# TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

# Real-Time Minimization of the Piecewise Smooth Mumford-Shah Functional

Evgeny Strekalovskiy and Daniel Cremers

TUM-I146

# Real-Time Minimization of the Piecewise Smooth Mumford-Shah Functional

Evgeny Strekalovskiy and Daniel Cremers

TU Munich, Germany

**Abstract.** We propose an algorithm for efficiently minimizing the piecewise smooth Mumford-Shah functional. The algorithm is based on an extension of a recent primal-dual algorithm from convex to non-convex optimization problems. The key idea is to rewrite the proximal operator in the primal-dual algorithm using Moreau's identity. The resulting algorithm computes piecewise smooth approximations of color images at 15-20 frames per second at VGA resolution using GPU acceleration. Compared to convex relaxation approaches [1], it is orders of magnitude faster and does not require a discretization of color values. In contrast to the popular Ambrosio-Tortorelli approach [2], it naturally combines piecewise smooth and piecewise constant approximations, it does not require an epsilon-approximation and it is not based on an alternation scheme. The achieved energies are in practice at most 5% off the optimal value for one-dimensional problems. Numerous experiments demonstrate that the proposed algorithm is well-suited to perform discontinuity-preserving smoothing and real-time video cartooning.

## 1 Introduction

With around 3900 citations to date, the Mumford-Shah functional [3] is among the most influential publications in the field of image analysis. This and related publications by Blake and Zisserman [4] and others have sparked enormous research activity on discontinuity-preserving smoothing, piecewise-smooth approximations and minimal partition problems [5]. Yet, the computation of the *piecewise-smooth* approximation has rarely made it into practical image and video analysis methods because minimization of this non-convex functional is difficult and existing algorithmic solutions are far from real-time capability. The contribution of this paper is to propose what we believe to be the first real-time capable algorithm for computing piecewise smooth approximations of color images based on the Mumford-Shah functional.

### 1.1 The Mumford-Shah Problem

The Mumford-Shah functional [3] provides a prototypical form of all regularizers which aim at combining a smoothing of homogeneous regions with the

Input image Piecewise smooth approximation

**Fig. 1.** We propose a fast algorithm to approximately solve the piecewise smooth Mumford-Shah model. With 20 frames per second and above, it allows real-time image processing with applications such as denoising and cartooning.

enhancement of edges. Given a bounded open set $\Omega \subset \mathbb{R}^d$, $d \geq 1$, the vectorial Mumford-Shah problem is given by

$$\min_{u,K} \left\{ \int_{\Omega} |u - f|^2 \, dx + \alpha \int_{\Omega \setminus K} |\nabla u|^2 \, dx + \lambda \, |K| \right\}, \tag{1}$$

where $f : \Omega \to \mathbb{R}^k$ is a vector-valued input image with $k \geq 1$ channels. This model approximates $f$ by a function $u : \Omega \to \mathbb{R}^k$ which is smooth everywhere in $\Omega$ except for a possible $(d-1)$-dimensional jump set $K$, at which $u$ is discontinuous. The weight $\lambda > 0$ controls the length of the jump set $K$ (less jumps for larger $\lambda$) and $\alpha > 0$ penalizes the smoothness of $u$ outside of $K$. The limiting case $\alpha \to \infty$ imposes zero gradient outside $K$ and is known as the *piecewise constant* Mumford-Shah model or the "cartoon" limit [3]. The norm of the gradient $|\nabla u|$ in (1) is the Euclidean norm $|\nabla u|^2 = \sum_i |\nabla u_i|^2$, and the norm in the term $|u - f|$ is also Euclidean.

Since the jump set $K$ is defined as the union of the jump sets of the individual channels $u_i : \Omega \to \mathbb{R}$, we have a coupling among the channels assuring that jumps in different channels preferably coincide – see also [1].

### 1.2 Related Work

The Mumford-Shah problem has been intensively studied in the applied math community [5]. In practice its applicability is substantially limited because of its non-convexity. While it is often replaced by the convex total variation, this is a poor substitute because of its tendency to reduce the contrast at edges and oversmooth flat regions (staircasing). As a consequence, researchers have developed different optimization strategies to tackle the non-convex Mumford-Shah problem.

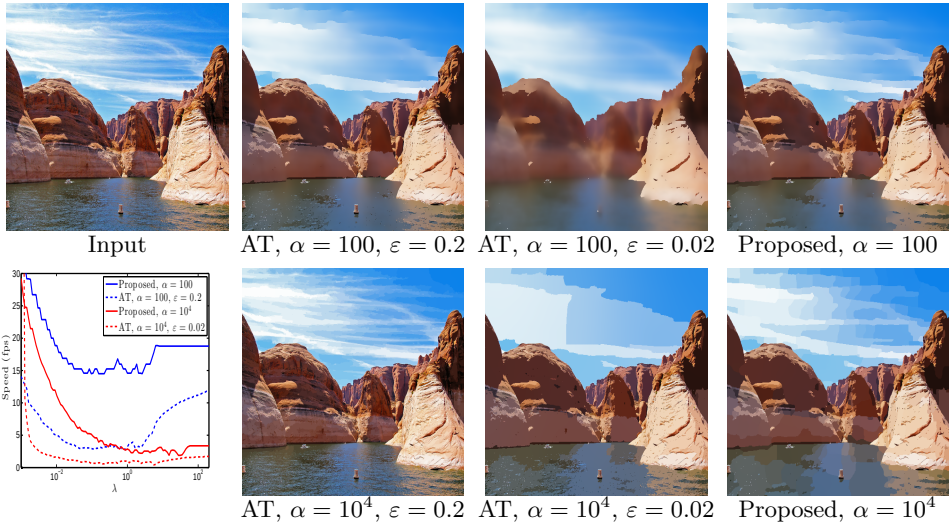


**Fig. 2. Comparison with the Ambrosio-Tortorelli model.** The images show piecewise smooth approximation results for $\lambda = 0.1$, $\alpha = 100$ **(top row)** and $\lambda = 0.1$, $\alpha = 10000$ **(bottom row)**. While the AT results vary with the parameter $\varepsilon$ (and it is not clear how to choose it appropriately), our method has no additional parameters, is stable for every $\lambda$ and $\alpha$, and about 3–5 times faster.

**Alternating Minimization Schemes.** One kind of methods consists of non-convex approximations of the original Mumford-Shah functional, where one alternatingly minimizes for $u$ and for $K$ [2, 6, 7].

*The Ambrosio-Tortorelli Approach.* The non-convex phase-field model of Ambrosio and Tortorelli [2], for example, is given by:

$$\min_{u,s} \int_\Omega |u - f|^2 \, dx + \alpha \int_\Omega (1-s)^2 |\nabla u|^2 \, dx + \lambda \int_\Omega \left( \varepsilon |\nabla s|^2 + \frac{1}{4\varepsilon} s^2 \right) dx \quad (2)$$

with a small parameter $\varepsilon > 0$. The key idea is to introduce the additional variable $s : \Omega \to \mathbb{R}$ as an edge set indicator, in the sense that points $x \in \Omega$ are part of the edge set $K$ if $s(x) \approx 1$ and part of the smooth region if $s(x) \approx 0$.

It was shown in [2] that this approximation $\Gamma$-converges to the Mumford-Shah functional for $\varepsilon \to 0$, i.e. minimizers of (2) approach minimizers of (1). One finds $u$ and $s$ by alternating minimization, computing $s$ for fixed $u$ and vice versa. Each of these subproblems is elliptic and can be solved quickly, e.g. by the linearly converging primal-dual method [8]. Extensions of this approximation to color images have been proposed in [9].

One disadvantage of this model, beside its non-convexity, is its dependancy on an additional parameter $\varepsilon$. To obtain a good approximation of minimizers of (1), $\varepsilon$ will depend on $\alpha$, $\lambda$ and $f$. Generally, $\varepsilon$ must be chosen small for increasing $\alpha$, and for large $\alpha$ the dependancy becomes sensitive and a good choice is unclear. This makes the approach infeasible for the piecewise constant case $\alpha = \infty$.

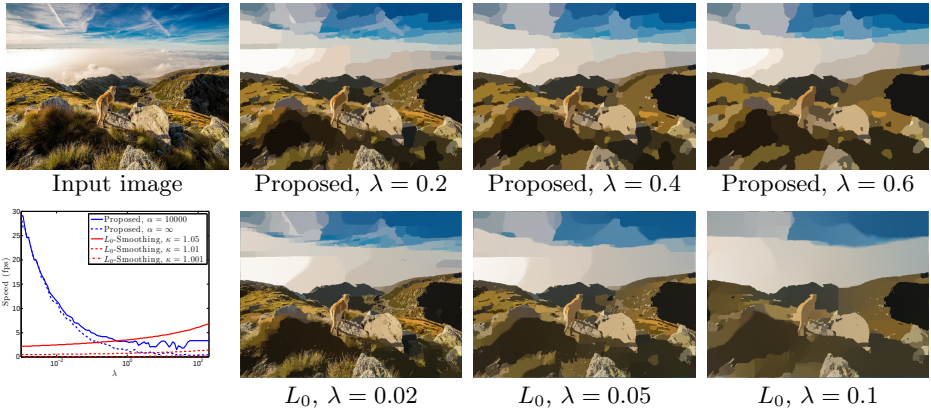


**Fig. 3. Real-time unsupervised segmentation.** The proposed method (**top row**) directly includes the piecewise *constant* case $\alpha = \infty$, which allows to segment an image with an automatic selection of suitable color models in real-time. This is in contrast to $L_0$-Smoothing (**bottom row**, $\kappa = 1.05$), which leads to piecewise smooth solutions. Avoiding this by choosing small $\kappa > 1$ leads to significant run times.

*The $L_0$-Smoothing Approach of Xu et al.* For the piecewise constant case, Xu et al. [10] recently proposed a fast approximating method. Assuming the image domain has been discretized into a finite rectangular grid, again denoted by $\Omega$, the piecewise constant Mumford-Shah limit corresponds to $L_0$ penalization of the gradient:

$$\min_u \sum_{x \in \Omega} |u(x) - f(x)|^2 + R_{MS_0}(\nabla u(x)) \tag{3}$$

where

$$R_{MS_0}(g) = \begin{cases} \lambda & \text{if } g \neq 0 \\ 0 & \text{else} \end{cases} \tag{4}$$

and the gradient $\nabla u = (\nabla u_i)_{1 \leq i \leq k}$ is discretized e.g. using forward differences. Intuitively, the regularizer $R_{MS_0}$ summed up over all pixels counts how many times $u$ changes its value. This way, it prefers regions of constancy instead of smooth variations. Xu et al. propose a quadratic decoupling strategy to solve (3), introducing new variables $g$ which approximate the gradient:

$$\min_{u,g} \sum_{x \in \Omega} |u(x) - f(x)|^2 + \beta |\nabla u(x) - g(x)|^2 + R_{MS_0}(g(x)) \tag{5}$$

with a parameter $\beta > 0$ and the Euclidean norm for the coupling. This approximation is again solved via alternating minimization. After having computed the next $u$ and $g$, the parameter $\beta$ is increased to $\kappa\beta$ with a $\kappa > 1$ until a final $\beta_{\max}$ is reached. Starting value for $\beta$ is chosen automatically as $\beta_0 = 2\lambda$. Multiplier $\kappa$ is set either to 2 (fast but smooth result) or 1.05 (slow and more piecewise constant).

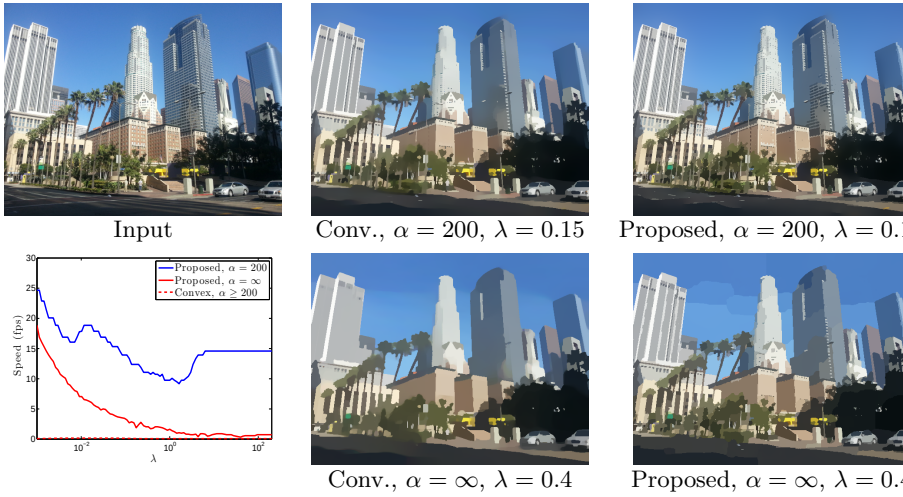


**Fig. 4. Comparison with the convexification approach [1] for the MS model (1).** The images show denoising results for different parameters $\alpha$ and $\lambda$. While both algorithms provide discontinuity-preserving smoothing, in the piecewise constant limit (bottom row), the solutions of [1] appear too smooth or blurry. Moreover, since [1] adds an additional dimension to the problem because of color quantization, it requires much more memory ($> 2$ GB) and is about 50–500 times slower than our approach.

Because of the empirical nature of the coupling, it is not clear how the computed solutions $u$ mathematically relate to the original model (3), or even to (1). In fact, the computed solutions are actually not piecewise constant, but vary smoothly over large areas.

**Convex Relaxation Methods.** In the recent past, several authors have overcome the issue of non-convexity by suggesting convex relaxations for respective functionals [11, 12]. Convex relaxations for the piecewise constant Mumford-Shah functional were proposed in [13–15]. Convex relaxations for the piecewise smooth Mumford-Shah model were proposed for the scalar [16] and for the vectorial case [1]. The key idea is to rewrite the multi-label problem as a binary labeling problem in a higher-dimensional space. Relaxation of the binary constraint leads to a convex problem which can be minimized optimally. Subsequent binarization provides an approximate solution of the original problem.

Some of these approaches were clearly inspired by the Markov random field (MRF) community, where the discrete variant of the Mumford-Shah regularizer is typically referred to as a truncated quadratic penalizer [17–19].

Unfortunately, these methods to compute approximate minimizers are currently far from real-time capability because the added label space dimension drastically increases memory and run time. For the Mumford-Shah model the run time even grows quadratically in the number of considered color values. Thus,

the problem of computing good approximate minimizers of the Mumford-Shah energy in real-time remains an important challenge.

### 1.3 Contribution

In this paper, we propose a novel algorithm to efficiently compute approximate minimizers of the full Mumford-Shah functional (1). It combines several important advantages:

- The Mumford-Shah energy is minimized directly, instead of alternatingly. Therefore there is a clear correspondence between the computed solutions and the original functional.
- In contrast to other methods like the Ambrosio-Tortorelli approximation or the $L_0$-Smoothing, our method naturally combines the piecewise smooth and the piecewise constant approximation. In the latter case, the solution is guaranteed to be piecewise constant, in contrast to the $L_0$-smoothing.
- The method runs in real-time. For VGA $640\times480$ images, the achieved frame rates are 15–20 Hz for piecewise smooth approximations, and still about 5–10 Hz for almost piecewise constant ones ($\alpha$ large).
- The algorithm is easy to implementIt is based on a simple state-of-the-art primal-dual algorithm. In order to apply it to the non-convex MS regularizer, we reformulate one step of this algorithm in a slight but crucial way. All computations remain simple and elementary.

## 2 Proposed Finite-Difference Discretization

Similar to [10] we work with an already discretized image domain $\Omega$. We propose to minimize the energy

$$\min_u E_{MS}(u) = \sum_{x\in\Omega} |u(x) - f(x)|^2 + R_{MS}(\nabla u(x)) \tag{6}$$

with

$$R_{MS}(g) := \min\left(\alpha|g|^2, \lambda\right). \tag{7}$$

Just as in the term $|\nabla u|$ in (1), the gradient norm in (7) is the Euclidean one, regarding the matrix $g = \nabla u = (\nabla u_i)_{1\le i\le k} \in (\mathbb{R}^d)^k$ as an element of $\mathbb{R}^{dk}$. We use the gradient discretization by forward differences as they give visually good results at minimal implementation costs. Another possibility would be to use a "staggered grid" [15].

The idea of (6) is to model the edge set explicitly in terms of the function $u$ itself. Namely, the edge set $K$ consists here of all points where the minimum value is $\lambda$ in (7), i.e. where the gradient is large enough:

$$K_{MS} := \left\{x \in \Omega \;\middle|\; |\nabla u(x)| \ge \sqrt{\lambda/\alpha}\right\}. \tag{8}$$

Indeed, for $x \in K_{MS}$ we have $R_{MS}(\nabla u(x)) = \lambda$, while for $x \notin K_{MS}$ we have $R_{MS}(\nabla u(x)) = \alpha|\nabla u(x)|^2$.

In the cartoon limit $\alpha \to \infty$ expression (7) becomes (4). Applying this regularizer in (6) yields *piecewise constant* approximations of the input image. This model is the same as (3) which is considered in [10]. Note that in contrast to convex relaxation methods [16, 1] we do not need to discretize the range of $u : \Omega \to \mathbb{R}^k$ into a finite set in order to compute solutions, and therefore fully avoid the corresponding dramatic increase in memory consumption and computation time. The model (6) can be regarded as a natural discretization of the Mumford-Shah energy (1) for a discrete image domain, as will be explained next.

*Connection to the Mumford-Shah Energy* (1). For $2D$ scalar images, i.e. $k = 1$ and $d = 2$, in [20] Chambolle considered a variant $E_{l^1}$ of the Mumford-Shah energy (1) where the usual Euclidean length $|K|$ of the edge set is replaced by the $l^1$-norm length $L_{l^1}(K)$. Intuitively, if $K$ were approximated by small line pieces, everyone of which being either horizontal or vertical, $L_{l^1}(K)$ would be the usual length of this approximation. It was shown that, if the image domain is discretized into a finite rectangular grid of pixel width $h$, the discrete energies

$$E^h(u) = \sum_{x \in \Omega} h^2 \, |u(x) - f^h(x)|^2 + R^h_{MS}\big(\nabla u(x)\big) \tag{9}$$

$\Gamma$-converge to $E_{l^1}$ for $h \to 0$, with

$$R^h_{MS}(g) := \min\big(\alpha|g_1|^2, \tfrac{1}{h}\lambda\big) + \min\big(\alpha|g_2|^2, \tfrac{1}{h}\lambda\big) \tag{10}$$

and $f^h(x) := \frac{1}{h^2} \int_{\text{Pixel } x} f^{\text{continuous}}(y)\, dy$. The gradient $\nabla$ is discretized by forward differences. For instance, for the approximation of $E_{l^1}$ by (9) we can choose $h = 1$, assuming the continuous domain has the size of the pixel grid.

Our proposed energy (6) is motivated by (9). Indeed, observe that penalizing $g_1 = \partial_x u$ and $g_2 = \partial_y u$ separately in (10) (no coupling) results in the $l^1$-norm length $L_{l^1}(K)$ for the edge set length. A natural conjecture is that, changing the coupling to the $l^2$-norm as in the proposed model (7), one would obtain the usual $l^2$-norm length $|K|$. Experiments suggest that this is indeed the case, since the obtained solutions do not show visible signs of grid dependance.

In addition to considering the $l^2$-coupling of the gradient in (7), we simultaneously extend the approach to the vectorial case $k \geq 1$, again by considering the $l^2$-norm of all possible derivatives of $u$ in (7).

## 3 Minimization Algorithm

### 3.1 Algorithm for Convex Regularizers $R$

To give a motivation for our algorithm, we first consider the energy minimization problem (6) for *convex* regularizers $R$ in place of $R_{MS}$. First we need a few definitions. For a function $R : \mathbb{R}^{d \times k} \to \mathbb{R}$ the Legendre-Fenchel *convex conjugate* [21] is defined as

$$R^*(p) := \sup_{g \in \mathbb{R}^{d \times k}} \langle p, g \rangle - R(g), \tag{11}$$

where $\langle \cdot, \cdot \rangle$ is the standard scalar product on $\mathbb{R}^{d \times k}$. A well-known fact about the convex conjugate is that for convex and lower-semicontinuous $R$ it holds $R = (R^*)^*$, i.e.

$$R(g) = \sup_{p \in \mathbb{R}^{d \times k}} \langle p, g \rangle - R^*(p). \tag{12}$$

For general $R$, expression (12) gives the *convex envelope* of $R$, which is the largest convex function pointwise below or equal to $R$. The *proximal operator* [8] of $R$ is defined as

$$\operatorname{prox}_{\tau,\, R}(\tilde{g}) := \arg\min_{g \in \mathbb{R}^{d \times k}} \frac{|g - \tilde{g}|^2}{2\tau} + R(g) \tag{13}$$

for parameters $\tau > 0$ and arguments $\tilde{g} \in \mathbb{R}^{d \times k}$.

*Primal-Dual Algorithm.* Consider now the energy (6) with a *convex* (and lower-semicontinuous) regularizer $R$ instead of $R_{MS}$. The first step is to use (12) as a means of variable decoupling. We get

$$E(u) = \sup_{p: \Omega \to \mathbb{R}^{d \times k}} \sum_{x \in \Omega} |u(x) - f(x)|^2 + \big\langle p(x), \nabla u(x) \big\rangle - R^*\big(p(x)\big). \tag{14}$$

Taking the minimum of (14) over $u$, we obtain a classical saddle-point problem. The state-of-the-art primal-dual algorithm [8] is developed especially for this kind of problems. Furthermore, we can use the accelerated Algorithm 2 of [8] since the data term $D(u) := \sum_{x \in \Omega} |u(x) - f(x)|^2$ in (14) is uniformly convex with constant $\gamma = 2$: for any $u$ and $u'$, $D(u) \geq D(u') + \langle 2f, u - u' \rangle + \frac{\gamma}{2}\|u - u'\|^2$. The update equations are as follows:

$$p^{n+1} = \operatorname{prox}_{\sigma_n,\, R^*}\left(p^n + \sigma_n \nabla \overline{u}^n\right), \tag{15}$$

$$u^{n+1} = \operatorname{prox}_{\tau_n,\, D}\left(u^n + \tau_n \operatorname{div} p^{n+1}\right), \tag{16}$$

$$\theta_n = \tfrac{1}{\sqrt{1 + 4\tau_n}}, \quad \tau_{n+1} = \theta_n \tau_n, \quad \sigma_{n+1} = \sigma_n / \theta_n, \tag{17}$$

$$\overline{u}^{n+1} = u^{n+1} + \theta_n (u^{n+1} - u^n). \tag{18}$$

The divergence $\operatorname{div} := -\nabla^T$ is defined as the negative adjoint of the gradient. The starting values $u^0$ and $p^0$ are arbitrary, with $\overline{u}^0 = u^0$ and $\tau_0 \sigma_0 \|\nabla\|^2 < 1$. Since $\|\nabla\| < \sqrt{4d}$ [8], we can set $\tau_0 = \frac{1}{2d}$, $\sigma_0 = \frac{1}{2}$. As proved in [8], the iterates $(u^n, p^n)$ converge to a solution of the saddle-point problem (14) with energy rate $\mathcal{O}(1/n^2)$.

The proximal operator for $u$ in (16) can be easily computed explicitly:

$$\operatorname{prox}_{\tau,\, D}(\tilde{u}) = \frac{\tilde{u} + 2\tau f}{1 + 2\tau}. \tag{19}$$

*Reformulation of* (15) *by Moreau's Identity.* Note that the only place where the regularizer enters the algorithm is step (15), and this dependancy is in terms of the convex conjugate $R^*$. Plugging in an arbitrary, possibly non-convex regularizer $R$, through (12) this means that the algorithm would work as if the *convex*

**Algorithm 1:** Fast Mumford-Shah Minimization

**Input**: Image $f : \Omega \to \mathbb{R}^k$, parameters $0 < \alpha, \lambda \leq \infty$ and $\varepsilon > 0$ $(d = \dim \Omega)$
**Initialize**: $u^0 = f$, $\overline{u}^0 = u^0$, $p^0 = 0$, $\tau_0 = \frac{1}{2d}$, $\sigma_0 = \frac{1}{2}$

**for** $n \geq 0$ **until** $\|u^{n+1} - u^n\| < \varepsilon$ **do**

`// Dual ascent in` $p$

$\tilde{p}_i(x) = p_i^n(x) + \sigma_n \nabla \overline{u}_i^n(x)$

$$p^{n+1}(x) = \begin{cases} \frac{2\alpha}{\sigma_n + 2\alpha}\, \tilde{p}(x) & \text{if } \|\tilde{p}(x)\| \leq \sqrt{\frac{\lambda}{\alpha}\, \sigma_n(\sigma_n + 2\alpha)} \\ 0 & \text{else} \end{cases}$$

`// Primal descent in` $u$

$\tilde{u}_i(x) = u_i^n(x) + \tau_n \operatorname{div} p_i^{n+1}(x)$

$u^{n+1}(x) = \big(\tilde{u}(x) + 2\tau_n f(x)\big) / \big(1 + 2\tau_n\big)$

`// Extrapolation step`

$\theta_n = \frac{1}{\sqrt{1+4\,\tau_n}}, \quad \tau_{n+1} = \theta_n \tau_n, \quad \sigma_{n+1} = \sigma_n / \theta_n$

$\overline{u}^{n+1} = u^{n+1} + \theta_n \big(u^{n+1} - u^n\big)$

**end**

*envelope* of $R$ had been given as the regularizer instead of $R$. For example, the convex envelope of $R_{MS}$ is the zero function, so the algorithm for $R_{MS}$ would behave as if there was no regularization at all.

To make the algorithm applicable also to non-convex $R$, the central idea is to reformulate step (15) by reducing the proximal operator of $R^*$ to that of $R$. For this, we make use of Moreau's identity [21]:

$$\operatorname{prox}_{\sigma,\, R^*}(p) = p - \sigma \operatorname{prox}_{\frac{1}{\sigma},\, R}(p/\sigma)\,. \tag{20}$$

Applying this to (15), the algorithm only becomes written in a slightly different way. For convex $R$ the sequence $u^n$ is still the same and all convergence guarantees still hold.

### 3.2 Proposed Algorithm for the MS-Energy

The main advantage of the reformulating the step (15) by (20) is that the algorithm now becomes applicable also to non-convex regularizers, because only $R$ itself enters the right hand side of (20), and not $R^*$. We propose to apply the reformulated algorithm to the Mumford-Shah case with the *non-convex* regularizer $R_{MS}$ in (7).

It remains to compute the right hand side of (20), and for this we need to compute the proximal operator

$$\operatorname{prox}_{\tau,\, R_{MS}}(\tilde{g}) = \arg\min_{g \in \mathbb{R}^{d \times k}} \frac{|g - \tilde{g}|^2}{2\tau} + \min\big(\alpha |g|^2, \lambda\big).$$

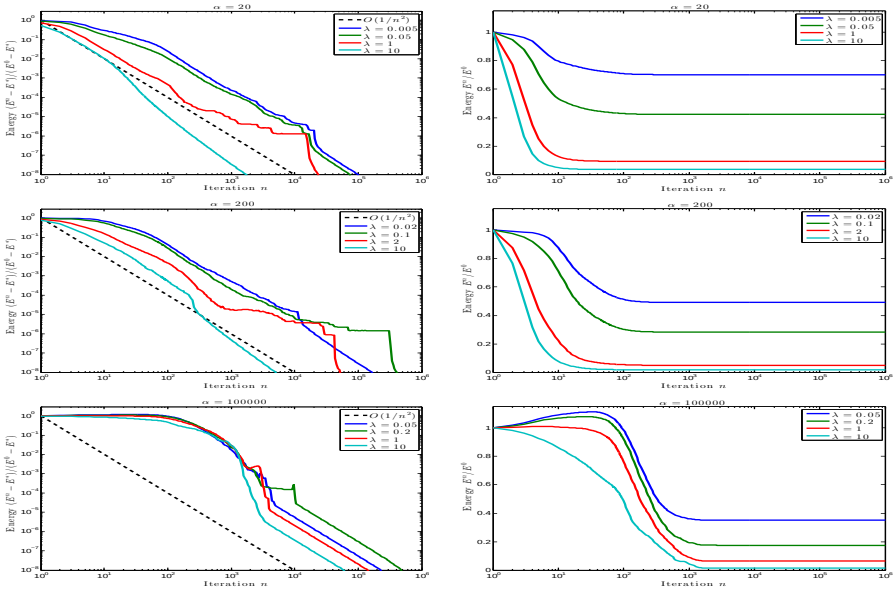


**Fig. 5. Convergence of the proposed algorithm.** For different values of $\alpha$ and $\lambda$ the algorithm steadily decreases the energy (6) (solid lines) to a limit energy $E^*$, here approximated by $E^* = E(u^{10^6})$. Experimentally the convergence rate is roughly $\mathcal{O}(1/n^2)$ (dashed lines).

Although this energy is not convex, its simple structure allows us to find an *explicit formula* for the minimizer:

$$\operatorname{prox}_{\tau,\, R_{MS}}(\tilde{g}) = \begin{cases} \frac{1}{1+2\tau\alpha}\,\tilde{g} & \text{if } |\tilde{g}| \le \sqrt{\frac{\lambda}{\alpha}(1+2\tau\alpha)}, \\ \tilde{g} & \text{else}. \end{cases}$$

Inserting this into (20) results in

$$\operatorname{prox}_{\sigma,\, R^*_{MS}}(\tilde{p}) = \begin{cases} \frac{2\alpha}{\sigma+2\alpha}\,\tilde{p} & \text{if } |\tilde{p}| \le \sqrt{\frac{\lambda}{\alpha}\,\sigma(\sigma+2\alpha)}, \\ 0 & \text{else}. \end{cases} \tag{21}$$

For the piecewise constant limit case $\alpha \to \infty$, the right hand side of (21) simplifies to

$$\operatorname{prox}_{\sigma,\, R^*_{MS\,0}}(\tilde{p}) = \begin{cases} \tilde{p} & \text{if } |\tilde{p}| \le \sqrt{2\lambda\sigma}, \\ 0 & \text{else}. \end{cases} \tag{22}$$

We obtain Algorithm 1 for the minimization of the proposed Mumford-Shah functional (6). The algorithms always converges experimentally, with roughly $O(1/n^2)$ energy rates, see Fig. 5. Although we do not have a proof of convergence yet, nonetheless we can prove the boundedness of $u^n$ for the piecewise smooth case $\alpha < \infty$:

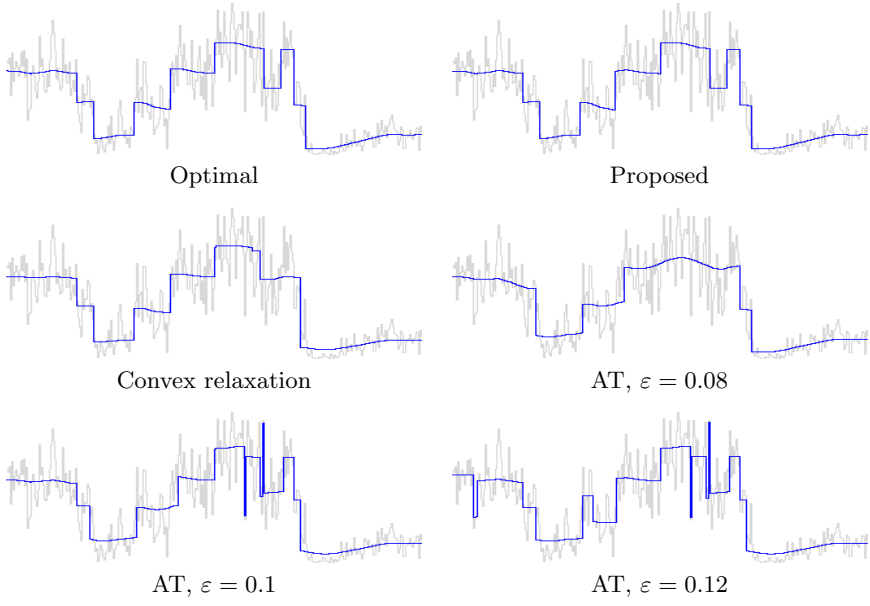


**Fig. 6. Denoising quality in dimension $d = 1$ with $\alpha = 1000$ and $\lambda = 0.3$.** Our approach yields a solution most closely resembling the optimal solution. We do not have any further parameters, while in contrast the AT method is highly sensitive w.r.t. $\varepsilon$.

**Proposition 1.** *The sequence $(u^n, p^n)$ generated by Algorithm 1 is bounded and thus compact for $\alpha < \infty$, for instance it* has a convergent subsequence.

*Proof.* See appendix.

In the experiments, we also observed convergence for $\alpha = \infty$. We terminate once the solution does not change significantly anymore, where

$$\|\tilde{u}\| := \frac{1}{|\Omega|} \sum_{x \in \Omega} \sum_{1 \le i \le k} |\tilde{u}_i(x)|.$$

We set $\varepsilon = 5 \cdot 10^{-5}$. To reduce run time, we compute $\|u^{n+1} - u^n\|$ only once every 10 iterations. We use a parallel CUDA implementation on the NVIDIA GTX 680 GPU. The stopping criterion is met already in a few hundred iterations (for $640 \times 480$ color images).

## 4 Experiments

### 4.1 Energy in One-Dimensional Case

In dimension $d = 1$ the MS energy (6) can be efficiently minimized using dynamic programming [20]. Fig. 6 compares the results of our approach, convex relaxation and the Ambrosio-Tortorelli approximation, with our result most closely

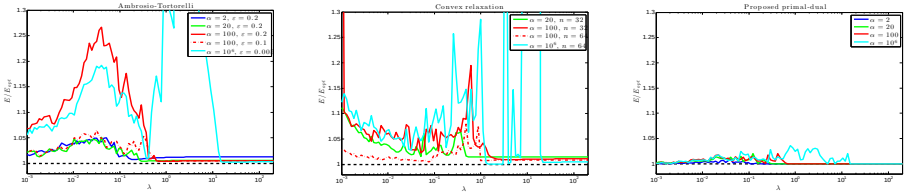


**Fig. 7. Energy optimality in dimension $d = 1$.** For the proposed method, in practice the obtained energy is always at most 5% off the optimal energy $E_{\text{opt}}$. The AT and convex relaxation methods both yield higher energies. In addition, the $\varepsilon$-dependancy of AT is sensitive for big $\alpha$ (red lines), and the convex relaxation method needs a sufficiently fine range discretization into $n$ levels (red lines).

resembling the optimal solution (input is the lower row of the image in Fig. 3). In fact, in practice our obtained energy is always at most 5% off the optimal energy, Fig. 7.

### 4.2 Comparison with Convex Relaxation

The method [1] computes *relaxed* solutions of (1) through convexification. Since it discretizes the color space, it requires huge amounts of memory ($> 2$ GB for VGA resolution and 32 levels for each channel) and is slow (30–300 seconds). In contrast, we only need about 15 MB and the run time is real-time. In general, our results are always visually similar with [1], see Fig. 4, which indicates the appropriateness of our method for solving the MS model. While the results of [1] are often too smooth and blurry, even for the piecewise constant case $\alpha = \infty$, our model yields results with visually well-defined sharp transitions.

### 4.3 Comparison with Ambrosio-Tortorelli

The AT approximation has an additional parameter $\varepsilon$. It is not clear how to choose a suitable $\varepsilon$ given $\lambda$ and $\alpha$ in order to obtain "true" minimizers of the MS model (1). The results depend on $\varepsilon$ and this dependancy becomes more and more sensitive for larger $\alpha$, see Fig. 2. In contrast, our method comes without any additional parameters, is stable in $\lambda$ and $\alpha$, and also turns out to be about 5–10 times faster. While the AT model requires alternative minimization and the two involved subproblems can be solved only approximately, we propose a direct well-defined algorithm which is also applicable for the case $\alpha = \infty$.

### 4.4 Comparison with $L_0$-Smoothing

The method [10] is devised for the piecewise constant case $\alpha = \infty$, but the solutions depend on a parameter $\kappa > 1$. To reduce the smooth variations, it must be chosen near 1, which increases the run time significantly. In contrast, our method is directly applicable with $\alpha = \infty$ and solutions are guaranteed to

Input [10], $\kappa = 1.01$ Proposed

**Fig. 8. Grid artifacts of $L_0$-Smoothing [10].** The method [10] (center, $\lambda = 0.035$) measures edge lengths with the anisotropic $l^1$-norm. This leads to significant artifacts as result edges tend to align with coordinate axes. We use the $l^2$-norm (right, $\lambda = 0.82$, $\alpha = \infty$), yielding more natural results.

be piecewise constant, see Fig. 3. Because of the heuristic decoupling there is no clear relation between the parameter $\lambda$ in (5) (through (4)) and the original parameter $\lambda$ in (1). Since in (5) the $g$-subproblem is solved exactly, one can see that $R_0(g)$ then measure edges in the *anisotropic* $l^1$-norm. This leads to grid artifacts, Fig. 8, in contrast to our primal-dual formulation.

### 4.5 Real-Time Unsupervised Image Segmentation

The proposed method naturally includes the piecewise constant limit $\alpha = \infty$, see Fig. 3. This allows one to segment an image with automatic selection of the most suitable color models in real-time. In contrast, results of the $L_0$-Smoothing always have smooth variations, which can only be avoided at considerable increase in run time. Also, [10] approximates (1) worse and worse when increasing the edge set penalization $\lambda$. This leads to artifacts such failing to get rid of small scale structures (grass) despite an overall smooth solution.

### 4.6 Real-Time Video Cartooning

The proposed method remains fast even when approaching the cartoon limit of the MS model (1), i.e. for large $\alpha$, with more than 20 frames per second on three GPUs, and still about 5–10 Hz on a single GPU. This allows us to apply the MS model to videos in real time, processing them frame by frame, see Fig. 9.

To reduce artificial solution variations from frame to frame we employ temporal regularization:

$$E_{MS}(u) + \gamma |u - u_{\text{prev}}|^{1.5} \tag{23}$$

with a small $\gamma > 0$, e.g. 0.4. This only affects the proximal operator (16), so that line 5 of the Algorithm 1 becomes slightly altered, see appendix. A practical side effect, in addition to regularization, is that this further accelerates the convergence of our algorithm. Furthermore, to accommodate for low contrast edges, we use an adaptive variant of (7): $\min\left(\alpha|g|^2, \lambda w(x)\right)$ with the weights $w(x) := \exp\left(-|\nabla f(x)|/s\right)$ and $s := \frac{1}{|\Omega|}\sum_{x\in\Omega}|\nabla f(x)|$.

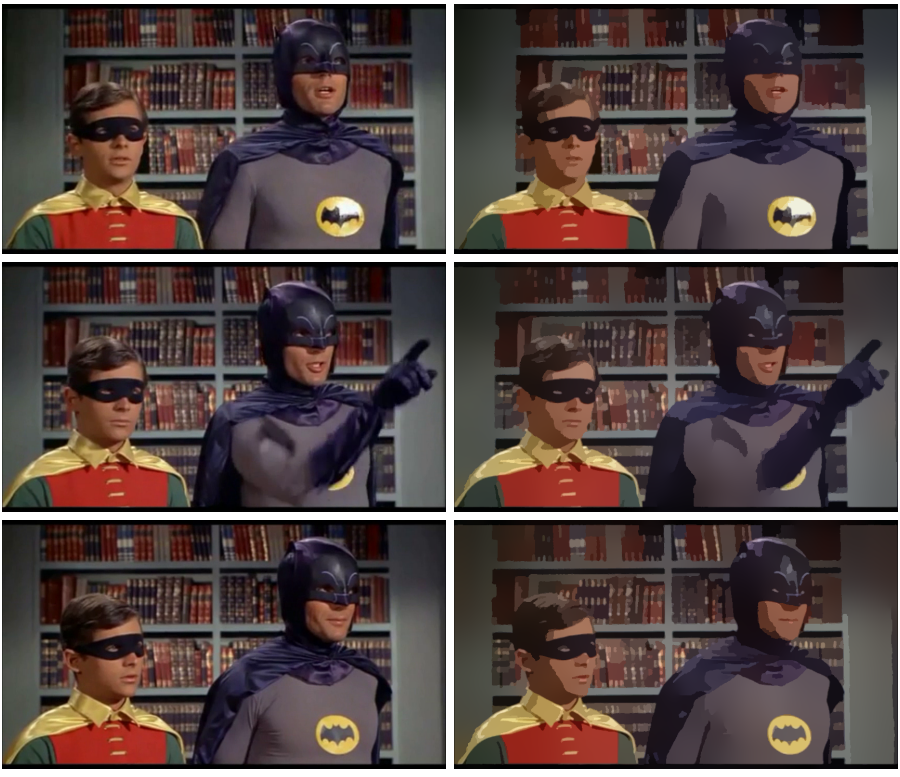

**Fig. 9. Video cartooning.** The cartoon limit $\alpha \to \infty$ of our method allows to compute real-time cartoonings of video sequences (**top:** an input frame, **bottom:** cartooned frame), $\lambda = 1.5$ and $\alpha = 500$. Average run time per frame: 0.08 s.

## 5 Conclusion

We proposed an algorithm which allows to efficiently minimize the piecewise smooth and piecewise constant Mumford-Shah model. Using Moreau's identity to simplify respective proximal operators, we were able to generalize a recent primal-dual algorithm from convex to non-convex optimization. The resulting method computes piecewise smooth or piecewise constant approximations of color-images at 15-20 Hz at VGA resolution. Compared to existing convex relaxation methods, it does not require a discretization of color values and it is orders of magnitude faster. In contrast to the popular Ambrosio-Tortorelli approach, it does not require an epsilon-approximation and pursues a direct rather than an alternating minimization scheme. Numerous experiments demonstrate that the proposed algorithm is well-suited to perform discontinuity-preserving smoothing and real-time video cartooning.

## References


1. Strekalovskiy, E., Chambolle, A., Cremers, D.: A convex representation for the vectorial Mumford-Shah functional. In: CVPR. (June 2012)
2. Ambrosio, L., Tortorelli, V.M.: Approximation of functionals depending on jumps by elliptic functionals via $\Gamma$–convergence. Communications on Pure and Applied Mathematics **43** (1990) 999–1036
3. Mumford, D., Shah, J.: Optimal approximations by piecewise smooth functions and associated variational problems. Comm. Pure Appl. Math. **42**(5) (1989) 577–685
4. Blake, A., Zisserman, A.: Visual Reconstruction. MIT Press (1987)
5. Morel, J.M., Solimini, S.: Variational Methods in Image Segmentation. Birkhäuser, Boston (1995)
6. Vese, L., Chan, T.: A multiphase level set framework for image processing using the Mumford–Shah functional. Int. J. of Computer Vision **50**(3) (2002) 271–293
7. Grady, L., Alvino, C.: The piecewise smooth Mumford-Shah functional on an arbitrary graph. IEEE Transactions on Image Processing **18**(11) (2009)
8. Chambolle, A., Pock, T.: A first-order primal-dual algorithm for convex problems with applications to imaging. J. Math. Imaging Vis. **40** (2011) 120–145
9. Brook, A., Kimmel, R., Sochen, N.A.: Variational restoration and edge detection for color images. Journal of Mathematical Imaging and Vision **18**(3) (2003) 247–268
10. Xu, L., Lu, C., Xu, Y., Jia, J.: Image smoothing via $l_0$ gradient minimization. In: Proceedings of SIGGRAPH Asia, ACM (2011) 174:1–174:12
11. Alberti, G., Bouchitté, G., Dal Maso, G.: The calibration method for the Mumford-Shah functional and free-discontinuity problems. Calc. Var. Partial Differential Equations **16**(3) (2003) 299–333
12. Giaquinta, M., Modica, G., Souček, J.: Cartesian currents in the calculus of variations I, II. Volume 37-38 of Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Springer-Verlag, Berlin (1998)
13. Lellmann, J., Schnörr, C.: Continuous multiclass labeling approaches and algorithms. SIAM J. Imaging Sciences **4**(4) (2011) 1049–1096
14. Chambolle, A., Cremers, D., Pock, T.: A convex approach to minimal partitions. SIAM Journal on Imaging Sciences **5**(4) (2012) 1113–1158
15. Zach, C., Hane, C., Pollefeys, M.: What is optimized in convex relaxations for multilabel problems: Connecting discrete and continuously inspired map inference. Pattern Analysis and Machine Intelligence, IEEE Transactions on **36**(1) (2014) 157–170
16. Pock, T., Cremers, D., Bischof, H., Chambolle, A.: An algorithm for minimizing the piecewise smooth Mumford-Shah functional. In: ICCV. (2009)
17. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. on Patt. Anal. and Mach. Intell. **23**(11) (2001) 1222–1239
18. Komodakis, N., Tziritas, G.: Approximate labeling via graph-cuts based on linear programming. IEEE Trans. on Patt. Anal. and Mach. Intell. **29**(8) (2007) 1436–1453
19. Veksler, O.: Graph cut based optimization for mrfs with truncated convex priors. In: CVPR. (2007)
20. Chambolle, A.: Image segmentation by variational methods: Mumford and Shah functional and the discrete approximations. SIAM J. Appl. Math. **55**(3) (June 1995) 827–863
21. Rockafellar, R.T.: Convex Analysis. Princeton University Press (1996)

# A Proof of Proposition 3.1

*Proof. Boundedness of p:*

First, let us prove that $(p^n)$ is bounded. Consider the update equation for $p^{n+1}$ in line 3 of the algorithm. If $|\tilde{p}(x)| \leq \sqrt{\frac{\lambda}{\alpha}\sigma_n(\sigma_n + 2\alpha)}$, then

$$\begin{aligned} |p^{n+1}(x)| &= \frac{2\alpha}{\sigma_n + 2\alpha}\,|\tilde{p}(x)| \\ &\leq \frac{2\alpha}{\sigma_n + 2\alpha}\sqrt{\frac{\lambda}{\alpha}\sigma_n(\sigma_n + 2\alpha)} \\ &= 2\sqrt{\alpha\lambda\,\frac{\sigma_n}{\sigma_n + 2\alpha}} \\ &\leq 2\sqrt{\alpha\lambda}. \end{aligned} \tag{24}$$

Otherwise, $p^{n+1}(x) = 0$ and the same bound holds. Together with $p^0 = 0$ we obtain that $p$ is uniformly bounded:

$$|p^n(x)| \leq 2\sqrt{\alpha\lambda} \qquad \forall x \in \Omega,\ n \geq 0. \tag{25}$$

*Boundedness of u:*

Using this, we will now prove that also $u$ is bounded. Lines 4 and 5 of the algorithm can be written as

$$u^{n+1}(x) = \frac{u^n(x) + 2\tau_n z^n(x)}{1 + 2\tau_n} \tag{26}$$

with, for $1 \leq i \leq k$,

$$z_i^n(x) := \tfrac{1}{2}\operatorname{div} p_i^{n+1}(x) + f_i(x). \tag{27}$$

Due to (25) also the discrete divergence with backward-differences is bounded ($d = \dim \Omega$):

$$|\operatorname{div} p_i^{n+1}(x)| \leq 2d \cdot 2\sqrt{\alpha\lambda} = 4d\sqrt{\alpha\lambda}, \tag{28}$$

and thus $\operatorname{div} p^{n+1}(x) \in \mathbb{R}^k$ is bounded by

$$|\operatorname{div} p^{n+1}(x)| \leq k \cdot 4d\sqrt{\alpha\lambda} = 4dk\sqrt{\alpha\lambda}. \tag{29}$$

Furthermore,

$$|f(x)| \leq \max_{\hat{x}\in\Omega} |f(\hat{x})| =: M_f. \tag{30}$$

Bounds (29) and (30) together yield

$$\begin{aligned} |z^n(x)| &\leq \tfrac{1}{2}|\operatorname{div} p^{n+1}(x)| + |f(x)| \\ &\leq 2dk\sqrt{\alpha\lambda} + M_f =: M_z. \end{aligned} \tag{31}$$

Define

$$M_{u_0} := \max_{\hat{x}\in\Omega} |u_0(\hat{x})| \tag{32}$$

$$\text{and} \quad M := \max(M_{u_0}, M_z). \tag{33}$$

We will prove that

$$|u^n(x)| \leq M \qquad \forall x \in \Omega,\ n \geq 0 \tag{34}$$

by induction. The bound holds for $n = 0$ by (32) and (33). If (34) holds for some $n \geq 0$, then (26), (31) and (33) yield

$$\begin{aligned} |u^{n+1}(x)| &\leq \frac{|u^n(x)| + 2\tau_n |z^n(x)|}{1 + 2\tau_n} \\ &\leq \frac{M + 2\tau_n M_z}{1 + 2\tau_n} \\ &\leq \frac{M + 2\tau_n M}{1 + 2\tau_n} = M. \end{aligned} \tag{35}$$

This proves (34) for all $n \geq 0$. Together with (25), it follows that the sequence $(u^n, p^n)$ is uniformly bounded. □

# B Video Cartooning

## B.1 Temporal Regularization

In Section 4.6, an additional temporal $\ell^p$-regularization in

$$E^{p,\gamma}_{MS}(u) := D^{p,\gamma}(u) + R_{MS}(\nabla u(x)) \tag{36}$$

with the data term

$$D^{p,\gamma}(u) := \sum_{x\in\Omega} |u(x) - f(x)|^2 + \gamma L^p\big(u(x) - u_{\text{prev}}(x)\big) \tag{37}$$

with

$$L^p(z) := |z|^p, \quad z \in \mathbb{R}^k, \tag{38}$$

is suggested with $p = 1.5$ and a $\gamma > 0$. To use it in the proposed Algorithm 1, this amounts to computing the new proximal operator (16) for the line 5 of the algorithm.

*Prox Operator.* The solution $u := \text{prox}_{\tau,\, D^{p,\gamma}}(\tilde{u})$ is given by

$$u(x) = (1 - t(x))u_{\text{prev}}(x) + t(x)u_0(x) \tag{39}$$

where

$$u_0(x) := \frac{\tilde{u}(x) + 2\tau f(x)}{1 + 2\tau} \tag{40}$$

is the solution without temporal regularization, and

$$t(x) := \arg\min_{t\geq 0} \left\{ \frac{(t-1)^2}{2} + \hat{\tau}(x)t^p \right\} \in [0, 1] \tag{41}$$

$$= \begin{cases} \max(0, 1 - \hat{\tau}) & \text{if } p = 1, \\ \frac{1}{1+2\hat{\tau}} & \text{if } p = 2, \\ \frac{1}{\left(\frac{3\hat{\tau}}{4} + \sqrt{1 + (\frac{3\hat{\tau}}{4})^2}\right)^2} & \text{if } p = 1.5 \end{cases} \tag{42}$$

with $\hat{\tau}(x) := \frac{\gamma\tau}{1+2\tau}|u_0(x) - u_{\text{prev}}(x)|^{p-2}$.

*Regularization Effect for Different p.* With $p = 1$ the influence of $u_{\text{prev}}$ is too strong if the current-frame solution without regularization $u_0(x)$ would be similar to the previous-frame solution $u_{\text{prev}}(x)$. In practice, parts of the solutions tend to "stick" to their location over several frames. If the new value without regularization $u_0(x)$ would be similar to the previous-frame solution $u_{\text{prev}}(x)$, then $\hat{\tau}(x)$ is big and from (42) we see that $t(x) = 0$ and $u(x) = u_{\text{prev}}(x)$, i.e. the old value persists.

With $p = 2$ the previous solution $u_{\text{prev}}$ always has the same influence on the current solution $u(x)$, even if $u_0(x) - u_{\text{prev}}(x)$ is big. In practice, the previous frame appears as a "ghost" image overlaid over the current solution.

Value $p = 1.5$ performs the best in practice. If the current-frame solution without regularization $u_0(x)$ would be similar to the previous solution $u_{\text{prev}}(x)$, $u_0(x)$ still has an influence on the actual current-frame solution $u(x)$. On the other hand, for dissimilar $u_0(x)$ and $u_{\text{prev}}(x)$, $u_{\text{prev}}(x)$ has almost no influence on $u(x)$.

### B.2 Test Video

We have tested the video cartooning using our method on a real-world sequence (807 frames). Three frames of the sequence together with their cartooning versions are shown in Fig. 9. The image resolution is VGA ($640 \times 480$). The parameters are set to $\lambda = 1.5$ and $\alpha = 500$, where $\alpha$ is chosen large in order for the results to be almost piecewise constant. For this sequence and these parameters, the proposed algorithm achieves an average frame rate of 12.6 Hz on a single GPU, and 25.1 Hz using three GPUs in parallel.

The temporal regularization has an accelerating effect on convergence, making the algorithm about 22% faster: Without temporal regularization, the frame rate is 10.3 Hz on one GPU, and 20.1 Hz using three GPUs.

# C Additional Results

## C.1 Edge Highlighting

For visualization of regions of piecewise smoothness, we overlay the edge set $K_{MS}$ in (8) onto the result image $u$. If $x \in K_{MS}$ then $|\nabla u(x)| \geq \sqrt{\lambda/\alpha}$, and for forward differences we always have $|\nabla u(x)| \leq \sqrt{2}$, so that

$$1 \leq \frac{|\nabla u(x)|}{\sqrt{\lambda/\alpha}} \leq \frac{\sqrt{2}}{\sqrt{\lambda/\alpha}} \tag{43}$$

and thus

$$0 \leq \log\left(\frac{|\nabla u(x)|}{\sqrt{\lambda/\alpha}}\right) \leq \log\left(\frac{\sqrt{2}}{\sqrt{\lambda/\alpha}}\right). \tag{44}$$

Therefore

$$c \cdot \log\left(\frac{|\nabla u(x)|}{\sqrt{\lambda/\alpha}}\right) \in [0,1] \tag{45}$$

with $c := \left(\log \frac{\sqrt{2}}{\sqrt{\lambda/\alpha}}\right)^{-1}$, and the value (45) increases with increasing $|\nabla u(x)|$, so that it can serve as an edge indicator.

For $x \in K_{MS}$ we multiply the RGB value $u(x)$ by

$$1 - c \cdot \log\left(\frac{|\nabla u(x)|}{\sqrt{\lambda/\alpha}}\right) \in [0,1] \tag{46}$$

For $x \notin K_{MS}$ the value $u(x)$ is left unchanged. This paints the points $x \in \Omega$ with strong edges darker.

## C.2 Result for Different Parameters

Fig. 10 shows the effect of choosing different parameters for the proposed Mumford-Shah minimization algorithm. As expected, for increasing $\alpha$ the solution becomes more and more "flat" between the edges, and is piecewise constant for $\alpha = \infty$. An increasing $\lambda$ penalizes the just set length more and more, so that the number of color discontinuities is reduced and the solution becomes smoother over larger regions.

## C.3 Piecewise Constant Approximations

Fig. 11 shows the computed segmentations with automatic color model selection, using the proposed algorithm with $\alpha = \infty$ and $\lambda = 0.2$. We highlight the edges as described in Section C.1, with $\sqrt{\lambda/\alpha}$ replaced by 0.03 (because $\sqrt{\lambda/\alpha}$ is zero for $\alpha = \infty$).

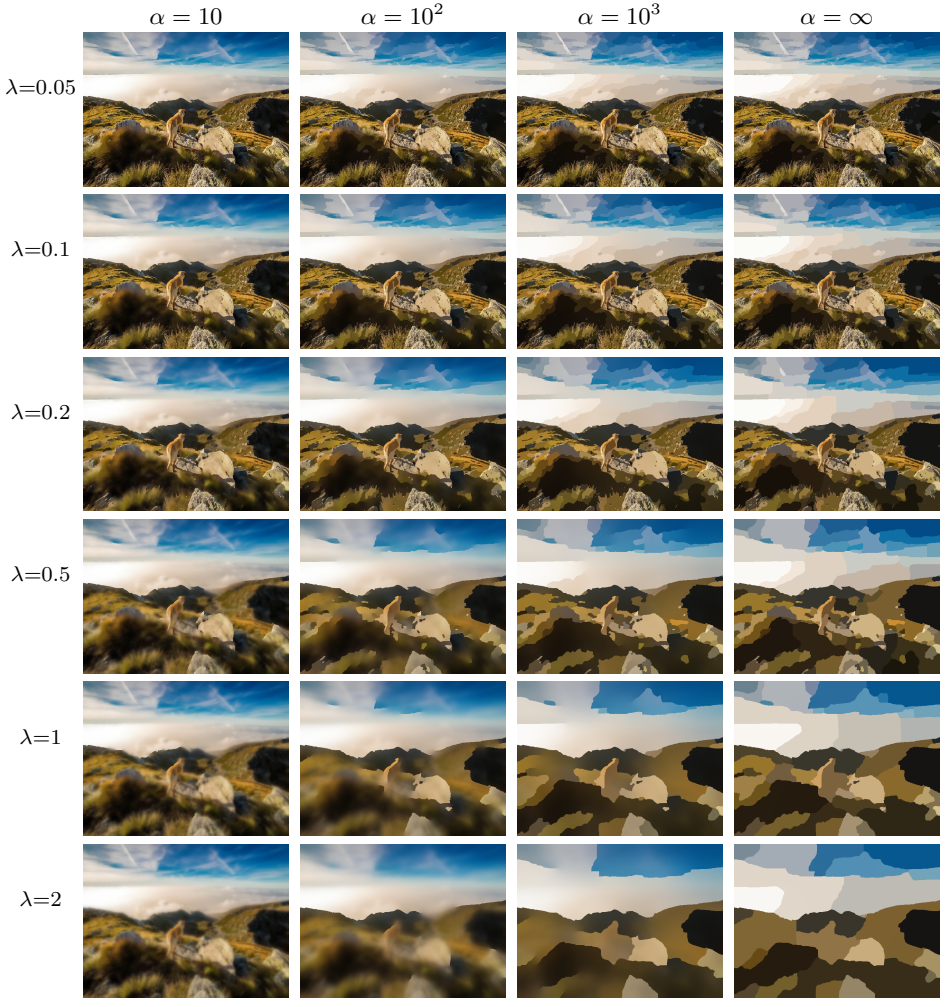


**Fig. 10. Mumford-Shah results using the proposed algorithm for different parameters $\alpha$ and $\lambda$.** The input image is the upper left image of Fig. 11. The number of discontinuities decreases for larger $\lambda$, resulting in just smoothing by quadratic regularization for sufficiently large $\lambda$. The solution becomes more cartoon-like for larger $\alpha$, i.e. smooth variations disappear and only sharp discontinuities remain.

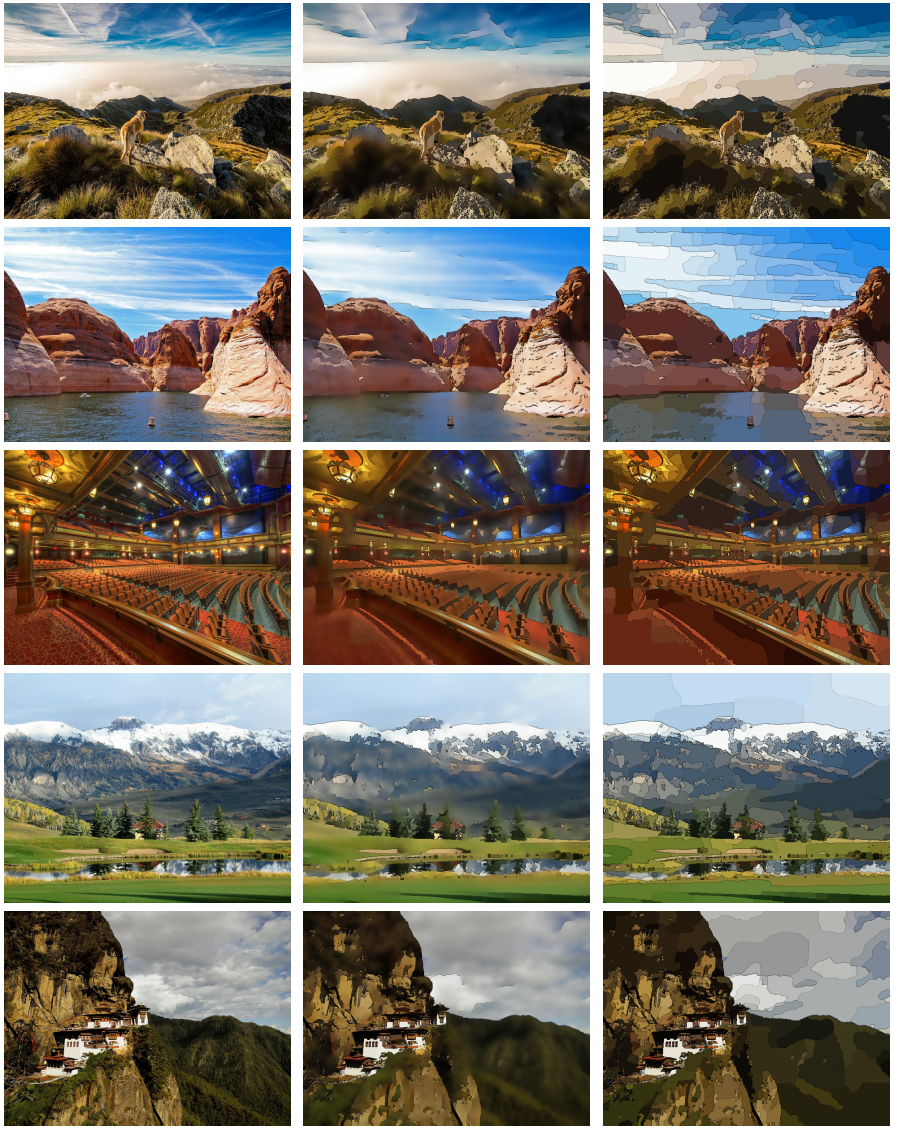


**Fig. 11. Piecewise smooth and piecewise constant approximations using the Mumford-Shah model.** Results are computed using the proposed algorithm with $\lambda = 0.1$. Edges are highlighted as described in Section C.3.

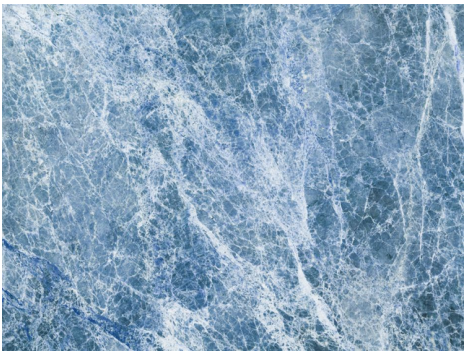

**Fig. 12. Input image for Fig. 13 and 14.**

### C.4 Comparison on Texture Images

Fig. 13 compares piecewise smooth approximations using the proposed approach, the convex relaxation method, and the Ambrosio-Tortorelli approximation on a texture image of Fig. 12. Note that the AT method is only applicable for $\alpha < \infty$. Fig. 14 compares piecewise constant approximation by $L_0$-smoothing, which is only applicable for $\alpha = \infty$.

Only the proposed method and the convex relaxation are able to compute quality solutions, which do not have speckle or grid artifacts. Note that the convex relaxation needs to discretize the color channels into $n$ levels for each channel, with in practice $n = 32$ or $n = 64$. This leads to significant memory consumption (GBs) and runtimes (30–120 seconds). Furthermore, in the piecewise constant case $\alpha = \infty$ the solution still contains smooth variations, likely because of a finite range discretization.

In contrast, the proposed method runs in real-time, does not discretize the color range, and the solutions are indeed constant between the edges.

Thus, the proposed approach is the only one which computes quality solutions quickly, outperforming the current state-of-the-art methods (AT, convex relaxation, $L_0$-smoothing).

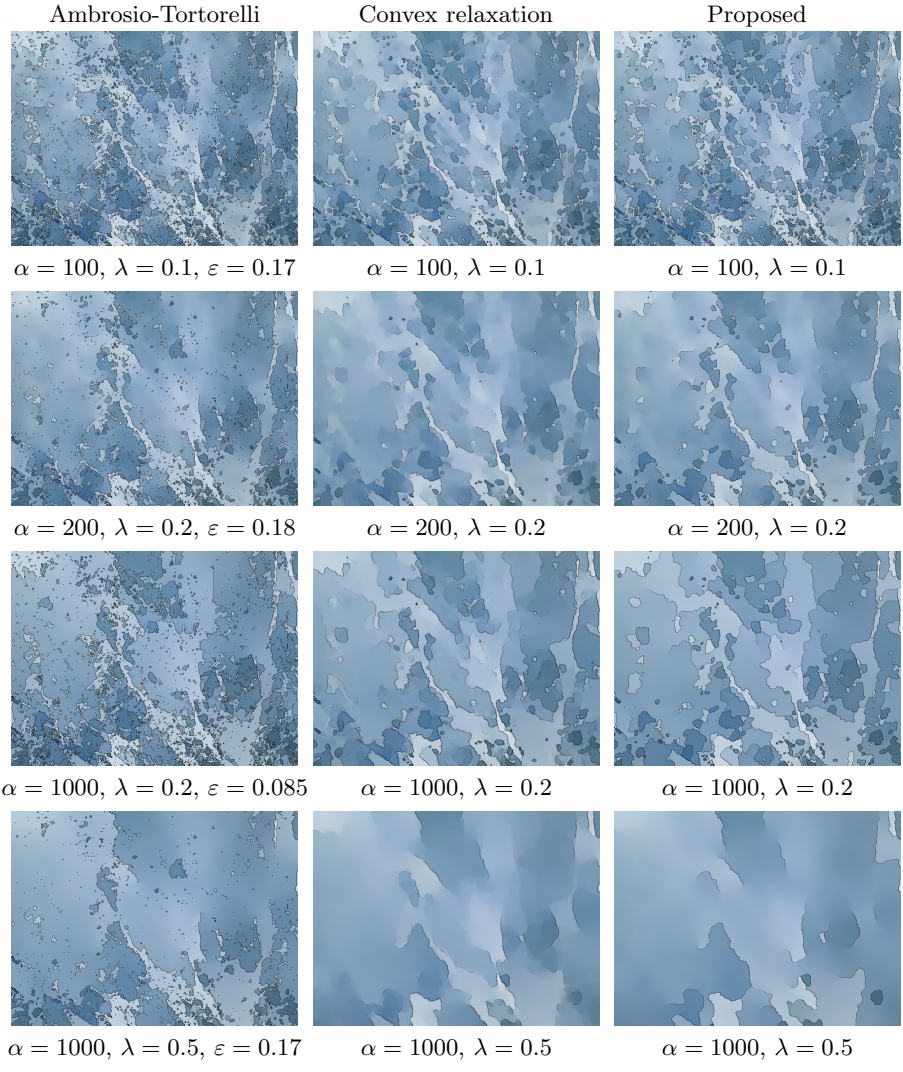


**Fig. 13. Piecewise smooth approximations for a texture image input shown in Fig. 12.** Results using the proposed model, convex relaxation and the Ambosio-Tortorelli approximation (edge highlighting as in Section C.3). Our results is always similar to the convex relaxation result, while the AT approximation depends on $\varepsilon$ (unclear how to choose suitably) and always has speckle in an otherwise smooth solution.

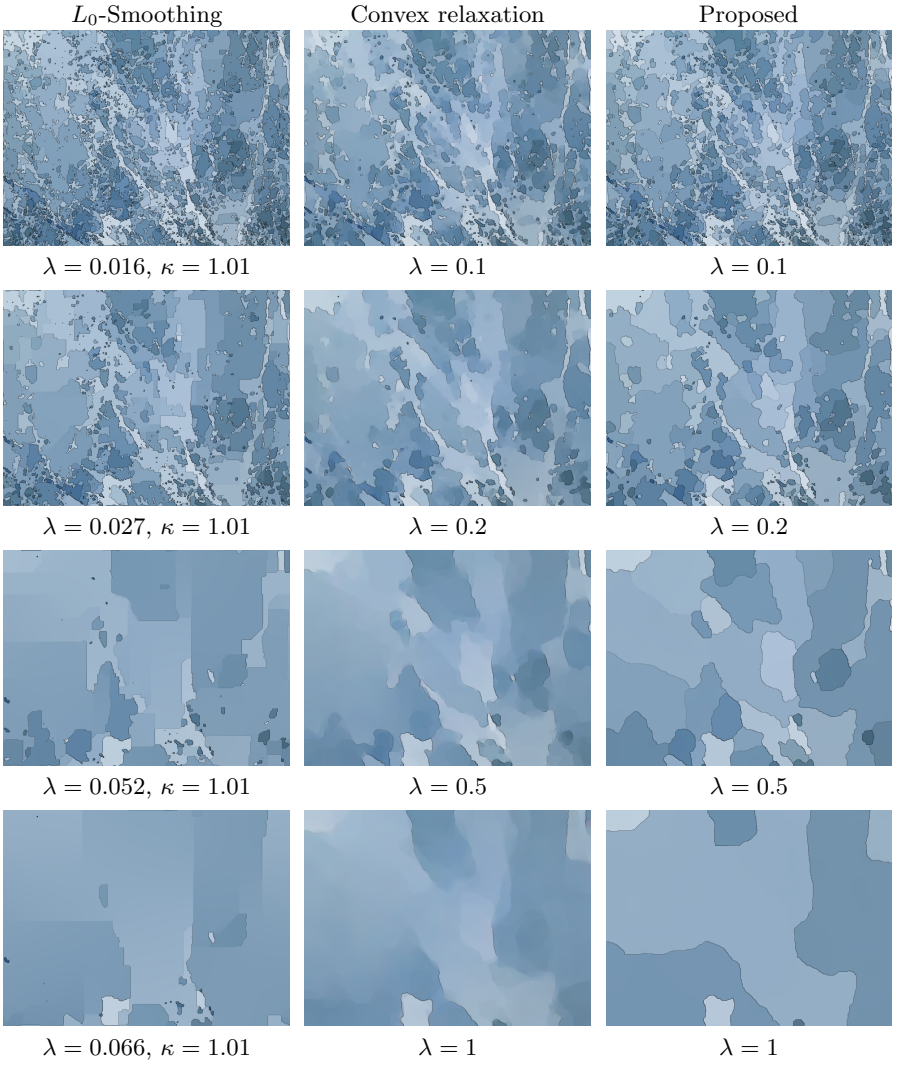


**Fig. 14. Piecewise constant approximations ($\alpha = \infty$) with the Mumford-Shah model for a texture image in Fig. 12.** Edge highlighting as in Section C.3. The proposed method computed similar solutions to the convex relaxation. In contrast, the $L_0$-smoothing prefers grid aligned edges, depends on a parameter $\kappa$, and there is no clear relationship between the $\lambda$ of the Mumford-Shah model and the $\lambda$ of the $L_0$-smoothing approximation.