choose the one that leads to the smallest intensity change in the center pixel, i.e., the smallest increase in data-fitting energy. This is inspired by the iterative reweighting scheme [21], but avoids the construction of the sparse matrix required therein. Therefore, the present local projections are fast and data-independent.

In summary, we present fast data-local discrete filters for *reducing* the energy of regularization-dominated variational models containing generic (potentially black-box) data-fitting terms, but regularizers of the type TV, MC, or GC. We consider the following properties of the present approach appealing in applications that require approximate solutions:

- ⋆ The filters do not need the gradient of the energy and can therefore handle generic data-fitting terms, as long as they can be evaluated point-wise.
- ⋆ They do not assume differentiability of the signal. Therefore, edges are preserved.
- ⋆ They have linear algorithmic complexity in the number of image pixels and only require as much memory as the image, plus 17 numbers.

The filters are easy to implement and parallelize, and they are parameter-free. All algorithms presented here have been implemented in MATLAB (40 lines), Java, and C++ (100 lines). The software and source code are publicly available from the MOSAIC Group's web site.

A. Organization of the Paper

Section II introduces the notation and basic concepts. In Sections III to V, we derive and present the filters to locally minimize (over the image domain) GC, MC, and TV regularization energies. In Section III, we consider GC regularizers, embodying the prior that U is piecewise developable. In Section IV, we analogously present a filter to reduce MC, corresponding to the prior that U is a minimal surface. In Section V, we present the filter for TV regularization, assuming U to be piecewise constant. In Section VI, according to the concept illustrated in Fig. 1, these three filters are then used as basic operators to reduce the total energy $\mathcal{E}(U)$ of variational models. We show that the resulting algorithms can handle generic data-fitting terms, as illustrated in cases that cannot be handled by traditional solvers.

II. Mathematical Framework

Let $\vec{x} = (x, y) \in \Omega$ denote the spatial coordinate and Ω the 2D image domain. Let $I(i, j) : \mathbb{Z}_0^+ \times \mathbb{Z}_0^+ \mapsto [0, 1]$ be the given discrete digital image with integer pixel indices i and j and continuous intensity $I \in [0, 1]$. Let $U(\vec{x}) \in [0, 1]$ denote the current reconstruction, i.e., the image obtained by reducing the total energy in Eq. 1 starting from $U_0 = I$.

We interpret U and I as geometric surfaces over Ω, i.e., $\Psi(\vec{x}) = (\vec{x},\ U(\vec{x}))$. From this, curvature can be computed by taking partial derivatives. For GC we have:

$$K(U(\vec{x})) = \frac{U_{xx}U_{yy} - U_{xy}^2}{(1 + U_x^2 + U_y^2)^2}\,, \tag{2}$$

where subscripts denote partial derivatives. Recall that the total GC (i.e., the integral of K) of any surface is related to the surface's topology through the Gauss-Bonnet theorem. Since total GC is therefore a topological invariant, one can only minimize total *absolute* GC [10]–[14].

The total absolute GC regularizer is

$$\mathcal{E}_{\Phi_1}^{\mathrm{GC}}(U) = \int_\Omega |\kappa_1\kappa_2|\,\mathrm{d}\vec{x} = \int_\Omega |K(U)|\,\mathrm{d}\vec{x}\,, \tag{3}$$

where κ_1 and κ_2 are the two principal curvatures of Ψ. Analogously, the regularization energy for MC is:

$$\mathcal{E}_{\Phi_1}^{\mathrm{MC}}(U) = \int_\Omega \left|\frac{\kappa_1 + \kappa_2}{2}\right| \mathrm{d}\vec{x} = \int_\Omega |H(U)|\,\mathrm{d}\vec{x}\,, \tag{4}$$

with the MC H computed from U as [22]:

$$H(U) = \frac{(1 + U_y^2)U_{xx} - 2U_xU_yU_{xy} + (1 + U_x^2)U_{yy}}{2(1 + U_x^2 + U_y^2)^{3/2}}\,. \tag{5}$$

The MC is hence related to the Laplace operator ∇^2 as:

$$H(U) = \frac{\nabla^2 U}{2(1+U_x^2 + U_y^2)^{1/2}} - \frac{U_y^2U_{yy} + 2U_xU_yU_{xy} + U_x^2U_{xx}}{2(1 + U_x^2 + U_y^2)^{3/2}}, \tag{6}$$

which relates MC regularization to gradient-adaptive diffusion along intensity edges in the image.

Finally, the regularization energy for TV is classical:

$$\mathcal{E}_{\Phi_1}^{\mathrm{TV}}(U) = \|\nabla U\|_p\,, \tag{7}$$

where $\|\cdot\|_p$ is any L^p norm and ∇ is the gradient operator.

For these regularizers (i.e., GC, MC, and TV) all minimizers $\hat{S}_i(\vec{x}), i = 1, \ldots, N_s$ in a local 3×3 pixel neighborhood around pixel $\vec{x}$ can be enumerated, as shown below. From this finite set, we choose the element that leads to the smallest change in the intensity of pixel $\vec{x}$:

$$S_m(\vec{x}) = \arg\min_{\hat{S}_i(\vec{x})} |\hat{S}_i(\vec{x}) - U(\vec{x})| \tag{8}$$

and update $\hat{U}(\vec{x}) = S_m(\vec{x})$. When used together with a data-fitting term, an additional condition is added (Eq. 16) in order to ensure non-increasing total energy. This is detailed in Section VI. In the sections until then, we describe the filters by considering the regularizer alone, without data-fitting term.

III. Gaussian Curvature Filter

We first derive the most general filter, the one for reducing GC. Surfaces with zero GC can be isometrically mapped onto a plane without distortion, which is why they are called *developable*. Minimizing GC is hence equivalent to making the image "as developable as possible". The fact that GC is intrinsic and related to developability renders it a desirable regularizer, as has, e.g., been noted in image registration [2].

Despite its desirability, using GC as a regularizer has been hampered by two issues: First, the algorithms available for minimizing GC converge slowly, since they are based on diffusion flows. Second, GC needs to be explicitly computed in every iteration of the algorithm, which is only possible if the image is at least twice differentiable (Eq. 2).

The present GC filter relaxes these two limitations. The filter converges orders of magnitude faster than previous methods and reduces GC *without* explicitly computing it. This relaxes the differentiability assumption about the image. We provide proofs of the filter's correctness and convergence and demonstrate its performance in image processing. We first prove a theorem that guarantees that our filter reduces total absolute GC. Then, we show a domain-decomposition technique that removes dependencies between neighboring pixels. This increases computational efficiency of the filter. Finally, we prove convergence of the total absolute GC to a local minimum over filter iterations.

A. Theoretical Foundation

Any developable surface $\mathcal{S}$ can be locally approximated by its tangent plane $\mathcal{TS}$. Then, the following holds:

Theorem 1:

$$\forall \vec{x} \in \mathcal{S},\ \forall \epsilon > 0,\ \exists \vec{x}_0 \in \mathcal{S} \quad \text{s.t. } 0 < |\vec{x} - \vec{x}_0| < \epsilon \text{ and } \vec{x} \in \mathcal{TS}(\vec{x}_0). \tag{9}$$

Proof: Let $\vec{x} = \vec{r}(u, v) \in \mathcal{S}$, where (u, v) is the parametric coordinate. Since $\mathcal{S}$ is developable, $\vec{r}(u, v)$ can be represented as $\vec{r}(u, v) = \vec{r}_A(u) + v\vec{r}_B(u)$ [23], where $\vec{r}_A(u)$ is the directrix and $\vec{r}_B(u)$ is a unit vector. Let $\vec{x}_0 = \vec{r}_0 = \vec{r}(u, v_0) \in \mathcal{S}$, where $v_0 = v + \epsilon$ and $\epsilon \neq 0$. The tangent plane at $\vec{x}_0$ can be represented with two scalars α_1 and α_2:

$$\begin{aligned}\mathcal{TS}(\vec{x}_0) &= \vec{r}_0 + \alpha_1 \frac{\mathrm{d}\vec{r}}{\mathrm{d}u} + \alpha_2 \frac{\mathrm{d}\vec{r}}{\mathrm{d}v} \\ &= \vec{r} + \epsilon\vec{r}_B + \alpha_1 \frac{\mathrm{d}\vec{r}}{\mathrm{d}u} + \alpha_2 \vec{r}_B \\ &= \vec{r} + \alpha_1 \frac{\mathrm{d}\vec{r}}{\mathrm{d}u} + (\alpha_2 + \epsilon)\vec{r}_B. \end{aligned} \tag{10}$$

Therefore, $\vec{x}$ is on the plane that contains $\vec{x}_0$ and is spanned by the vectors $\frac{\mathrm{d}\vec{r}}{\mathrm{d}u}$ and $\vec{r}_B$. Hence, $\vec{r} \in \mathcal{TS}(\vec{x}_0)$. ■

In differential geometry, GC of a 2D surface is defined as $\kappa_1\kappa_2$, where $\kappa_{1,2}$ are the principal curvatures. Equation 9 requires that one principal curvature of any developable surface must be zero. Therefore, *minimizing one of the principal curvatures is equivalent to minimizing GC.* This is why our GC filter does not need to explicitly compute GC. Thanks to

$$\kappa_1\kappa_2 = 0 \Longleftrightarrow \min\{|\kappa_1|, |\kappa_2|\} = 0\,, \tag{11}$$

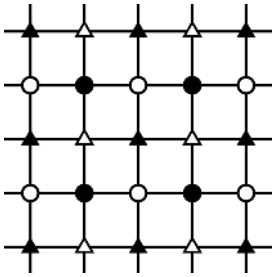


Fig. 2. Illustration of the minimum vertex coloring domain decomposition. The pixels are decomposed into four neighborhood-disjoint sets: black circles B_C, black triangles B_T, white circles W_C, and white triangles W_T.

we instead minimize the principal curvature of smaller absolute value, in order to apply the least change to the image.

B. Algorithm

We design the filter algorithm with computational efficiency and data locality in mind, as described below.

1) Domain Decomposition: Locally minimizing the smaller absolute principal curvature is hampered by dependencies between neighboring pixels [21]. We propose here a domain-decomposition algorithm to circumvent these dependencies.

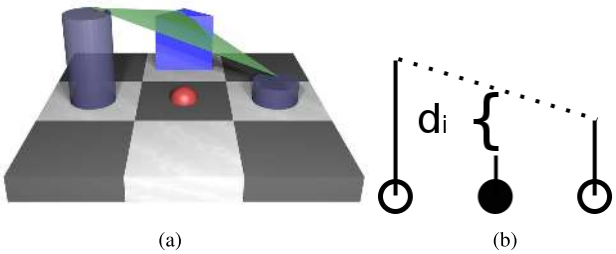
We use a checkerboard decomposition of the discrete pixel domain into two disjoint subsets: the "white" points W and the "black" points B. We further split each of these two subsets into triangles and circles, leading to: white triangles W_T, white circles W_C, black triangles B_T, and black circles B_C. This decomposition guarantees that neighboring pixels in a 4-connected neighborhood are in different subsets (i.e., is a minimum vertex coloring of the pixel-neighborhood graph), as illustrated in Fig. 2.

This decomposition has several benefits: First, it removes dependencies between neighboring pixels. For example, all pixels in B_C can be updated simultaneously. Second, thanks to this independence, the update can use neighbors that have already been updated. This guarantees convergence (see Sec. III-C). Third, in a 3×3 local window, all tangent planes $\mathcal{TS}$ can be enumerated. Therefore, proximal projection can be used to make the surface $U(\vec{x})$ more developable, which implies locally reducing GC.

According to Eq. 9, we need to project $U(\vec{x})$ to $\hat{U}(\vec{x})$ such that $\hat{U}(\vec{x})$ is on the closest tangent plane of any neighboring pixel. First, we show how to represent a tangent plane $\mathcal{TS}$. Then, we show how to do the projection.

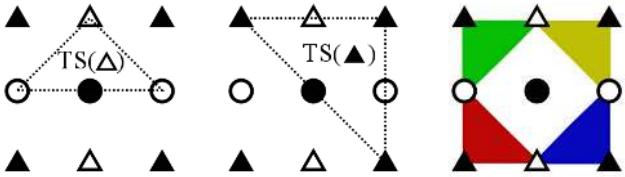
2) Discrete Representation of a Tangent Plane: There are several ways of representing the tangent planes $\mathcal{TS}$ through neighboring pixels, e.g., as triangles or rectangles. We use triangles because of their easy projection. This is illustrated in Fig. 3a. The tangent plane $\mathcal{TS}(\triangle)$ at the blue triangle cylinder is drawn in green. Projecting $U(\vec{x})$ (the red ball) onto the green tangent plane is the same as projecting it onto the edge of the green triangle that passes over it. Computing the distance d_i to this edge is hence sufficient (Fig. 3b).

3) Enumeration of all Projections: In order to find the tangent plane that has the smallest $|d_i|$, we enumerate all possible tangent triangles in a 3×3 pixel neighborhood $\mathcal{N}(\vec{x})$ around $\vec{x}$ that do not include $\vec{x}$ as a vertex (Fig. 4). There are



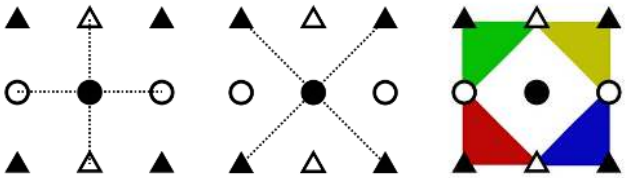
(a) (b)

Fig. 3. (a) A tangent plane $\mathcal{TS}$ (the green triangle) at the blue triangle pixel. Cylinder heights represent pixel intensities. (b) Illustration of the projection distance d_i of $U(x)$ (the red ball) onto the edge of the green triangle from (a) passing over it.



(a) $\mathcal{TS}(W)$ (upper) (b) $\mathcal{TS}(B)$ (upper-right) (c) $\mathcal{TS}(\blacktriangle)$

Fig. 4. The twelve possible tangent planes in a 3×3 neighborhood: (a) example $\mathcal{TS}$ for $\vec{x} \in W$; (b) example $\mathcal{TS}$ for $\vec{x} \in B$; (c) the four $\mathcal{TS}$ through mixed-color neighbors.



(a) $d_{1,2}$ for $\mathcal{TS}(W)$ (b) $d_{3,4}$ for $\mathcal{TS}(B)$ (c) $d_{5\ldots8}$ for $\mathcal{TS}(\blacktriangle)$

Fig. 5. The eight d_i to the tangent triangles passing over $\vec{x}$: (a) two to the common edges from the four $\mathcal{TS}(W)$; (b) two to the common edges from the four $\mathcal{TS}(B)$; (c) four to the tangent planes through mixed neighbors (no shared edges).

12 such triangles: four through each of the four white neighbors W (one of the four possibilities is shown in Fig. 4a), four through the four black neighbors B (one possibility is shown in Fig. 4b), and four through mixed black/white neighbors (all shown in Fig. 4c).

Since some of the 12 tangent triangles share common edges over $\vec{x}$, and projecting onto these edges is sufficient, there are only 8 different d_i: two to the common edges from W (Fig. 5a), two to the common edges from B (Fig. 5b), and four to the tangent planes through the mixed neighbors (Fig. 5c).

4) Minimal Projection Operator: After computing all $\{d_i,\ i = 1, \ldots, 8\}$, we use the smallest absolute distance as the minimum projection of the current intensity $U(\vec{x})$ to the target intensity $\hat{U}(\vec{x})$, guaranteeing that $\hat{U}(\vec{x})$ is on a tangent plane through a neighboring pixel. More specifically, we find d_m such that $|d_m| = \min\{|d_i|,\ i = 1, \ldots, 8\}$. Then, we let $\hat{U}(\vec{x}) = U(\vec{x}) + d_m$. We denote the resulting pixel-local update by $\mathcal{P}_g$. This compact operator is summarized in Algorithm 1.

Algorithm 1 Minimal Projection Operator $\mathcal{P}_g$

Input: $U(i,j)$

$d_1 = (U(i-1,j) + U(i+1,j))/2 - U(i,j)$

$d_2 = (U(i,j-1) + U(i,j+1))/2 - U(i,j)$

$d_3 = (U(i-1,j-1) + U(i+1,j+1))/2 - U(i,j)$

$d_4 = (U(i-1,j+1) + U(i+1,j-1))/2 - U(i,j)$

$d_5 = U(i-1,j) + U(i,j-1) - U(i-1,j-1) - U(i,j)$

$d_6 = U(i-1,j) + U(i,j+1) - U(i-1,j+1) - U(i,j)$

$d_7 = U(i,j-1) + U(i+1,j) - U(i+1,j-1) - U(i,j)$

$d_8 = U(i,j+1) + U(i+1,j) - U(i+1,j+1) - U(i,j)$

find d_m, such that $|d_m| = \min\{|d_i|, i = 1, \ldots, 8\}$

Output: $\hat{U}(i,j) = U(i,j) + d_m$

Algorithm 2 Gaussian Curvature Filter $\mathcal{G}_c$

Input: $U(i,j)$

$\forall \vec{x} \in B_C$, $\mathcal{P}_g(U(\vec{x}))$

$\forall \vec{x} \in B_T$, $\mathcal{P}_g(U(\vec{x}))$

$\forall \vec{x} \in W_C$, $\mathcal{P}_g(U(\vec{x}))$

$\forall \vec{x} \in W_T$, $\mathcal{P}_g(U(\vec{x}))$

Output: $U(i,j)$

The set $\{d_i,\ i = 1, \ldots, 8\}$ is a complete description of the local discrete geometry at $\vec{x}$. For any given $U(\vec{x})$ and its $\{d_i,\ i = 1, \ldots, 8\}$, $U(\mathcal{N}(\vec{x}))$ can be obtained by solving a linear system of equations. Geometrically, d_i is the directional curvature in the corresponding direction. This follows from the Euler theorem, $d_i \approx \kappa_1 \cos^2\theta_i + \kappa_2 \sin^2\theta_i$, where κ_1, κ_2 are the principle curvatures and θ_i is the angle to the principle plane. Therefore, if the angular sampling is dense enough in $[-\pi, \pi]$, we have $|d_m| \approx \min\{|\kappa_i|\}$ when $\kappa_1\kappa_2 \geq 0$. Even though we only have eight angles in the local window, we can still use d_m as a *discrete approximation* of the minimal absolute principle curvature on the pixel grid. Updating U with d_m hence reduces $\min\{|\kappa_1|, |\kappa_2|\}$, as desired [22].

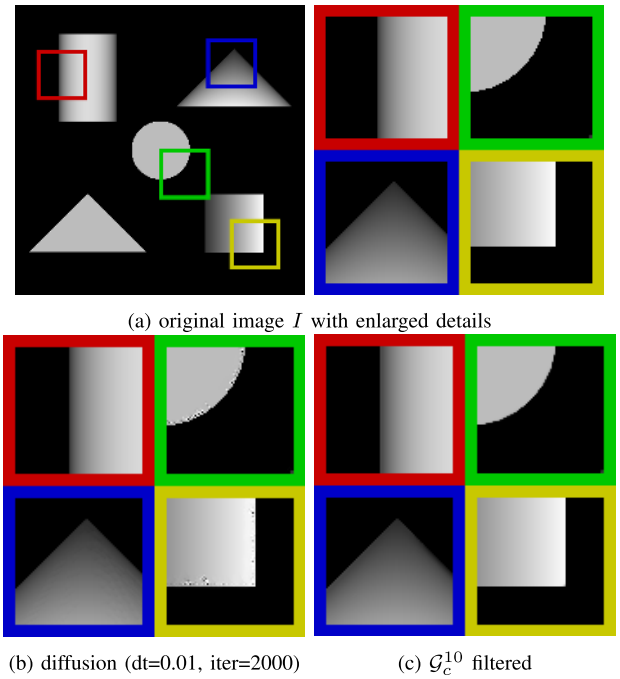
5) Gauss Curvature Filter: We iterate $\mathcal{P}_g$ over all pixels in each of B_T, B_C, W_T, and W_C. Since the pixels within each set are independent, the iteration order does not matter and the projections can be applied in parallel. This yields our Gaussian curvature filter $\mathcal{G}_c$, as summarized in Algorithm 2.

C. Convergence

We prove convergence of $\mathcal{G}_c$ to a local minimum of the total absolute GC. For this, we first show that $\mathcal{P}_g$ reduces the absolute local GC $|K(U)|$. Then, we show that iterating this reduces the total absolute GC energy of Eq. 3.

1) Convergence of $\mathcal{P}_g$: $|K(\mathcal{P}_g(U(\vec{x})))| \leq |K(U(\vec{x}))|,\ \forall \vec{x}$. *Proof:* we only prove this theorem for $\vec{x} \in B_T$. Similar proofs also hold for B_C, W_C, and W_T.

$\forall \vec{x}_1, \vec{x}_2 \in B_T$ and $\vec{x}_1 \neq \vec{x}_2$, $\mathcal{P}_g(U(\vec{x}_1)))$ is independent of $\mathcal{P}_g(U(\vec{x}_2)))$. Therefore, we only need to consider one projection $\mathcal{P}_g(U(\vec{x}_1)))$. $\forall \vec{x}_1 \in B_T$, if $|d_m(\vec{x}_1)| = 0$, then $\mathcal{P}_g(U(\vec{x}_1))) = U(\vec{x}_1)$. Therefore, $\mathcal{E}(\mathcal{P}_g(U)) = \mathcal{E}(U)$. Otherwise, $\exists \vec{x}_1 \in B_T$, s.t. $|d_m(\vec{x}_1)| \neq 0$. $\mathcal{P}_g(U(\vec{x}_1)))$ is on a tangent plane $\mathcal{TS}$ through one of its neighbors while $U(\vec{x}_1)$ is not



(a) original image I with enlarged details

(b) diffusion (dt=0.01, iter=2000) (c) $\mathcal{G}_c^{10}$ filtered

Fig. 6. Illustration on different developable surfaces. The diffusion process to reduce GC with interpolation correction [10] leads to artifacts, whereas the GC filter exactly preserves developable surfaces.

on any tangent plane. According to Eq. 9: $|K(\mathcal{P}_g(U(\vec{x}_1)))| < |K(U(\vec{x}_1))|$. ■

2) Convergence of $\mathcal{G}_c$: Based on the convergence of $\mathcal{P}_g$, we have $\mathcal{E}^{\mathrm{GC}}_{\Phi_1}(\mathcal{P}_g(U)) < \mathcal{E}^{\mathrm{GC}}_{\Phi_1}(U)$ (cf. Eq. 3). Let $\mathcal{G}_c^n$ denote n iterations of $\mathcal{G}_c$. For all $n_1, n_2 \in \mathbb{Z}^+$, $n_2 > n_1$, we have: $\mathcal{E}^{\mathrm{GC}}_{\Phi_1}(\mathcal{G}_c^{n_2}(U)) \leq \mathcal{E}^{\mathrm{GC}}_{\Phi_1}(\mathcal{G}_c^{n_1}(U)) \leq \mathcal{E}^{\mathrm{GC}}_{\Phi_1}(U)$.

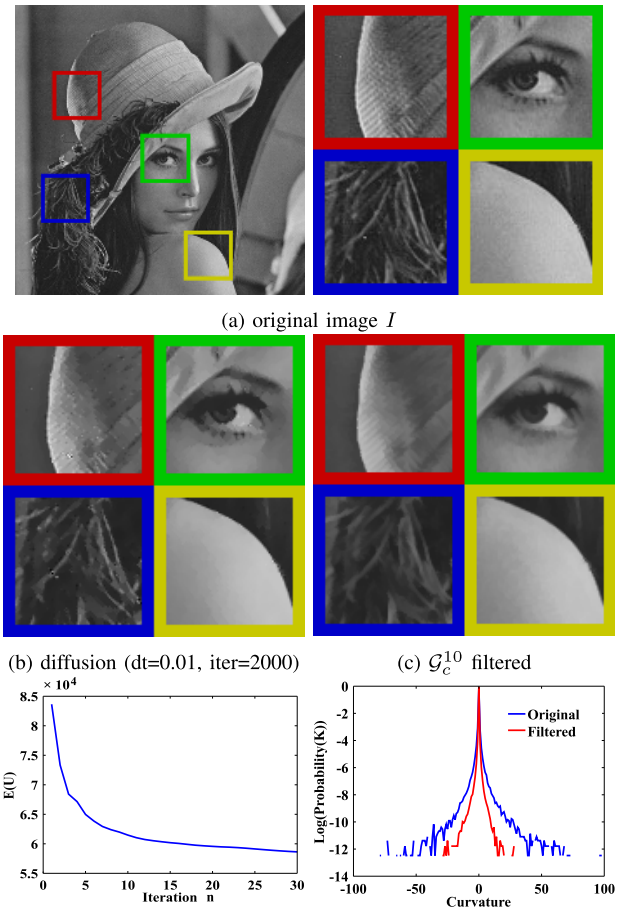
In words, the total absolute GC energy $\mathcal{E}^{\mathrm{GC}}_{\Phi_1}(\mathcal{G}_c^n(U))$ is monotone with respect to n. At the same time, it has the obvious lower bound $\mathcal{E}^{\mathrm{GC}}_{\Phi_1}(U) \geq 0$. According to the Monotone Bounded Theorem, we thus have: $\mathcal{G}_c$ converges to a (local) minimum of $\mathcal{E}^{\mathrm{GC}}_{\Phi_1}$ over the domain Ω.

D. Properties of the Filter

The most important property of the present GC filter is that it preserves developable surfaces: If $K(U) = 0$, then $\mathcal{G}_c(U) = U$. This means that the filter leaves all parts of the image unchanged that correspond to a developable surface in intensity space. An example is shown in Fig. 6, compared with a diffusion approach to reducing GC (without data-fitting term). Run until convergence, diffusion generates artifacts at the edges, even when using interpolation correction [10]. The GC filter exactly preserves the developable surfaces.

The present GC filter is parameter-free. This is a key difference to other edge-preserving filters, which require parameters to be tuned by the user, such as in the Bilateral Filter (BF) [24], Guided Filter (GF) [25], CLMF [26], and RTV [27].

The runtime complexity of $\mathcal{G}_c$ is in $O(N)$, where N is the total number of pixels. The pre-factor is 25, i.e., 25 operations are required per pixel. This is less than the pre-factor in diffusion methods, which typically is 176 [10].



(a) original image I

(b) diffusion (dt=0.01, iter=2000) (c) $\mathcal{G}_c^{10}$ filtered

(d) $\mathcal{E}^{\mathrm{GC}}_{\Phi_1}$ evolution under $\mathcal{G}_c^n$ (e) Curvature histograms

Fig. 7. GC filter on the *Lena* image, compared with the original image and the result from GC-reducing diffusion with correction [10]. The diffusion was run until convergence, such that further iterations do not change the result. $\mathcal{E}^{\mathrm{GC}}_{\Phi_1}(\mathcal{G}_c^n(I))$ vs. n for is shown in (d). (e) Histogram of G in log scale before and after $\mathcal{G}_c^{10}$ filtering.

E. Experiments

We exemplify the use of the GC filter in image smoothing and denoising. In all denoising examples, we report the peak signal-to-noise ratio (PSNR) as a distribution-independent measure of noise level. In the image-smoothing examples, no noise is added.

1) Illustration: For any image, the GC filter reduces the total absolute GC, as shown in Fig. 7. Since GC alone tunes smoothing based on image contents, one of the simplest applications for a GC filter is image denoising, which we use here as an illustrative example (Fig. 8) for different types of noise. Ten iterations of the GC filter are enough in practice. The evolution of the regularizer energy $\mathcal{E}^{\mathrm{GC}}_{\Phi_1}$ for the noisy *Lena* image is shown in Fig. 8(e) next to the curvature histograms before and after $\mathcal{G}_c^{10}$ filtering. The corresponding PSNR are given in Table I as compared with the diffusion-solver result. There is no data-fitting term considered here, as we simply iterate the filter in order to reduce the GC regularizer.

Since the GC filter preserves developable surfaces, it is sufficient to almost perfectly denoise images that only contain developable surfaces. No data-fitting term is needed in this

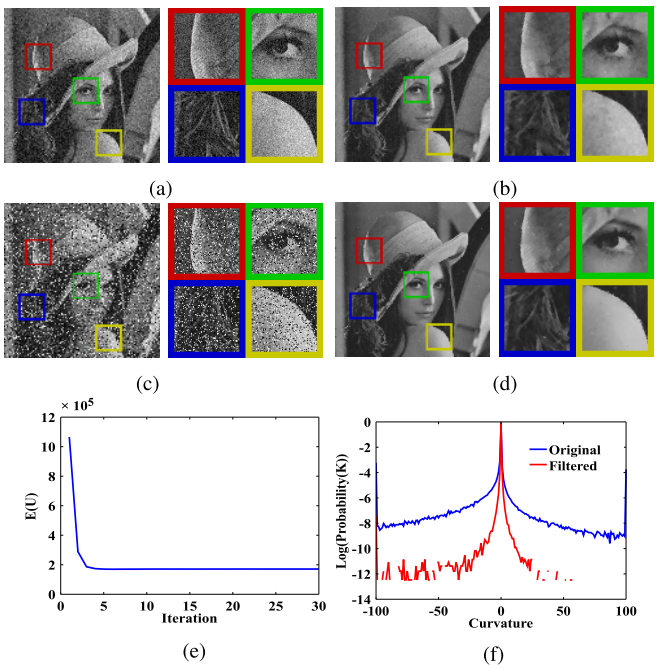


Fig. 8. Ten iterations of $\mathcal{G}_c$ for different noise types. (a) Gaussian noise, PSNR=21.6. (b) $\mathcal{G}_c^{10}$ filtered, PSNR=29.1. (c) Salt+Pepper noise, PSNR=12.3. (d) $\mathcal{G}_c^{10}$ filtered, PSNR=31.7. (e) $\mathcal{E}_{\Phi_1}^{\mathrm{GC}}$ evolution of (c)/(d). (f) Curvature histograms of (c)/(d).

TABLE I

PSNR OF THE NOISY AND FILTERED *Lena* IMAGES, COMPARING THE PRESENT GC FILTER WITH A TRADITIONAL DIFFUSION SCHEME [10] RUN TO STEADY STATE

noise type	Gaussian	Salt+Pepper
noisy image	21.6	12.3
diffusion	28.2	29.8
after $\mathcal{G}_c^{10}$	29.2	31.7

case. This is illustrated in Fig. 9 and compared with the results from a split-Bregman iteration solver [28] for the L2-TV-L1 model including an L^2 data-fitting term. Due to its piecewise constant assumption, the TV regularizer generates staircase artifacts when the regularization coefficient λ in the L2-TV-L1 model is large. Reducing the regularization coefficient, however, makes the denoising fail. The GC filter preserves the developable surfaces and removes noise efficiently without requiring a regularization parameter to be tuned.

Table II compares the runtimes of the GC filter with those of a split-Bregman solver [28] for the L2-TV-L1 model, a diffusion model [10], and iteratively reweighted GC [21]. The present filter is two to three orders of magnitude faster than the fastest solver.

2) Benchmark on a Standard Dataset: We compare the GC filter with the diffusion scheme from [10] on the 500-image Berkeley Segmentation Dataset and Benchmark (BSDS500) [29]. We define the metric $T = \mathcal{E}_{\Phi_1}^{\mathrm{GC}}(\hat{U})/\mathcal{E}_{\Phi_1}^{\mathrm{GC}}(I)$, where $\hat{U}$ and I are the final and original images, respectively. Lower T is better. For the GC filter, we always perform 10 iterations. For the diffusion, we use $dt = \frac{1}{16}$ and successively increase the iteration number from 300 to 1200 in order to check for convergence. The cumulative distributions

TABLE II

RUNTIMES IN SECONDS ON A MACBOOK PRO (2 GHZ INTEL CORE I7) FOR DIFFERENT IMAGE SIZES. ALL METHODS ARE IMPLEMENTED IN C++ WITH OPENCV. WE USE THE SPLIT-BREGMAN SOLVER FROM [28] WITH MAX ITERATION=80, $\epsilon = 1$. FOR THE DIFFUSION SCHEME, WE USE THE SOLVER FROM [10] WITH ITERATION=2000, $dt = 0.01$

Image	Method	64×64	128×128	256×256
Lena	[28]	2.187	2.354	6.041
	[10]	0.073	0.301	1.438
	[21]	0.025	0.131	0.596
	$\mathcal{G}_c^{10}$	0.000138	0.00055	0.0022
Camera-man	[28]	0.941	2.892	5.983
	[10]	0.078	0.316	1.465
	[21]	0.025	0.131	0.596
	$\mathcal{G}_c^{10}$	0.000138	0.00055	0.0022

of T are shown in Fig. 10, where color indicates the iteration numbers of the diffusion scheme, and the red line is for the GC filter. The average runtimes per image for the GC filter with 10 iterations and the diffusion scheme with 600 iterations are 0.005 and 1.17 seconds, respectively. This shows that the GC filter is on average two orders of magnitude faster in reaching similar energy levels.

IV. MEAN CURVATURE FILTER

The concept of the GC filter can also be extended to mean curvature (MC). MC regularizers are increasingly popular, as they amount to data-adaptive diffusion with low smoothing in areas of high intensity gradients, and vice versa [30]–[33]. The MC regularization term has also recently been proven to be convex [34].

Typically, MC models have the form

$$\mathcal{E}(U) = \|U(\vec{x}) - I(\vec{x})\|_\star + \lambda \int_\Omega |H(\vec{x})|^q \, \mathrm{d}\vec{x}, \qquad (12)$$

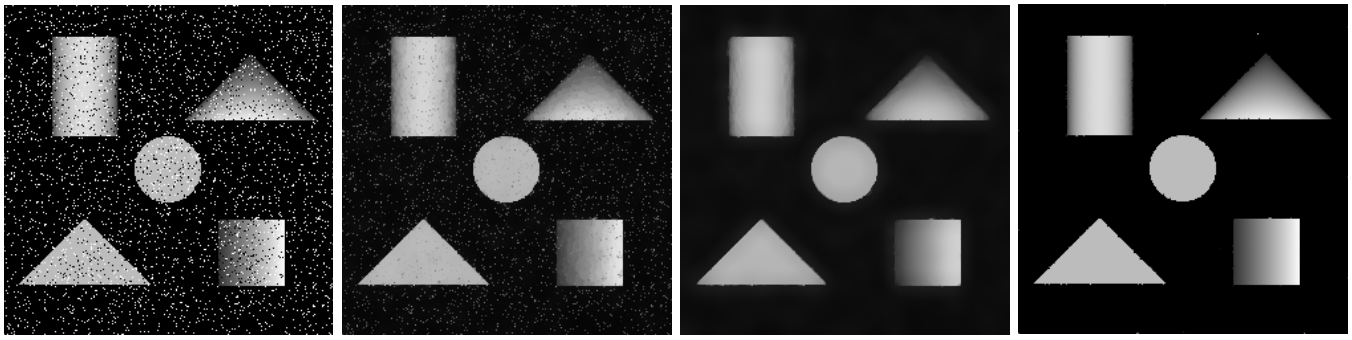
where $\|\cdot\|_\star$ is a proper norm, λ is a regularization coefficient, H is the mean curvature, and $q > 0$ is a positive real number. We use $q = 1$ here. This model can be solved by gradient decent, augmented Lagrangian methods [30], [31], or fixed point methods [32]. These solvers require differentiability of the norm $\|\cdot\|_\star$ and explicit computation of H. Recent methods relax the norm issue by introducing an auxiliary variable.

We avoid both of the above issues by exploiting the equivalence between piecewise linearity of U and minimal MC, which directly follows from Bernstein's Theorm:

Theorem 2: Given an image surface $\Psi(\vec{x}) = (\vec{x}, U(\vec{x})) \in \mathbb{R}^n$ where $n \leq 8$, if Ψ is a minimal surface, then $U(\vec{x})$ is a linear function. This statement becomes false when $n > 8$.

We hence minimize MC without explicitly computing it by making $U(\vec{x})$ "as piecewise linear as possible". We use the local directional curvatures $\{d_i\}$ to construct such a filter, leading to the approximation [22, eq. (6.12)]:

$$H \approx \frac{1}{8}\sum_{i=1}^{8} d_i = \begin{bmatrix} -\frac{1}{16} & \frac{5}{16} & -\frac{1}{16} \\ \frac{5}{16} & -1 & \frac{5}{16} \\ -\frac{1}{16} & \frac{5}{16} & -\frac{1}{16} \end{bmatrix} * U, \qquad (13)$$



(a) (b) (c) (d)

Fig. 9. Comparison of TV regularization and $\mathcal{G}_c$ for the denoising of developable surfaces. (a) Noisy developable surface image (Gaussian noise, PSNR=13.3). (b) Result from split-Bregman iteration [28] using the L2-TV-L1 model with λ=50, ϵ=0.01 (PSNR=21.6); (c) L2-TV-L1 with λ=200, ϵ=0.01 (PSNR=23.9). (d) Result after 10 iterations of the $\mathcal{G}_c$ filter without data-fitting term (PSNR=35.6). In L2-TV-L1, the small λ does not remove the noise completely, while the large λ generates staircase artifacts. The $\mathcal{G}_c$ filter removes the noise efficiently without generating artifacts and has no parameter to be tuned.

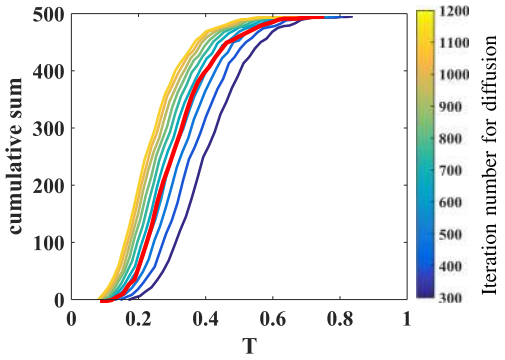


Fig. 10. Cumulative distributions of T over the 500 images of the BSDS500 benchmark for the present GC filter and the diffusion scheme from [10]. Colored curves are for different iteration numbers of the diffusion scheme. The red line is for the present GC filter. Lower T is better.

where $*$ denotes convolution. This follows from the Euler theorem (relation between directional curvature and MC) applied over the 8 directions of the d_i [22] and avoids explicit computation of the derivatives in Eq. 5.[1] When applied symmetrically using half-window regression in order to preserve edeges (cf. Eq. 6), it leads to the projection distances $\hat{d}_i$ given in Algorithm 3, which are different from the directional curvatures d_i. The MC filter $\mathcal{M}_c$ is in Algorithm 4. The convergence proof for $\mathcal{M}_c$ is analogous to that of $\mathcal{G}_c$ and is omitted here.

A. Experiments

State-of-the-art methods to minimize an MC regularizer require tens of seconds for images of size 256×256 [31], [32]. Our MC filter can perform six iterations on an image of size 400×400 in six milliseconds, which is four orders of magnitude faster. This difference is even larger on bigger images, because of the linear algorithmic complexity of $\mathcal{M}_c$. Six iterations are enough on our test images to pass the kink in the L-shaped energy evolution curve.

An example of applying increasing iterations of the MC filter is shown in Fig. 11. The energy evolutions for two tested images, *Lena* and *Barbara*, are shown in Fig. 12. We further test this filter on the complete BSDS500 dataset. The distribution of energy ratios between filtered and original images is shown in Fig. 13a. The runtime distribution on the complete BSDS500 dataset is shown in Fig. 13b.

[1]When *evaluating* the MC energy (Eq. 4) in our benchmarks, however, we still use the computation according to Eq. 5.

Algorithm 3 Projection Operator $\mathcal{P}_m$

Input: $U(i,j)$

1: $\hat{d}_1 = \frac{5}{16}(U(i-1,j)+U(i+1,j)) + \frac{5}{8}U(i,j+1) - \frac{1}{8}(U(i-1,j+1)+U(i+1,j+1)) - U(i,j)$

2: $\hat{d}_2 = \frac{5}{16}(U(i-1,j)+U(i+1,j)) + \frac{5}{8}U(i,j-1) - \frac{1}{8}(U(i-1,j-1)+U(i+1,j-1)) - U(i,j)$

3: $\hat{d}_3 = \frac{5}{16}(U(i,j-1)+U(i,j+1)) + \frac{5}{8}U(i-1,j) - \frac{1}{8}(U(i-1,j-1)+U(i-1,j+1)) - U(i,j)$

4: $\hat{d}_4 = \frac{5}{16}(U(i,j-1)+U(i,j+1)) + \frac{5}{8}U(i+1,j) - \frac{1}{8}(U(i+1,j-1)+U(i+1,j+1)) - U(i,j)$

5: find d_m such that $|d_m| = \min\{|\hat{d}_i|, i = 1, \cdots, 4\}$

Output: $\hat{U}(i,j) = U(i,j) + d_m$

Algorithm 4 Mean Curvature Filter $\mathcal{M}_c$

Input: $U(i,j)$

1: $\forall \vec{x} \in B_C$, $\mathcal{P}_m(U(\vec{x}))$

2: $\forall \vec{x} \in B_T$, $\mathcal{P}_m(U(\vec{x}))$

3: $\forall \vec{x} \in W_C$, $\mathcal{P}_m(U(\vec{x}))$

4: $\forall \vec{x} \in W_T$, $\mathcal{P}_m(U(\vec{x}))$

Output: $U(i,j)$

V. Total Variation Filter

We extend our filter concept to also construct a filter to rapidly minimize TV regularizers of the form:

$$\mathcal{E}^{\mathrm{TV}}_{\Phi_1}(U) = \|\nabla U(\vec{x})\|_p \,. \tag{14}$$

The TV filter $\mathcal{TV}$ is summarized in Algorithms 5 and 6. In the local projection operator $\mathcal{P}_{TV}$ we directly exploit the piecewise constant assumption. Again, the projection distances $\hat{d}_i$ are different from the directional curvatures d_i.

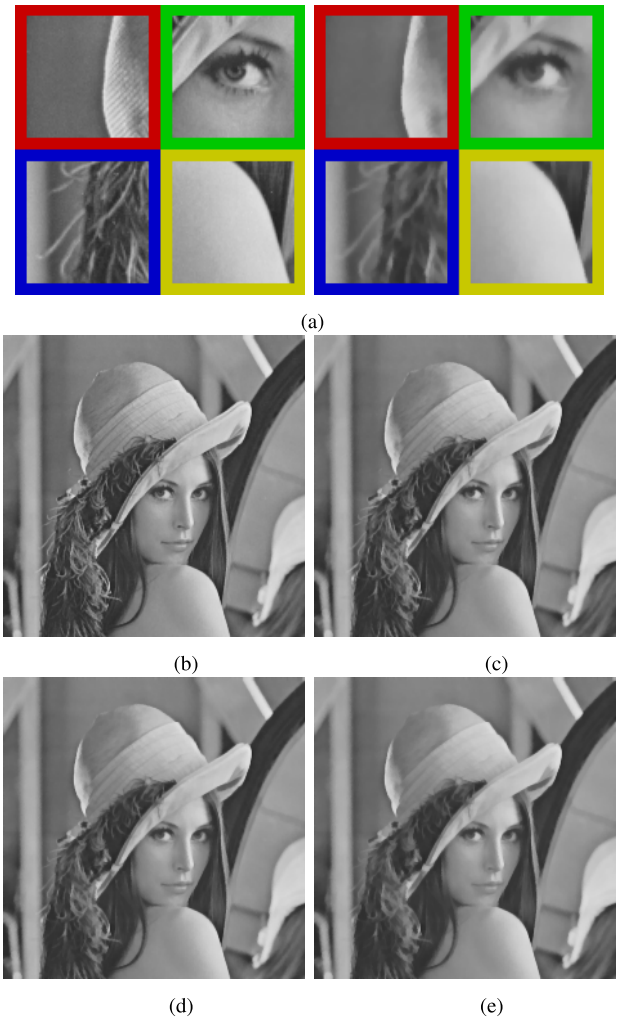


Fig. 11. Illustration of $\mathcal{M}_c$ on the *Lena* image for different iteration numbers. (a) original image regions and $\mathcal{M}_c^6$-filtered regions. (b) $\mathcal{M}_c^2$. (c) $\mathcal{M}_c^6$. (d) $\mathcal{M}_c^{10}$. (e) $\mathcal{M}_c^{20}$.

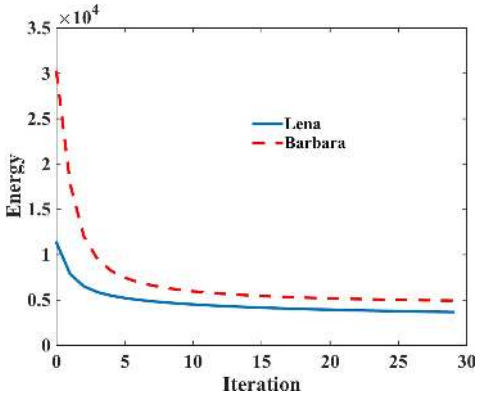


Fig. 12. $\mathcal{E}_{\Phi_1}^{\mathrm{MC}}$ evolution for the images *Lena* and *Barbara*.

A. Experiments

We again illustrate the filter on the *Lena* image. The result is shown in Fig. 14 and its energy evolution in Fig. 15. Because the filter is based on local operations in small windows, its spatial propagation requires large numbers of iterations, as shown in Fig. 14. This is in contrast to global solvers, like split-Bregman, which yield the approximation in few iterations.

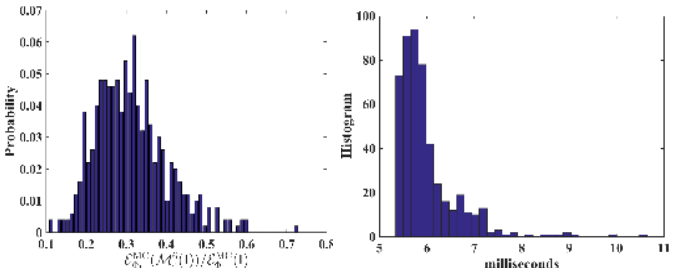


Fig. 13. Energy ratio statistics for $\mathcal{M}_c^6$ on BSDS500. The result shows that $\mathcal{M}_c^6$ can reduce the mean curvature energy $\mathcal{E}_{\Phi_1}^{\mathrm{MC}}$ by about 70% in six milliseconds. The filter is implemented in C++ using OpenCV and run on a 2 GHz Intel Core i7. (a) Distribution of energy ratios. (b) Runtime histogram.

Algorithm 5 Projection Operator $\mathcal{P}_{TV}$

Input: $U(i, j)$

$\hat{d}_1 = (U_{i-1,j-1} + U_{i-1,j} + U_{i,j-1} + U_{i+1,j-1} + U_{i+1,j})/5 - U_{i,j}$

$\hat{d}_2 = (U_{i-1,j} + U_{i-1,j+1} + U_{i,j+1} + U_{i+1,j} + U_{i+1,j+1})/5 - U_{i,j}$

$\hat{d}_3 = (U_{i-1,j-1} + U_{i-1,j} + U_{i-1,j+1} + U_{i,j-1} + U_{i,j+1})/5 - U_{i,j}$

$\hat{d}_4 = (U_{i+1,j-1} + U_{i+1,j} + U_{i+1,j+1} + U_{i,j-1} + U_{i,j+1})/5 - U_{i,j}$

$\hat{d}_5 = (U_{i-1,j-1} + U_{i-1,j} + U_{i-1,j+1} + U_{i,j-1} + U_{i+1,j-1})/5 - U_{i,j}$

$\hat{d}_6 = (U_{i-1,j-1} + U_{i-1,j} + U_{i-1,j+1} + U_{i,j+1} + U_{i+1,j+1})/5 - U_{i,j}$

$\hat{d}_7 = (U_{i+1,j-1} + U_{i+1,j} + U_{i+1,j+1} + U_{i-1,j-1} + U_{i,j-1})/5 - U_{i,j}$

$\hat{d}_8 = (U_{i+1,j-1} + U_{i+1,j} + U_{i+1,j+1} + U_{i-1,j+1} + U_{i,j+1})/5 - U_{i,j}$

find d_m, such that $|d_m| = \min\{|\hat{d}_i|, i = 1, \cdots, 8\}$

Output: $\hat{U}(i, j) = U(i, j) + d_m$

Algorithm 6 Total Variation Filter $\mathcal{TV}$

Input: $U(i, j)$

$\forall \vec{x} \in B_C$, $\mathcal{P}_{TV}(U(\vec{x}))$

$\forall \vec{x} \in B_T$, $\mathcal{P}_{TV}(U(\vec{x}))$

$\forall \vec{x} \in W_C$, $\mathcal{P}_{TV}(U(\vec{x}))$

$\forall \vec{x} \in W_T$, $\mathcal{P}_{TV}(U(\vec{x}))$

Output: $U(i, j)$

VI. Application to Variational Models With Generic Data-Fitting Terms

We show how the filters for the three regularization terms (GC, MC, TV) can be used to reduce the total energy of regularization-dominated variational models with generic data-fitting terms and spatially adaptive regularization function $\lambda(\vec{x})$. We hence consider models

$$\mathcal{E}(U) = \int_\Omega \Phi_0(U, I)\, \mathrm{d}\vec{x} + \int_\Omega \lambda(\vec{x}) \Phi_1(U)\, \mathrm{d}\vec{x} = \mathcal{E}_{\Phi_0} + \mathcal{E}_{\Phi_1}, \quad (15)$$

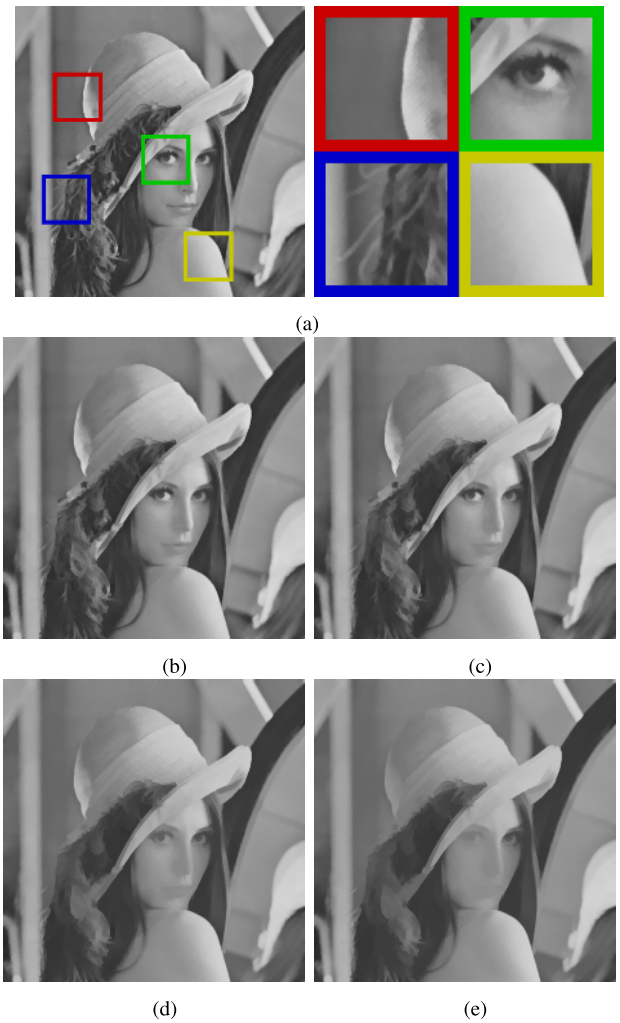


Fig. 14. $\mathcal{TV}$ filter on the *Lena* image for different iterations. (a) $\mathcal{TV}^{10}$ filter on the *Lena* image. (b) $\mathcal{TV}^{50}$. (c) $\mathcal{TV}^{100}$. (d) $\mathcal{TV}^{1000}$. (e) $\mathcal{TV}^{4000}$.

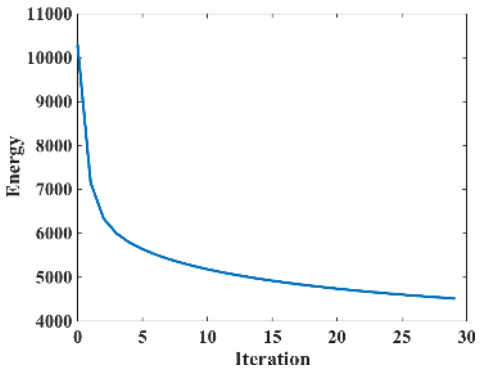


Fig. 15. $\mathcal{E}^{\text{TV}}_{\Phi_1}$ evolution for the $\mathcal{TV}$ filter on the *Lena* image.

where Φ_1 is either one of $|H|$, $|K|$, or $\mathcal{E}_{\Phi_1} = \|\nabla U\|_p$. This generic variational model covers many models that use geometric regularization.

Typically, solvers for such models exploit knowledge about the form of Φ_0. Based on our three filters, however, we can construct a local approximation to such models only requiring that Φ_0 can be evaluated (i.e., Φ_0 is a black box). Therefore, Φ_0 can be arbitrarily complex and need not even have an analytical form. This is impossible for solvers such as gradient decent, split-Bregman [28], [35], [36], Multi-Grid [37], and Primal/Dual methods [38]. Moreover, the regularization function $\lambda(\vec{x})$ can also be arbitrarily complex.

Algorithm 7 Spatially Adaptive Variational Filter $\mathcal{V}$

Input: $U(i,j)$, $\mathcal{P} = \mathcal{P}_g$, $\mathcal{P}_m$, or $\mathcal{P}_{TV}$

$\forall \vec{x} \in B_C$, $\hat{U}(\vec{x}) = \mathcal{P}(U(\vec{x}))$
  If $\lambda(\vec{x})(\Phi_1(\hat{U}(\vec{x})) - \Phi_1(U(\vec{x}))) \leq \Phi_0(U(\vec{x})) - \Phi_0(\hat{U}(\vec{x}))$
    $U(\vec{x}) = \hat{U}(\vec{x})$
$\forall \vec{x} \in B_T$, $\hat{U}(\vec{x}) = \mathcal{P}(U(\vec{x}))$
  If $\lambda(\vec{x})(\Phi_1(\hat{U}(\vec{x})) - \Phi_1(U(\vec{x}))) \leq \Phi_0(U(\vec{x})) - \Phi_0(\hat{U}(\vec{x}))$
    $U(\vec{x}) = \hat{U}(\vec{x})$
$\forall \vec{x} \in W_C$, $\hat{U}(\vec{x}) = \mathcal{P}(U(\vec{x}))$
  If $\lambda(\vec{x})(\Phi_1(\hat{U}(\vec{x})) - \Phi_1(U(\vec{x}))) \leq \Phi_0(U(\vec{x})) - \Phi_0(\hat{U}(\vec{x}))$
    $U(\vec{x}) = \hat{U}(\vec{x})$
$\forall \vec{x} \in W_T$, $\hat{U}(\vec{x}) = \mathcal{P}(U(\vec{x}))$
  If $\lambda(\vec{x})(\Phi_1(\hat{U}(\vec{x})) - \Phi_1(U(\vec{x}))) \leq \Phi_0(U(\vec{x})) - \Phi_0(\hat{U}(\vec{x}))$
    $U(\vec{x}) = \hat{U}(\vec{x})$

Output: $U(i,j)$

Iterating the corresponding filter $\mathcal{V}$ (any of $\mathcal{G}_c$, $\mathcal{M}_c$, $\mathcal{TV}$, depending on Φ_1) over the image, $\mathcal{E}_{\Phi_1}$ reduces as shown in previous sections, while $\mathcal{E}_{\Phi_0}$ increases when starting from I as the initial condition. The following additional condition ensures non-increasing total energy $\mathcal{E}$ and accounts for information from the data-fitting term:

$$\lambda(\vec{x})(\Phi_1(\hat{U}(\vec{x})) - \Phi_1(U(\vec{x}))) \leq \Phi_0(U(\vec{x})) - \Phi_0(\hat{U}(\vec{x})) \,\forall \vec{x} \implies \mathcal{E}(\mathcal{V}(U)) \leq \mathcal{E}(U), \tag{16}$$

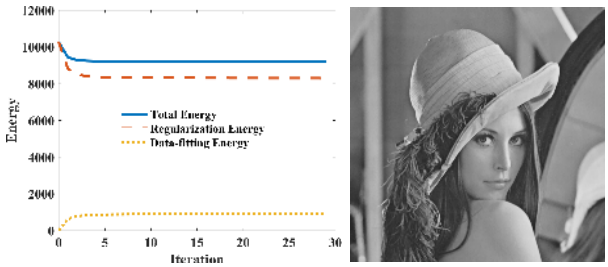
where $\hat{U} = \mathcal{P}(U)$ and $\mathcal{P}$ is the projection operator for $\mathcal{G}_c$, $\mathcal{M}_c$, or $\mathcal{TV}$, respectively. Φ_1 is evaluated using finite-difference approximations to the differential operators given in Section II. Since the above condition is evaluated point-wise at each pixel, it is sufficient to imply that $\mathcal{E}_{\Phi_1}$ reduces more than $\mathcal{E}_{\Phi_0}$ increases. However, the condition is not necessary, and potentially better moves could be obtained by other methods. We accept the projection at $\vec{x}$ if the condition is locally satisfied, and reject otherwise. This guarantees convergence of the total energy, albeit not necessarily to a mininum. It also allows us to choose different $\lambda(\vec{x})$ for different $\vec{x}$, which is why we call this filter "Spatially Adaptive Variational Filter" $\mathcal{V}$. The complete algorithm is summarized in Algorithm 7. The iteration loop stops if the above condition is false for all $\vec{x}$.

A. $\mathcal{V}$ on an Adaptive Data-Fitting Energy

We show the application of the filter to a variational model with a data-fitting energy that cannot be handled by previous solvers: a data-adaptive norm. While Bregman divergences in Φ_0 can be handled by split-Bregman iterations [9], data-adaptive norms have never been considered in global solvers. However, our filter can still provide approximate solutions.

Specifically, we consider the model:

$$\mathcal{E}(U) = \int_\Omega |U - I|^{2-|\nabla U|} \, \mathrm{d}\vec{x} + \lambda \|\nabla U\|_1 \tag{17}$$



(a) (b)

Fig. 16. The filter $\mathcal{V}$ on a variational model with content-adaptive noise model (Eq. 17 with $\lambda = 1$). (a) Energy evolution for filter $\mathcal{V}$. (b) $\mathcal{V}^{20}$-filtered *Lena* image.

with $\lambda = 1$. Since $U \in [0, 1]$, and we measure distances in units of pixels, the adaptive power $2 - |\nabla U|$ in the data-fitting term automatically balances between 0 and 2, depending on $|\nabla U(\vec{x})|$. This uses a sparse (hyper-Laplacian) noise model at corners, a Laplacian noise model at edges, and a Gaussian noise model in flat image regions.

Solving this model is difficult, because neither its Euler-Lagrange equation nor the gradient of the total energy can be easily obtained. Our filter $\mathcal{V}$, however, can straightforwardly be used to approximate this model without the need for an analytical form. The result is shown in Fig. 16.

B. $\mathcal{V}$ on the L2-TV-L1 Model

In order to show the influence of the data-fitting term, we test $\mathcal{V}$ on the standard L2-TV-L1 model for which the split-Bregman iteration method [28] is particularly popular.

The L2-TV-L1 model is defined as:

$$\mathcal{E}(U) = \|U - I\|_2^2 + \lambda \|\nabla U\|_1 \,. \tag{18}$$

We set $\lambda = 1$ for our tests. For the split-Bregman method, we set the stopping tolerance $\epsilon = 0.001$. The energy evolution for both solvers is shown in Fig. 17. Both methods achieve similar results, but the running times for split-Bregman and our filter are 0.24 seconds and 0.14 seconds, respectively (both implemented in C++ and run on a 2 GHz Intel i7 core). The approximate solution our filter finds in this case is comparable (in energy) to the optimal solution found by split-Bregman.

C. $\mathcal{V}$ on the L2-TV-L2 Model

We further illustrate the filter $\mathcal{V}$ on the standard Rudin-Osher-Fatemi (ROF) model [20] and compare it with a Primal/Dual method [38].

The ROF model is defined as:

$$\mathcal{E}(U) = \|U - I\|_2^2 + \lambda \|\nabla U\|_2 \,. \tag{19}$$

The result for $\lambda = 0.2$ is shown in Fig. 18. The filter reduces the total energy efficiently over the first few iterations, but then plateaus. The global Primal/Dual method converges slower, due to the global information exchange, but reaches a lower energy. Since this model is convex, $\mathcal{V}$ converges to a point that is not a minimum of $\mathcal{E}$, but still has lower total energy than the initial condition.

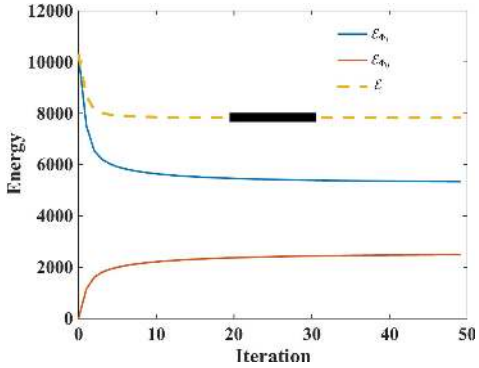
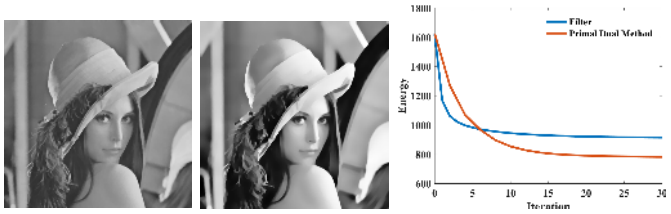
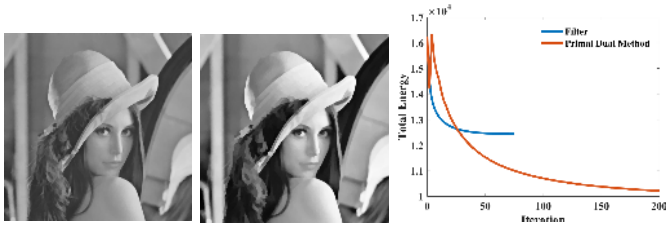


Fig. 17. Energy evolution for the filter $\mathcal{V}$ and final converged energy of the split-Bregman solver (horizontal black bar). The split-Bregman method [28] needs 18 iterations to converge to the energy indicated, while $\mathcal{V}$ needs 20 iterations.



(a) (b) (c)

Fig. 18. The filter $\mathcal{V}$ on the ROF model with $\lambda = 0.2$, compared with a global Primal/Dual solver [38]. (a) Result of $\mathcal{V}^{30}$. (b) Primal/Dual result. (c) Energy profiles for both.



(a) (b) (c)

Fig. 19. The filter $\mathcal{V}$ on the L1-TV-L2 model with $\lambda = 2$, compared with a global Primal/Dual solver [38]. (a) Result of $\mathcal{V}^{75}$. (b) Primal/Dual result. (c) Energy profiles for both.

D. $\mathcal{V}$ on the L1-TV-L2 Model

We also compare our filter with a Primal/Dual method [38] on the standard L1-TV-L2 model, defined as:

$$\mathcal{E}(U) = \|U - I\|_1 + \lambda \|\nabla U\|_2 \,. \tag{20}$$

The result for $\lambda = 2$ is shown in Fig. 19. Again, the filter converges faster, but plateaus at a sub-optimal solution, whereas the Primal/Dual method reaches a lower energy level.

E. $\mathcal{V}$ on an MC-Regularized Model

Beyond TV-regularized models, the filter $\mathcal{V}$ can also handle GC and MC regularization. We illustrate this using the MC-regularized model:

$$\mathcal{E}(U) = \int_\Omega |U - I|^q \, \mathrm{d}\vec{x} + \lambda \int_\Omega |H(U)| \, \mathrm{d}\vec{x} \tag{21}$$

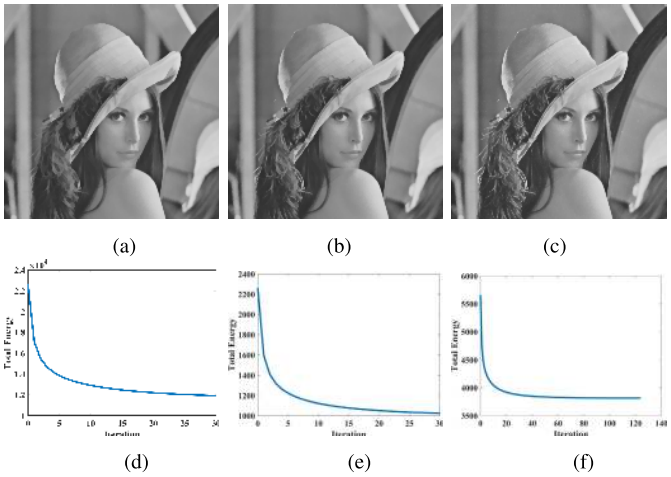


Fig. 20. The filter $\mathcal{V}$ for MC-regularized models with different data-fitting terms, and the corresponding evolution of the total energy $\mathcal{E}$ below. (a) L^1-MC, $\lambda = 2$. (b) L^2-MC, $\lambda = 0.2$. (c) $L^{1.5}$-MC, $\lambda = 0.5$. (d) Energy profile of (a). (e) Energy profile of (b). (f) Energy profile of (c).

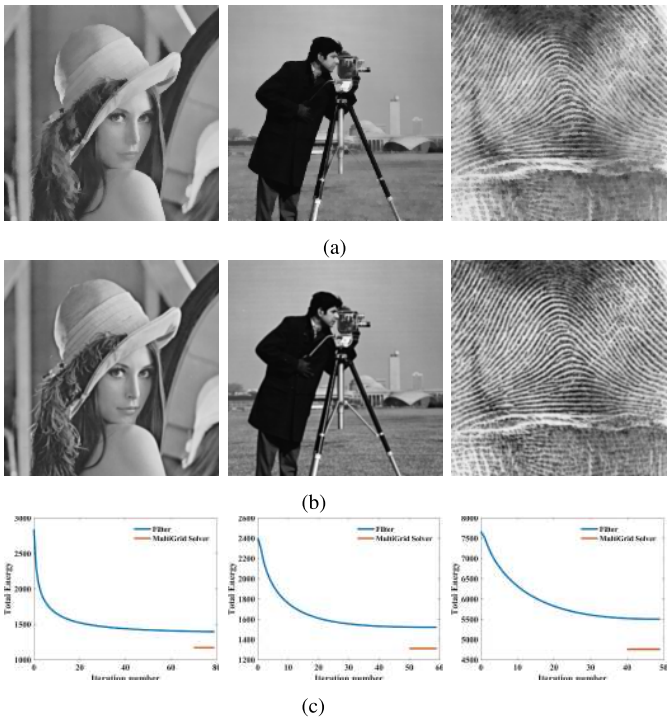


Fig. 21. Comparison of the filter $\mathcal{V}$ with a Multi-Grid solver [37] for the L^2-MC model. We set $\lambda = 1$ for both methods. From left to right, the running times for the Multi-Grid solver are 173, 193, and 217 seconds, respectively. The running times for our solver are 3.4, 4.1, and 5.5 seconds, respectively. Our filters were implemented in MATLAB, the Multi-Grid solver was obtained as binary MATLAB p-code. (a) Results from $\mathcal{V}$ on L^2-MC. (b) Results from the Multi-Grid solver [37] with $\gamma = 100$, $\beta = 1$, $\epsilon = 10^{-4}$. (c) The total energy profiles of the above. We only show the final energy for the Multi-Grid solver, because the code does not output intermediate results.

for $q = 1$, $q = 2$, and $q = 1.5$. The filter $\mathcal{V}$ can handle this model for any q. For $q = 1$, we set $\lambda = 2$, and the result is shown in Fig. 20(a/d). For $q = 2$, we set $\lambda = 0.2$, in order to make the two models comparable, and the result is shown in Fig. 20(b/e). For $q = 1.5$, the model becomes comparable for $\lambda = 0.5$, and the result is shown in Fig. 20(c/f).

For the L^2-MC model (i.e., $q = 2$), we compare our filter with a fast Multi-Grid solver [37]. This is the only value of q for which the Multi-Grid solver can handle the model. The comparison is shown in Fig. 21. The Multi-Grid solver reaches a lower final energy, but requires 40 to 50-fold more time to converge. The filter $\mathcal{V}$ also preserves edges better.

VII. Summary

We have presented a discrete filter-based approach to rapidly reduce the energy of regularization-dominated variational models with generic data-fitting terms. The approach is based on reducing the regularizer while ensuring non-increasing total energy. Convergence is guaranteed, but not necessarily to a minimum of the total energy. This is because the condition in Eq. 16 is sufficient, but not necessary, and also because no assumption is made about the monotonicity of the data-fitting term. When the filters are iterated over an image without considering Eq. 16, they converge to a (local) minimum of the regularization energy $\mathcal{E}_{\Phi_1}$ over the image domain Ω. When coupled with a data-fitting term, it is guaranteed that the final total energy is not higher than the initial one.

We have provided efficient energy-reduction filters for Gaussian curvature (GC), mean curvature (MC), and total variation (TV) energies. In all three cases, all exact minimizers in a 3×3 pixel neighborhood can be enumerated and selected based on evaluating the data-fitting energy. The filters run two to four order of magnitude faster than state-of-the-art global solvers. They have linear computational complexity in both runtime and memory. Our C++ implementation runs at 30 mega-pixel per second for GC-regularized denoising on a 2 GHz Intel Core i7, using only 17 memory words in addition to the image data. This allows handling images of large size, in particular when only loading a part of the image into memory, as in tile-based image processing [39]. Using graphics processing units (GPU), this can be further accelerated due to the streaming nature of the filters. The filters can also be parallelized on multi- and many-core hardware if additional performance is required.

We provided the mathematical background for these filters and analyzed them theoretically and experimentally. Then, we showed how these filters can be used to rapidly reduce the total energy in regularization-dominated variational models with regularizers of these types, but generic data-fitting terms. Comparisons with split-Bregman, Primal/Dual, and Multi-Grid solvers have confirmed that the resulting approximations are sub-optimal, but computationally efficient. In particular, the present filters yield rapid initial energy reduction, but then get stuck or are slow in moving towards the (global) minimum.

We have proven convergence of the proposed filters in the regularizer, but did not discuss their convergence rate. Since the filters are pixel-local, it is difficult to estimate the global convergence rate. Numerically, we find a convergence rate of 1.4 (1.37…1.43) across all BSDS500 images.

When rapid energy reduction is of interest, the present filters relax two limitations of diffusion and gradient-descent schemes: numerical stability and solution smoothness. In diffusion schemes, the pseudo-time step size has to fulfill the CFL (Courant-Friedrichs-Lewy) stability condition. The method is

unstable for large steps and does not advance for small time steps. While implicit solvers relax the CFL condition, they suffer from high condition numbers. Moreover, even the most efficient diffusion solvers require 123 additions, 45 multiplications, and 8 sign-change operations in each iteration [10]. This is more than the 25 operations needed by the present filters.

The second limitation concerns solution smoothness. Both diffusion methods and iteratively reweighted methods [21] require the regularizer to be explicitly computed in order to determine the geometric flow. Computing curvature over an image, however, requires the image to be at least twice differentiable, enforcing $U \in C^2$, which requires the final result to be smooth. Therefore, edges in the image will not be preserved. To relax this smoothness requirement, interpolation between neighboring pixels has been proposed [10]. This interpolation, however, generates artifacts (see Fig. 6), which are unavoidable in explicit minimization.

Our filter approach avoids the stability and smoothness problems. This is because we do not use a global flow, but instead locally approximate the image by piecewise developable, minimal, or constant surfaces (depending on the regularizer) that minimally increase the data-fitting energy. Due to the half-window regression used when doing so, we only require $U \in C^0$, but U not necessarily in C^1 or C^2. This renders our filters edge-preserving. While this is not true in the continuous domain, it holds in the discrete domain. This is equivalent to saying that the filters do not require the pixels in I to resolve (i.e., well-sample) all gradients in the continuous U, but can tolerate edges. In comparison with six other edge-preserving filters (WLS [40], AM [41], DT [42], GF [25], L0 [43], RTV [27]), the present GC filter consistently produced the visually most appealing results in the shortest runtime [22] (pp. 149–151).

Comparing the $\mathcal{TV}$, $\mathcal{M}_c$, and $\mathcal{G}_c$ filters, we note that they allow different geometric configurations. $\mathcal{TV}$ only allows constant intensity in a neighborhood, while $\mathcal{M}_c$ additionally allows linear planes. $\mathcal{G}_c$ allows the most geometric configurations, as long as they are developable, for example a cone structure. In this sense, GC regularization is a superset of MC regularization, which is a superset of TV regularization. This makes GC the least restrictive prior, allowing the most solutions. However, there are applications where stronger regularization may be required, notably at low PSNR. This explains our empirical observation that $\mathcal{TV}$ preserves large gradients, but removes small details, whereas $\mathcal{M}_c$ leads to a result that is smoother.

In the future, it could be interesting to extend the ideas presented here to higher-order models, like the fourth-order LLT model [44]. The results could then be compared with Augmented Lagrangian Methods (ALM) [45], which is not directly possible at this time. Moreover, extensions to color images and three-dimensional images are possible.

It is also worth noticing that the sequence of filtering operations presented here can be implemented as a layer in a Convolutional Neural Network (CNN), and network training can be done via standard error back-propagation. This is, e.g., similar to the recent effort of using mean-field Markov Random Field (MRF) operations in a CNN [46].

The filters presented here lead to fast algorithms for reducing the total energy of regularization-dominated variational models. They provide the flexibility of treating generic black-box data-fitting terms and spatially adaptive regularizers. While the results are sub-optimal, they are not restricted to particular image models. We therefore believe that these filters will benefit many practical tasks, for example in real-time or embedded image processing.

REFERENCES

[1] J. Xu and S. Osher, "Iterative regularization and nonlinear inverse scale space applied to wavelet-based denoising," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 534–544, Feb. 2007.

[2] M. Ibrahim, K. Chen, and C. Brito-Loeza. (2015). "A novel variational model for image registration using Gaussian curvature." [Online]. Available: https://arxiv.org/abs/1504.07643

[3] Y. Gong and I. F. Sbalzarini, "Image enhancement by gradient distribution specification," in *Proc. 12th Asian Conf. Comput. Vis. (ACCV)*, Singapore, Nov. 2014, pp. 47–62.

[4] Y. Gong and I. F. Sbalzarini, "A natural-scene gradient distribution prior and its application in light-microscopy image processing," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 1, pp. 99–114, Feb. 2016.

[5] Y. Xie, W. Zhang, D. Tao, W. Hu, Y. Qu, and H. Wang, "Removing turbulence effect via hybrid total variation and deformation-guided kernel regression," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4943–4958, Oct. 2016.

[6] P. Purkait and B. Chanda, "Super resolution image reconstruction through Bregman iteration using morphologic regularization," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4029–4039, Sep. 2012.

[7] Y. Chang, L. Yan, H. Fang, and C. Luo, "Anisotropic spectral-spatial total variation model for multispectral remote sensing image destriping," *IEEE Trans. Image Process.*, vol. 24, no. 6, pp. 1852–1866, Jun. 2015.

[8] J. Cardinale, G. Paul, and I. F. Sbalzarini, "Discrete region competition for unknown numbers of connected regions," *IEEE Trans. Image Process.*, vol. 21, no. 8, pp. 3531–3545, Aug. 2012.

[9] G. Paul, J. Cardinale, and I. F. Sbalzarini, "Coupling image restoration and segmentation: A generalized linear model/Bregman perspective," *Int. J. Comput. Vis.*, vol. 104, no. 1, pp. 69–93, 2013.

[10] S.-H. Lee and J. K. Seo, "Noise removal with Gauss curvature-driven diffusion," *IEEE Trans. Image Process.*, vol. 14, no. 7, pp. 904–909, Jul. 2005.

[11] H. Zhao and G. Xu, "Triangular surface mesh fairing via Gaussian curvature flow," *J. Comput. Appl. Math.*, vol. 195, no. 1, pp. 300–311, 2006.

[12] H. Zhu, H. Shu, J. Zhou, X. Bao, and L. Luo, "Bayesian algorithms for PET image reconstruction with mean curvature and Gauss curvature diffusion regularizations," *Comput. Biol. Med.*, vol. 37, no. 6, pp. 793–804, 2007.

[13] N. C. Overgaard and J. E. Solem, "The variational origin of motion by Gaussian curvature," in *Scale Space and Variational Methods in Computer Vision* (Lecture Notes in Computer Science), vol. 4485. Berlin, Germany: Springer, 2007.

[14] B. Lu, H. Wang, and Z. Lin, "High order Gaussian curvature flow for image smoothing," in *Proc. Int. Conf. Multimedia Technol. (ICMT)*, Jul. 2011, pp. 5888–5891.

[15] A. Shekhovtsov, P. Kohli, and C. Rother, "Curvature prior for MRF-based segmentation and shape inpainting," in *Pattern Recognition: Joint 34th DAGM and 36th OAGM Symposium*. Berlin, Germany: Springer 2012, pp. 41–51.

[16] A. I. El-Fallah and G. E. Ford, "Mean curvature evolution and surface area scaling in image filtering," *IEEE Trans. Image Process.*, vol. 6, no. 5, pp. 750–753, May 1997.

[17] X. Liu, Z. Ying, and S. Qiu, "A fourth-order partial differential equations method of noise removal," in *Proc. 4th Int. Congr. Image Signal Process. (CISP)*, vol. 2. Oct. 2011, pp. 641–645.

[18] D. Tschumperlé, "Fast anisotropic smoothing of multi-valued images using curvature-preserving PDE's," *Int. J. Comput. Vis.*, vol. 68, no. 1, pp. 65–82, 2006.

[19] D. Tschumperlé and R. Deriche, "Vector-valued image regularization with PDEs: A common framework for different applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 4, pp. 506–517, Apr. 2005.
[20] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D, Nonlinear Phenomena*, vol. 60, nos. 1–4, pp. 259–268, 1992.
[21] Y. Gong and I. F. Sbalzarini, "Local weighted Gaussian curvature for image processing," in *Proc. 20th IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2013, pp. 534–538.
[22] Y. Gong, "Spectrally regularized surfaces," Ph.D. dissertation, MOSAIC Group, ETH Zürich, Zürich, Switzerland, Mar. 2015.
[23] H. Pottmann and J. Wallner, *Computational Line Geometry*. Berlin, Germany: Springer, 2001.
[24] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. ICCV*, 1998, pp. 839–846.
[25] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proc. ECCV*, 2010, pp. 1–14.
[26] J. Lu, K. Shi, D. Min, L. Lin, and M. N. Do, "Cross-based local multipoint filtering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 430–437.
[27] L. Xu, Q. Yan, Y. Xia, and J. Jia, "Structure extraction from texture via relative total variation," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 139:1–139:10, Nov. 2012.
[28] T. Goldstein and S. Osher, "The split Bregman method for L1-regularized problems," *SIAM J. Imag. Sci.*, vol. 2, no. 2, pp. 323–343, 2009.
[29] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.
[30] W. Zhu and T. Chan, "Image denoising using mean curvature of image surface," *SIAM J. Imag. Sci.*, vol. 5, no. 1, pp. 1–32, 2012.
[31] W. Zhu, X.-C. Tai, and T. Chan, "Augmented Lagrangian method for a mean curvature based image denoising model," *Inverse Problems Imag.*, vol. 7, no. 4, pp. 1409–1432, Nov. 2013.
[32] F. Yang, K. Chen, B. Yu, and D. Fang, "A relaxed fixed point method for a mean curvature-based denoising model," *Optim. Method Softw.*, vol. 29, no. 2, pp. 274–285, Mar. 2014.
[33] M. Bertalmio and S. Levine, "Denoising an image by denoising its curvature image," *SIAM J. Imag. Sci.*, vol. 7, no. 1, pp. 187–211, 2014.
[34] Y. Gong, "Bernstein filter: A new solver for mean curvature regularized models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 1701–1705.
[35] J.-F. Cai, S. Osher, and Z. Shen, "Linearized Bregman iterations for compressed sensing," *Math. Comput.*, vol. 78, no. 78, pp. 1515–1536, 2009.
[36] B. He, Y. You, and X. Yuan, "On the convergence of primal-dual hybrid gradient algorithm," *SIAM J. Imag. Sci.*, vol. 7, no. 4, pp. 2526–2537, 2014.
[37] C. Brito-Loeza and K. Chen, "Multigrid algorithm for high order denoising," *SIAM J. Imag. Sci.*, vol. 3, no. 3, pp. 363–389, 2010.
[38] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *J. Math. Imag. Vis.*, vol. 40, no. 1, pp. 120–145, 2011.
[39] M. Patel, "A memory-constrained image-processing architecture," *Dr. Dobbs J.*, vol. 22, no. 7, p. 24, Jul. 1997.
[40] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 67:1–67:10, Aug. 2008.
[41] E. S. L. Gastal and M. M. Oliveira, "Adaptive manifolds for real-time high-dimensional filtering," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 33:1–33:13, 2012.
[42] E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM Trans. Graph.*, vol. 30, no. 4, p. 69, 2011.
[43] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L_0 gradient minimization," *ACM Trans. Graph.*, vol. 30, no. 6, 2011, Art. no. 174.
[44] M. Lysaker, A. Lundervold, and X.-C. Tai, "Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images in space and time," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1579–1590, Dec. 2003.
[45] C. Wu and X.-C. Tai, "Augmented lagrangian method, dual methods, and split Bregman iteration for ROF, vectorial TV, and high order models," *SIAM J. Imag. Sci.*, vol. 3, no. 3, pp. 300–339, 2010.
[46] S. Zheng *et al.*, "Conditional random fields as recurrent neural networks," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 1529–1537.

Yuanhao Gong received the B.Sc. degree in mathematics from Tsinghua University, Beijing, China, in 2007, the M.Sc. degree in computer science from Xiamen University, Fujian, China, in 2010, and the Ph.D. degree in computer science from ETH Zurich, Switzerland, in 2015. His M.Sc. project focused on microscopy image processing, reverse multi-scale spaces, and local symmetry detection.

In 2015, he has been a Post-Doctoral Researcher with the SINAPSE Institute, National University of Singapore. He is currently with the Medical Image Analysis Group, University of Bern, Switzerland.

Dr. Gong was a recipient of the Gold Medal in the National Mathematics Olympics in 1999. He won the National Support Scholarship in 2006. In 2012, he received the Best Student Paper Award at the IEEE International Symposium on Biomedical Imaging.

Ivo F. Sbalzarini received the Diplom (equiv. M.Sc.) degree in mechanical engineering, with majors in control theory and computational science and the Ph.D. degree in computer science from ETH Zurich, Switzerland, in 2002 and 2006, respectively.

In 2006-2012, he was an Assistant Professor of Computational Science with ETH Zurich. Since 2012, he is a Senior Research Group Leader with the Center for Systems Biology Dresden, Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany. Since 2014, he additionally holds the Chair of Scientific Computing for Systems Biology with the Faculty of Computer Science, TU Dresden. He is the founder of the MOSAIC Group. His research interests include biological image analysis, adaptive discretization schemes for PDEs, randomized algorithms, and parallel computing.

Dr. Sbalzarini is a Life-Long Honorary Member of the Technical Society of Zurich. He was awarded the Willi Studer Prize for his Diplom (equiv. M.Sc.) degree. His Ph.D. dissertation received the 2006 Chorafas Award from the Weizmann Institute of Science. He is an Associate Editor of *BMC Bioinformatics*, and serves on various conference, program, and reviewer boards.