Geodesic Image and Video Editing

ANTONIO CRIMINISI and, TOBY SHARP and, CARSTEN ROTHER
Microsoft Research Ltd, CB3 0FB, Cambridge, UK
and
PATRICK PÉREZ
Technicolor Research and Innovation, F-35576 Cesson-Sévigné, France

This paper presents a new, unified technique to perform general edge-sensitive editing operations on n-dimensional images and videos efficiently.

The first contribution of the paper is the introduction of a generalized geodesic distance transform (GGDT), based on soft masks. This provides a *unified* framework to address several, edge-aware editing operations. Diverse tasks such as de-noising and non-photorealistic rendering, are all dealt with fundamentally the same, fast algorithm. Second, a new, geodesic, symmetric filter (GSF) is presented which imposes contrast-sensitive spatial smoothness into segmentation and segmentation-based editing tasks (cutout, object highlighting, colorization, panorama stitching). The effect of the filter is controlled by two intuitive, geometric parameters. In contrast to existing techniques, the GSF filter is applied to real-valued pixel likelihoods (soft masks), thanks to GGDTs and it can be used for both interactive and automatic editing. Complex object topologies are dealt with effortlessly. Finally, the parallelism of GGDTs enables us to exploit modern multi-core CPU architectures as well as powerful new GPUs, thus providing great flexibility of implementation and deployment. Our technique operates on both images and videos, and generalizes naturally to n-dimensional data.

The proposed algorithm is validated via quantitative and qualitative comparisons with existing, state of the art approaches. Numerous results on a variety of image and video editing tasks further demonstrate the effectiveness of our method.

Categories and Subject Descriptors: I.4.6 [**Segmentation**]: Image processing and computer vision; I.4.4 [**Restoration**]: Image processing and computer vision; I.3.3 [**Picture/Image Generation**]: Image processing and computer vision

General Terms: Image and video, segmentation, non-photorealistic rendering, restoration

Additional Key Words and Phrases: Geodesic distance, geodestic segmentation, tooning, denoising

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

1. INTRODUCTION AND LITERATURE SURVEY

Recent years have seen an explosion of research in Computational Photography, with many exciting new techniques been invented to aid users accomplish difficult image and video editing tasks effectively. Much attention has been focused on: segmentation [Boykov and Jolly 2001; Bai and Sapiro 2007; Grady and Sinop 2008; Li et al. 2004; Rother et al. 2004; Sinop and Grady 2007; Wang et al. 2005], bilateral filtering [Chen et al. 2007; Tomasi and Manduchi 1998; Weiss 2006] and anisotropic diffusion [Perona and Malik 1990], non-photorealistic rendering [Bousseau et al. 2007; Wang et al. 2004; Winnemoller et al. 2006], colorization [Yatziv and Sapiro 2006; Levin et al. 2004; Luan et al. 2007], image stitching [Brown et al. 2005; Agarwala et al. 2004] and tone mapping [Lischinski et al. 2006]. Despite the many, different algorithms, all those tasks are related to one another by the common goal of obtaining spatially-smooth, edge-sensitive outputs (*e.g.*, a de-noised image, a segmentation map, a flattened texture, a smooth stitching map etc. See fig. 1). Building upon such realization, this paper proposes a new algorithm to address *all* those applications in a *unified* manner. The advantage of such unified approach is that the core processing engine needs be written and optimized only once, while maintaining a wide spectrum of applications.

Edge-sensitive and spatially smooth image editing can be achieved by modeling images as Markov Random Fields [Szeliski et al. 2007]. However, solving an MRF involves time-consuming energy minimization algorithms such as graph-cut [Kolmogorov and Zabih 2004] or belief propagation [Felzenszwalb and Huttenlocher 2004] in case of discrete labels, large sparse linear system solvers in case of continously valued MRFs, *e.g.* [Grady 2006; Szeliski 2006]. Today's image editing applications are required to run efficiently on image sizes up to 20 Mpixels, and unfortunately none of the existing algorithms scale well to such resolutions.[1] In order to address the efficiency problem researchers have resorted to approximate multi-resolution algorithms, with unavoidable loss of accuracy [Lombaert et al. 2005; Kopf et al. 2007; Liu et al. 2009]. When processing video frames, additional efficiency may be gained via dynamic MRFs [Juan and Boykov 2006; Kohli and Torr 2007].

Inspired by the work in [Bai and Sapiro 2007] we impose edge-sensitive smoothness by means of geodesic distance transform (GDT), thus avoiding energy minimization altogether. In contrast to *e.g.*, graph-cut our algorithm's memory and runtime requirements are both linear in the number of pixels. This allows us to work directly with the full image resolution and avoid loss of detail. The algorithm proposed here differs from that in [Bai and Sapiro 2007] in five important ways: i) It imposes spatial smoothness via geodesic filtering, ii) It is not limited to interactive segmentation and can be applied to automatic segmentation tasks. iii) It overcomes Bai's

[1]Graph-cut has $O(N^2)$ memory requirement and a worst case runtime of $O(N^3)$, where N is the number of pixels.

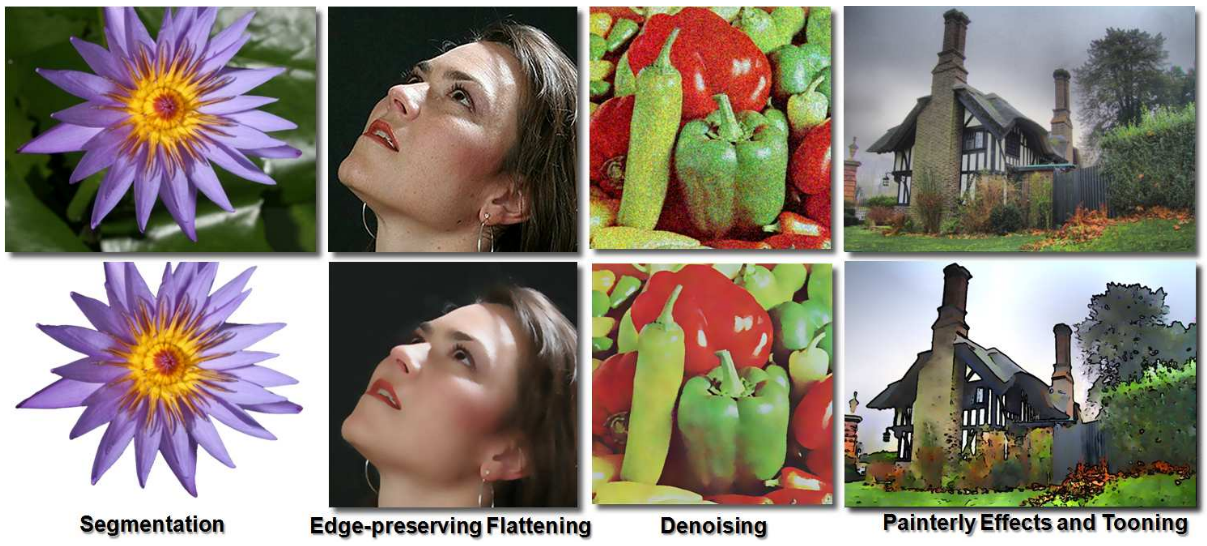


Fig. 1. **Examples of geodesic image editing. (Top row)** original images. (lady's original photo courtesy of D. Weiss). **(Bottom row)** processed images. Segmentation, filtering and non photo-realistic effects can all be achieved very efficiently with the single framework proposed in this paper. The algorithm's efficiency enables processing high-resolution, n-dimensional images at interactive rates.

topology restrictions (details later). iv) It handles editing tasks other than segmentation. Finally, v) our algorithm exploits modern parallel computer architectures to achieve greater efficiency.

When possible, a common technique to increase an algorithm's execution speed is to implement it on the GPU. For example a fast GPU-based bilateral filtering is described in [Chen et al. 2007], and GPU-driven abstraction in [Winnemoller et al. 2006]. However, the choice whether to use the CPU or the GPU is driven not only by efficiency issues but also by portability ones. In fact, CPUs are still far more prevalent than GPUs (*e.g.*, mobile devices and many laptops have CPUs but no GPUs). Also, the large diversity between GPUs makes it difficult to robustly deploy GPU-based software to a wide audience. On the other hand, modern GPUs are quickly moving towards becoming generic computing devices. The parallel algorithm proposed here is based on geodesic distance transforms which can be computed efficiently on the GPU [Weber et al. 2008]. As described later our algorithm is also extremely efficient when implemented on modern multi-core CPUs. This provides great flexibility of implementation and deployment.

This paper builds upon the segmentation algorithm in [Criminisi et al. 2008] and extends it in the following ways: i) it imposes spatial smoothness while avoiding expensive energy minimization; ii) it provides a single framework to address many different editing tasks; iii) it provides quantitative comparisons with competing state of the art algorithms.

The remainder of the paper is organized as follows. Section 2 provides background on distance transforms in digital images, before introducing their generalization to soft masks. The efficient implementations of such transforms is also discussed. Section 3 describes the geodesic symmetric filter (GSF) that exploits GGDTs for image segmentation. Please note that both the GGDT and the GSF operator were already presented in [Criminisi et al. 2008] but in the context of energy minimization for segmentation. Here we repeat the mathematical definitions for clarity and further elaborate on the link between GGDT and classic GDT. Furthermore, we explore the use of these recently introduced tools in the context of image editing while negating the need for complex energy minimization. Implementations of various image and video editing operations using these two tools are described in Section 4. Quantitative validation and comparative experiments are presented in Section 5 for applications based on GSF and in Section 6 for applications based on GGDT alone.

2. GEODESIC DISTANCE TRANSFORMS AND THEIR GENERALIZATION

This section first describes background on geodesic distance transforms and their implementation. It then introduces their generalization using soft masks. Efficient implementation of GGDTs is then briefly discussed.

2.1 Geodesic distance and its computation

2.1.1 *Geodesic distance from a binary region on an image.* Let $I(\mathbf{x}) : \Psi \to \mathbb{R}^d$ be an image ($d = 3$ for a color image), whose support $\Psi \subset \mathbb{R}^2$ is assumed to be continuous for the time being. Given a binary mask M (with $M(\mathbf{x}) \in \{0,1\}$, $\forall \mathbf{x} \in \Psi$) associated to a "seed" region (or "object" region) $\Omega = \{\mathbf{x} \in \Psi : M(\mathbf{x}) = 0\}$, the unsigned geodesic distance transform $D_0(.; M, \nabla I)$ assigns to each pixel $\mathbf{x}$ its geodesic distance from Ω defined as:

$$D_0(\mathbf{x}; M, \nabla I) = \min_{\{\mathbf{x}' | M(\mathbf{x}')=0\}} d(\mathbf{x}, \mathbf{x}'), \quad \text{with} \tag{1}$$

$$d(\mathbf{a}, \mathbf{b}) = \inf_{\mathbf{\Gamma} \in \mathcal{P}_{\mathbf{a},\mathbf{b}}} \int_0^{\ell(\mathbf{\Gamma})} \sqrt{1 + \gamma^2 \left(\nabla I(s) \cdot \mathbf{\Gamma}'(s)\right)^2}\, ds, \tag{2}$$

where $\mathcal{P}_{\mathbf{a},\mathbf{b}}$ is the set of all possible differentiable paths in Ψ between the points $\mathbf{a}$ and $\mathbf{b}$ and $\mathbf{\Gamma}(s) : \mathbb{R} \to \mathbb{R}^2$ indicates one such path, parametrized by its arclength $s \in [0, \ell(\mathbf{\Gamma})]$. The spatial derivative $\mathbf{\Gamma}'(s) = \partial \mathbf{\Gamma}(s)/\partial s$ is the unit vector tangent to the direction of the path. The dot-product in (2) ensures maximum influence for the gradient ∇I when it is parallel to the direction of the path $\mathbf{\Gamma}$. The *geodesic factor* γ weighs the contribution of the image gradient versus the spatial distances. Figure 2 shows an illustration of GDT to a binary mask in a given image.

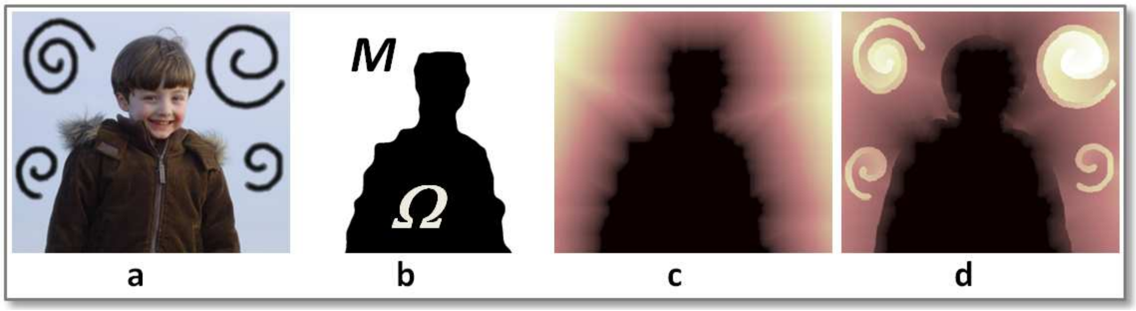


Fig. 2. **Geodesic distances**. **(a)** Original image, I; **(b)** Input mask M with "object" region Ω. **(c)** Euclidean distance from Ω ($D_0(\mathbf{x}; M, \nabla I)$ with $\gamma = 0$ in (2)); **(d)** *Geodesic* distance from Ω ($D_0(\mathbf{x}; M, \nabla I)$ with $\gamma > 0$). Note the large jump in the distance in correspondence with strong image edges.

In case of scalar images, the integral in (2) is the Euclidean length of the 3D path $\tilde{\mathbf{\Gamma}}$ that $\mathbf{\Gamma}$ defines on the $(x, y, \gamma I)$ surface: $\tilde{\mathbf{\Gamma}}(s) = [\mathbf{\Gamma}(s); \gamma I(\mathbf{\Gamma}(s))]$. Hence, $d(\mathbf{a}, \mathbf{b})$ is the length of the shortest path between $\mathbf{a}$ and $\mathbf{b}$ on this surface. Also, for $\gamma = 0$, Eq. 2 reduces to the Euclidean length of path $\mathbf{\Gamma}$.

It is worth noting that the term "geodesic distance" is often used more generally to indicate any weighted distance. In [Bai and Sapiro 2007] for instance, gradients of likelihoods are used as geodesic weights.

2.1.2 *Algorithms to compute GDTs.* Excellent surveys of techniques for computing (non-geodesic) DTs may be found in [Fabbri et al. 2008; Jones et al. 2006]. There, two main kinds of algorithms are described: *raster-scan* and *wave-front propagation*. Raster-scan algorithms are based on kernel operations applied sequentially over the image in multiple passes [Borgefors 1986]. Instead, wave-front algorithms such as Fast Marching Methods (FMM) [Sethian 1999] are based on the iterative propagation of a front with an appropriate velocity field.

Geodesic versions of both kinds of algorithms may be found in [Toivanen 1996] and [Yatziv et al. 2006], respectively. Both the Toivanen and Yatziv algorithms produce approximations to the actual distance and both have optimal complexity $\mathcal{O}(N)$ (with N the number of pixels). However, this does not mean that they are equally fast in practice. In FMM front pixels are expanded according to a priority function. This requires accessing image locations far from each other in memory. The limited memory access bandwidth of modern computers limits the speed of execution of such algorithms much more than their modest computational burden. In contrast, Toivanen's technique (employed here) reads the image memory in *contiguous* blocks, thus minimizing delays due to memory access. As demonstrated later this yields greater execution speed. Multiple-pass raster-scan algorithms can also deal with the difficult spiral-like patterns as shown in fig. 2d. We shall also see that the Toivanen algorithm is trivially extended to the generalized geodesic transform presented next, as opposed to fast marching methods.

2.2 Generalized geodesic distance transform (GGDT)

This section presents the *generalized* GDT introduced in [Criminisi et al. 2008] and discusses its relationship to GDT. Its applications will be discussed in Section 3.

2.2.1 *GDT generalized to a soft mask on an image.* The key difference between the GDT and its generalized variant is the fact that in the latter the input seed map M is more generally a *soft*, real-valued function. Given a map $M(\mathbf{x}) \in [0, 1]$ on the image domain Ψ, the GGDT is defined as follows:

$$D(\mathbf{x}; M, \nabla I) = \min_{\mathbf{x}' \in \Psi} \left(d(\mathbf{x}, \mathbf{x}') + \nu M(\mathbf{x}') \right), \qquad (3)$$

with $d(.)$ as in (2). Mathematically, this is a small change as compared to (1). However, the fact that (3) uses the soft *belief* of a pixel belonging to the object or region of interest means that the latter can be defined probabilistically. The advantage is that in several automatic or semi-automatic applications, extraction of such a probabilistic mask is achieved more economically than having to compute a binary segmentation, while conveying more information.

The parameter ν in (3) establishes the mapping between the beliefs M and the spatial distances. Figure 3 further clarifies these points with an explanatory 2D example. In this example, a soft mask of the object of interest is obtained based on user-entered foreground and background brush strokes (as detailed in Section 4.1). This soft mask is shown in fig. 3b and the corresponding GGDT in fig. 3c. Notice the abrupt distance changes corresponding to the contour of the object of interest (the flower in this case). If the soft mask includes pixels with no uncertainty (where $M(\mathbf{x}) = 0$), it is clear from its definition that the GGDT is equal to zero at these locations and only them (the minimum in (3) is achieved for $\mathbf{x}' = \mathbf{x}$). This is similar to the classic geodesic distance, which vanishes only within the object associated to the binary mask. Any processing based on a decreasing function of the GGDT of soft mask M will have maximal effect at these locations.

Another formal connection between GDT and GGDT can be worked out as follows. In case of binary masks, the expression of the GGDT in (3) for pixels outside the object of interest ($M(\mathbf{x}) = 1$) boils down to

$$\min\{ \min_{\substack{\mathbf{x}' \text{ s.t.} \\ M(\mathbf{x}')=0}} d(\mathbf{x}, \mathbf{x}'), \min_{\substack{\mathbf{x}' \text{ s.t.} \\ M(\mathbf{x}')=1}} d(\mathbf{x}, \mathbf{x}') + \nu \} = \min\{D_0(\mathbf{x}; M, \nabla I), \nu\}. \qquad (4)$$

Hence, in this particular case, the GGDT is simply the thresholded GDT to binary mask defined by M, with threshold ν, the two distances coinciding if the threshold is large enough.

A similar breaking down of the minimization (3) into separate minimizations over sets of constant mask values can be conducted in the general case. Let us denote $\Lambda \subset [0, 1]$ the discrete set of values actually taken by the function M on the discrete image grid

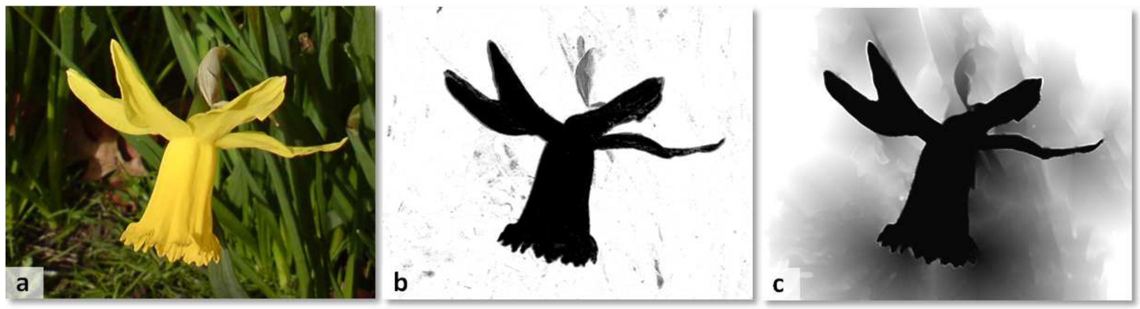


Fig. 3. **Generalized geodesic distances. (a)** An input image. **(b)** A probabilistic (soft) seed mask M (darker for foreground); Intermediate grey values indicate uncertain pixels. **(c)** The estimated Generalized Geodesic Distance (darker for smaller distance values). This is shown here only to provide an idea of what a generalized geodesic distance looks like.

Ψ. Then the GGDT can be rewritten as:

$$D(\mathbf{x}; M, \nabla I) = \min_{\lambda\in\Lambda} \min_{\substack{\mathbf{x}' \text{ s.t.} \\ M(\mathbf{x}')=\lambda}} [d(\mathbf{x},\mathbf{x}')+\nu\lambda] = \min_{\lambda\in\Lambda}[D_0(\mathbf{x}; M_\lambda, \nabla I)+\nu\lambda], \tag{5}$$

where the function M_λ is defined as $M_\lambda(\mathbf{x}') = 0$ if $M(\mathbf{x}') = \lambda$ and 1 otherwise. Hence the GGDT can be deduced from the $|\Lambda|$ GDTs associated to the level sets of M, which is an alternative way to think about it.

So far we have described the mathematical model of the GGDT. Next we describe the algorithm for computing it.

2.2.2 *Efficient computation of GGDTs.* Computation of GDTs and GGDTs requires the discretization of the image domain and of the geodesic distance d in (2). Considering an eight-neighborhood structure on the pixel grid, paths between $\mathbf{a}$ and $\mathbf{b}$ are chains of neighboring pixels ($\mathbf{x}_0 = \mathbf{a}, \mathbf{x}_1, \cdots, \mathbf{x}_{n-1}, \mathbf{x}_n = \mathbf{b}$). Along such a chain the integral in (2) can be approximated with the following sum:

$$\sum_{k=1}^{n} \left[|\mathbf{x}_k - \mathbf{x}_{k-1}|^2 + \gamma^2 |I(\mathbf{x}_k) - I(\mathbf{x}_{k-1})|^2\right]^{\frac{1}{2}}. \tag{6}$$

Based on this discrete definition of the geodesic distance between two neighboring pixels, the GDT for a given binary mask can be computed by wave-front algorithms that propagate a front starting at the boundary of the mask. Such methods can be exact, relying on the minimum cost path Dijkstra's algorithm [Dijkstra 1959], or approximate for lighter computation [Yatziv et al. 2006]. However, computation of GGDT defined in (3) requires, for each pixel $\mathbf{x}$, a minimization with respect to any possible destination $\mathbf{x}'$. Raster-scan methods extends naturally to this case, as follows. After initializing the distance map with scaled seed mask ($D = \nu M$), a first scan is executed from top-left to bottom-right, updating the distance map at the current pixel $\mathbf{x}$ according to

$$D(\mathbf{x}) = \min\left\{D(\mathbf{x}+\mathbf{a}_k) + \left[|\mathbf{a}_k|^2 + \gamma^2 |I(\mathbf{x}) - I(\mathbf{x}+\mathbf{a}_k)|^2\right]^{\frac{1}{2}},\ k = 0\cdots 4\right\}, \tag{7}$$

with $\mathbf{a}_0 = (0,0)$, $\mathbf{a}_1 = (-1,-1)$, $\mathbf{a}_2 = (-1,0)$, $\mathbf{a}_3 = (-1,+1)$ and $\mathbf{a}_4 = (0,-1)$. In the second scan the algorithm proceeds from the bottom-right to the top-left corners using the same udpate rule with opposite displacement vectors $\mathbf{a}_i$'s, to obtain the final map.[2]

This algorithm can be implemented efficiently on multi-core CPU using assembly and SIMD instructions for optimal performance. Note that four of the five elements in the required $\min$ computation (see eqn. 7), are independent. Therefore we compute these using data-level parallelism (SSE3 instruction set).

The same algorithm, up to slight modifications, can also be implemented on the graphics processor. It amounts to extending to soft masks the recent work in [Weber et al. 2008] for approximate computation of GDTs on the GPU (by exploiting NVidia's CUDA new architecture).

3. GSF: GEODESIC SYMMETRIC FILTERING OF IMAGES

The GGDT introduced above can be used as such for a number of edge-sensitive image editing and processing tasks, as we shall demonstrate in Sections 4.2 and 6. It can also serve as the basis of new morphological operators. In this section we introduce the GSF operator as an efficient tool for computing hard and soft bi-layer segmentations.

As discussed in [Fabbri et al. 2008; Jones et al. 2006] DT algorithms are useful for the efficient implementation of morphological operators [Heijmans 1995]. Here we employ geodesic distance transforms to implement efficient *geodesic morphology* [Soille 1999] (see fig. 4), the basis of the GSF filter.

3.1 Geodesic morphology

The definition of the distance D in (3) leads to the following *signed* generalized geodesic distance:

$$D_s(\mathbf{x}; M, \nabla I) = D(\mathbf{x}; M, \nabla I) - D(\mathbf{x}; 1-M, \nabla I). \tag{8}$$

In the case of a binary mask D_s is the signed distance from the object *boundary* (*cf.* fig. 5).

Thresholding D_s at different heights achieves basic morphological operations. For instance, geodesic dilation is obtained as $M_d(\mathbf{x}) = [D_s(\mathbf{x}; M, \nabla I) > \theta_d]$; with $\theta_d > 0$ indicating the diameter of a disk-shaped structuring element. The indicator function [.] returns 1 if the argument is true and 0 otherwise. Similarly, geodesic erosion is obtained as $M_e(\mathbf{x}) = [D_s(\mathbf{x}; M, \nabla I) > -\theta_e]$

[2]Larger kernels (*e.g.*, 5×5) produce better approximations to the exact distance, but with a significant reduction in speed. In our applications multiple distance transforms are mixed together in various ways. This mitigates the effects of inaccuracies in distance computation and lets us get away with smaller kernels. Also, note that 3×3 kernels are used effectively in the vast MRF literature.

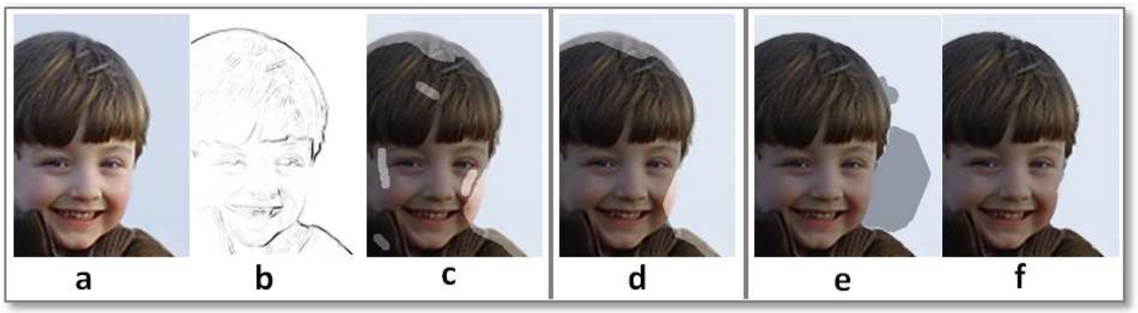


Fig. 4. **Geodesic closing**. **(a)** Original image. **(b)** Image gradient. Notice the weak edge on the cheek. **(c)** Original with *initial* mask superimposed (dark). **(d)** After conventional *closing*. **(e)** After *geodesic dilation*. Notice the leakage effect typical of geodesic transforms in the presence of weak edges. **(f)** After *geodesic closing*. Now the leakage is removed and the mask is aligned with the person's silhouette, more accurately than in (d).

(with $\theta_e > 0$). Finally, geodesic closing and opening are achieved as:

$$M_c(\mathbf{x}) = [D(\mathbf{x}; \overline{M_d}, \nabla I) < \theta_e], \quad (9)$$
$$M_o(\mathbf{x}) = [D(\mathbf{x}; M_e, \nabla I) > \theta_d], \quad (10)$$

respectively; with $\overline{M_d} = 1 - M_d$.

Figure 4 shows an example where an initial binary mask is filtered both by conventional closing (fig. 4d) and then by geodesic closing (fig. 4f). Geodesic filtering encourages the contour of the final mask to follow the object boundary. Notice how possible leakages which may arise from dilation (erosion) in correspondence to weak edges are removed when the opposite erosion (dilation) operation is performed.

Redefining known morphological filters in terms of operations on real-valued distances allows us to: i) implement those operators very efficiently, ii) introduce contrast sensitivity effortlessly, by means of geodesic processing and iii) handle soft masks in the same framework. Next a further modification to conventional morphology is introduced, adding symmetry.

3.2 The GSF operator

Closing and opening are asymmetrical operations in the sense that the final result depends on the order in which the two component operations are applied to the input mask (see also fig. 5d). However, when filtering a signal one would just wish to define the extent of the regions to be removed (*e.g.,* noise speckles) and apply the filter without worrying about the sequentiality of operations within the filter itself.[3]

This problem is solved by GSF filtering. The key idea can be summarized as follows: i) Given the noisy "signal" M, binary or not, we run geodesic dilation and erosion in two parallel tracks; ii) The results are then mixed (by mixing real-valued distances) to produce a distance function which, iii) when thresholded provides the final, spatially smooth segmentation.

The GSF filter is defined mathematically as follows:

$$M_{GSF}(\mathbf{x}; M, \nabla I) = [D^s_s(\mathbf{x}; M, \nabla I) > 0] \quad (11)$$

where the symmetric, signed distance D^s_s is defined as:

$$D^s_s(\mathbf{x}; M, \nabla I) = D(\mathbf{x}; M_e, \nabla I) - D(\mathbf{x}; \overline{M_d}, \nabla I) + \theta_d - \theta_e, \quad (12)$$

[3]Closing-opening filters such as the one used in [Bousseau et al. 2007] may be interpreted as approximations to our symmetrical filter.

with M_e and $\overline{M_d}$ defined earlier. The additional term $\theta_d - \theta_e$ enforces the useful *idempotence* property[4], *i.e.,* it keeps unaltered the remaining signal structure.

Figure 5 illustrates the effect of our GSF filter both on binary masks in 1D and 2D. Notice how isolated peaks and valleys are removed from the original signal while maintaining unaltered the remaining signal structure. The geometric parameters $\boldsymbol{\theta} = (\theta_d, \theta_e)$ establish the maximum size of noise regions to be removed. The effect of varying $\boldsymbol{\theta}$ may be observed also in the 2D example. Alternative, sequential filters such as "erode-dilate-erode" or "open-close" produce less good and (in general) different results from their symmetrical counterparts.

3.3 Parallelism of GSF

The most expensive operation in the GSF filter in (11) is by far the geodesic distance transform. However, note that the four distance transforms necessary to compute M_{GSF} naturally form two pairs of transforms. In each pair the operations are independent of each other. Thus, the transforms in each pair may be computed in parallel on a dual-core processor, as in our implementation. If t is the unit time required for each unsigned GGDT, then the total time T taken to run a GSF filtering operation is $T = 2t$.

Furthermore, as mentionned in Section 2.2.2, each GGDT can be computed approximately but efficiently on the GPU [Weber et al. 2008] (on the latest NVidia devices). The rest of the GSF operations (distance mixing and thresholding) may also be computed easily on modern graphics processors.

To summarize, the GSF operator: i) Generalizes existing morphological operations by adding symmetry and contrast-sensitivity ii) It can be applied to soft masks; ii) It is efficient due to its contiguous memory access and parallelism; iii) It can be implemented on the GPU; iv) Its controlling parameters are geometrically intuitive and easy to set.

Next, we show how to perform many common image and video editing tasks using either GSF operations or using just the generalized GDTs.

4. EFFICIENT EDGE-AWARE EDITING

Most image and video editing operations share the goal of producing a spatially smooth, contrast-sensitive output. For example, image de-noising algorithms tend to smooth out an image

[4]An operator f is idempotent iff $f(f(x)) = f(x)$.

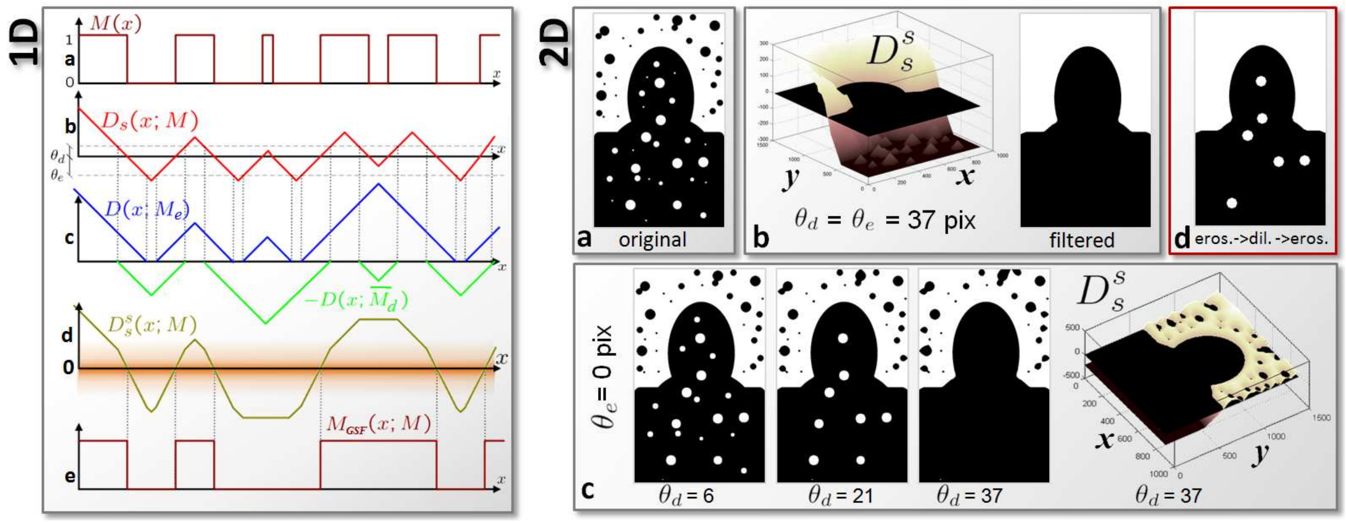


Fig. 5. **Explaining the GSF filter. (Left panel)** *An illustrative example in 1D with binary mask.* **(a)** Input binary signal. **(b)** The signed distance D_s. **(c)** The two further unsigned distances in (12), for selected values of $\boldsymbol{\theta} = (\theta_d, \theta_e)$. **(d)** The final signed distance D_s^s. **(e)** The output, filtered mask $M_{GSF}(x; M)$. The narrower peaks and valleys of $M(x)$ have been removed while maintaining the integrity of the remaining signal. **(Right panel)** *A 2D example.* **(a)** The original noisy binary image. **(b)** Our filtering results for large enough values of the parameters θ_d, θ_e. **(c)** The effect of varying θ_d. For fixed θ_e, more foreground noise is removed as θ_d increases; and vice-versa. **(d)** Existing filters such as "erode→dilate→erode", or "open→close" tend to produce worse results while being affected by the asymmetry problem (see text). For clarity of explanation, no image gradient was used here.

while preserving strong edges. Similarly, image segmentation techniques produce piece-wise flat label maps, with edge-aligned transitions. All these tasks can be efficiently accomplished by using our GGDTs or our GSF operator.

4.1 Image segmentation via Geodesic Symmetric Filtering

Given an image I we wish to select the foreground region ($\texttt{Fg}$) and separate it from the background ($\texttt{Bg}$) as quickly and accurately as possible. The input data is represented as an array of image pixels $\mathbf{z}$, indexed by the pixel position $\mathbf{x}$ as $z(\mathbf{x})$. The corresponding binary, per-pixel segmentation labeling is denoted $\boldsymbol{\alpha}$.

In segmentation algorithms the usual starting point is to define the pixel-wise foreground and background likelihoods $p(z(\mathbf{x})|\alpha(\mathbf{x}) = \texttt{Bg})$ and $p(z(\mathbf{x})|\alpha(\mathbf{x}) = \texttt{Fg})$. These likelihoods may be obtained interactively or automatically from a variety of sources, *e.g.*, from user strokes [Bai and Sapiro 2007; Boykov and Jolly 2001; Rother et al. 2004; Li et al. 2004] [5], from stereo correspondence [Kolmogorov et al. 2005], from comparison with a background model [Criminisi et al. 2006], from object classification [Shotton et al. 2007], etc.

Inferring the segmentation then proceeds by finding the solution $\boldsymbol{\alpha}$ which: i) "obeys" the likelihood, ii) is spatially smooth, and iii) is aligned with strong edges. A popular way to achieve this consists in defining and minimizing a global energy function over the binary label field [Boykov and Jolly 2001]. This function is composed of pixel-wise likelihoods, and pair-wise terms for contrast-sensitive regularization. A modern view of this prototypical approach is to associate the energy function to the posterior distribution of the hidden label field, which is turned this way into a conditional (Markov) random filed, or CRF. The global optimum of the energy is then the maximum a posteriori estimate (MAP) of the hidden field.

In [Criminisi et al. 2008] the GSF filter was used within such a CRF energy minimization framework. However, [Criminisi et al. 2008] showed that the output of *each* GSF operation is itself spatially smooth (for large values of $\boldsymbol{\theta}$). Thus, in a graphics application we can assume that the user has interactively set the geometric parameters $\boldsymbol{\theta}$ [6] and the output segmentation is that achieved directly as the output of the GSF filter. This approach, proposed here, removes the need for energy minimization altogether with considerable reduction of computation times.[7]

In this paper, we show that efficient segmentation, and associated tasks, can be simply achieved by applying the GSF to the *real-valued* log-odds map $M(\mathbf{x}) = \sigma\left(\ln \frac{p(z(\mathbf{x})|\alpha(\mathbf{x})=\texttt{Bg})}{p(z(\mathbf{x})|\alpha(\mathbf{x})=\texttt{Fg})}\right)$, with $\sigma(.)$ the sigmoid transformation $\sigma(t) = 1/(1 + \exp(-t/\mu))$. In all experiments in this paper we have fixed $\mu = 5$. The output segmentation mask M_{GSF} is simply obtained by selecting a value of $\boldsymbol{\theta}$ (*e.g.*, based on the observed spatial extent of noise speckles) and applying the GSF filter in (11) to the input mask M (*cf.* fig. 5).

Spatial smoothness and robustness to noise. Figure 6 illustrates the behaviour of the GSF filter in the presence of weak unaries (i.e.

[5] Bai's technique has been designed solely for *interactive* segmentation.

[6] *e.g.*, typically here $\theta_e = \theta_d = 10pix$, but depends on image resolution.

[7] Avoiding the definition and the minimization of a global CRF energy function might appear as conceptually less appealing. Note however that there is arguably no fundamental or practical superiority to the energy-based approach when the energy is designed in an ad-hoc fashion, as it is often the case. In fact, we shall see in sec.5.1.2 that good segmentation of fine structures can be obtained with our segmentation approach with no energy minimization. Finally, recent studies showed that there are cases where segmentation based on geodesic distances can in fact be related to the global minimization of new energy functions [Sinop and Grady 2007; Couprie et al. 2009].

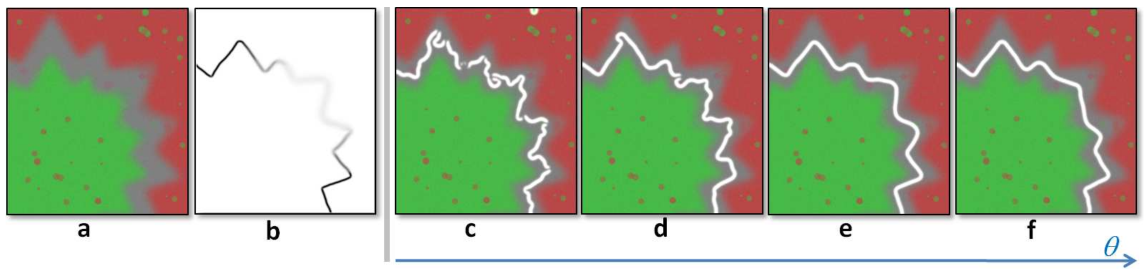


Fig. 6. **Spatial smoothness and robustness to noise**. **(a)** Input soft mask (i.e. pixel-wise likelihoods, green for `Fg` and red for `Bg`). Notice the large amount of noise and the large uncertain region (in grey). **(b)** Magnitude of gradient of input image. This is used in the computation of geodesic distance transforms. **(c-f)** Computed segmentation boundary (white curve) for increasing values of $\theta_d = \theta_e$. Larger values of θ yield smoother segmentation boundaries in the presence of weak edges and/or weak likelihoods; and thus stronger robustness to noise. In contrast, strong gradients "lock" the segmentation in place.

weak input likelihoods) and weak gradients, in a toy example. Despite the lack of a Markov Random Field energy model our GSF operator achieves spatially smooth segmentation. The key is in the geometric parameters $\boldsymbol{\theta}$. As illustrated also in fig. 5, larger values of $\boldsymbol{\theta}$ tend to produce smoother segmentations (less broken up and smoother contours) despite the fact that the GDT itself is not robust to noise. Robustness of the GSF filter comes from combining different distance transforms together. Robustness to bleeding effects is also illustrated in fig. 4.

Note also that the symmetric signed distance $D_s^s(., M, \nabla I)$ does not need to be thresholded and can often be used as a soft segmentation map. An example of such use for re-colorization is illustrated in section 5.3.3. Segmentation results and comparisons are presented in Section 5. Next, we describe application of the generalized geodesic distance to other non-linear image filtering operations.

4.2 Edge-sensitive smoothing via Generalized Geodesic Distances

In this section we propose a method to turn an input image into a piece-wise smooth image, while mantaining sharp transitions at object boundaries. Such edge-preserving image flattening is useful for a number of applications such as denoising, texture flattening and cartooning.

Edge-aware flattening can be achieved for example by anisotropic diffusion [Perona and Malik 1990], or bilateral filtering [Tomasi and Manduchi 1998]. Here instead we follow the approach used in gray-scale morphology, where binary morphology filters are run on each intensity level and the results combined into the final output. In order to achieve edge sensitivity we replace conventional morphology filtering with our *geodesic* transforms applied to small number of *softly* quantized image layers. The processed components are then recombined to produce the flattened output.

In detail, given a color image I and its luma channel Y taking values in the range $\{0, \cdots, \tau - 1\}$, all pixel intensities are quantized into k bins, *e.g.*, using conventional K-means clustering. Each bin or cluster is associated with a mean luma value μ_i and standard deviation σ_i.

At this point, k soft masks $M_i(\mathbf{x})$ are computed as a function of the probability of each pixel belonging to the i^{th} cluster as follows:

$$M_i(\mathbf{x}) = 1 - e^{-\frac{1}{2}\left(\frac{Y(\mathbf{x})-\mu_i}{\sigma_i}\right)^2}. \tag{13}$$

For each mask M_i we then compute the corresponding GGDT, $D_i(\mathbf{x}) = D(\mathbf{x}; M_i, \nabla I)$, as in (3). For each pixel $\mathbf{x}$ and each layer i, a weight measuring the contribution of the intensity μ_i at pixel $\mathbf{x}$ in the final image is computed as

$$W_i(\mathbf{x}) = e^{-\frac{D_i^2(\mathbf{x})}{\phi^2}}, \tag{14}$$

where $\phi > 0$ is a parameter (a value $\phi = 100$ is used here). The flattened luma at pixel $\mathbf{x}$ is finally obtained as a weighted average of cluster intensities:

$$Y'(\mathbf{x}) = \sum_{i=1}^{k} \mu_i W_i(\mathbf{x}) \;/\; \sum_{i=1}^{k} W_i(\mathbf{x}). \tag{15}$$

Combining the flattened luma Y' with the unprocessed chromaticity channels yields the output flattened color image. Note that in this work we have chosen to process only the luma channel for speed. Alternatives where all three RGB colour channels are processed independently may also be considered.

Since the different image levels are processed independently from one another the algorithm is intrinsically parallel and can be easily implemented to run on multiple cores. Furthermore, strong quantization of the input levels (*i.e.*, $k \ll \tau$) can produce artefact-free results at great speed (*cf.* fig. 20).

5. APPLICATION OF GSF-BASED SEGMENTATION AND COMPARISONS

This section demonstrates the use of the geodesic symmetric filter as an efficient and precise segmentation tool. Its performance is assessed through a number of quantitative and qualitative comparisons with state of the art. Note that all experiments presented in this section, unless specified differently, were run on a Core 2 Duo desktop machine with 4GB RAM.

5.1 Interactive segmentation of high-resolution images

In all the examples in this section the pixel likelihood $L(\mathbf{x})$ is computed from user-entered brush strokes as follows. The pixels in the strokes are accumulated into two $32 \times 32 \times 32$ RGB histograms, one for the `Fg` and one for the `Bg` strokes. Then $L(\mathbf{x})$ is estimated as the log of the ratio of those histograms, evaluated at each image pixel $\mathbf{x}$. This simple, non-parametric model avoids inefficient Expectation Maximization (as used in GrabCut [Rother et al. 2004]) while providing good accuracy.

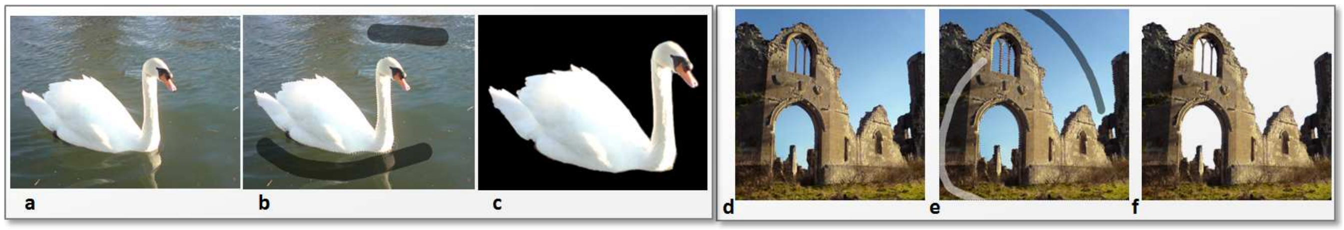
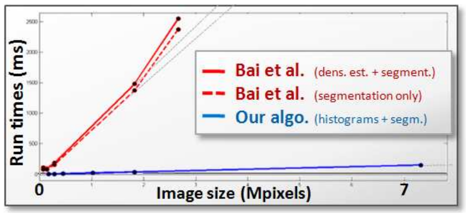


Fig. 7. **Examples of interactive image segmentation**. **(a)** Original photo of a swan (3200×2400 pix. Area $= 7.3$Mpix). **(b)** The user-entered background strokes (dark, superimposed). **(c)** The segmented foreground obtained in $\sim$ 43ms. **(d)** Photo of a derelict castle (4600×4627 pix. Area $= 20.3$Mpix). **(e)** The two `Fg` and `Bg` brush strokes. **(f)** The segmented foreground obtained in $\sim$ 120ms.



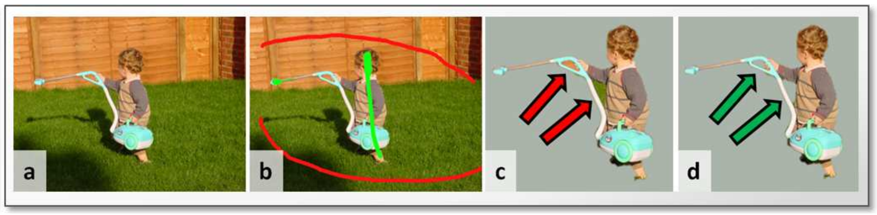


Fig. 8. **Comparisons with Bai et al. (Left panel)** *Run time comparisons* for Bai's segmentation algorithm (red curves) and ours (blue). Grey lines indicate linear fit to the curves. For Bai's algorithm we have used the run-times reported in Table 1 of [Bai and Sapiro 2008]. All curves were obtained on similar spec. machines (2GB RAM, 2GHz CPU). As predicted by their theoretical efficiency, a roughly linear behaviour is shown by the run-times of both algorithms; but the slope of ours is much lower, indicating greater efficiency. **(Right panel)** *Robustness to complex topology.* **(a)** Original. **(b)** User strokes. **(c)** Segmentation results from Bai et al. **(d)** Segmentation results from our algorithm. Identical strokes are used in both cases. In Bai et al. the implicit connectivity prior produces erroneous connected regions which can be removed only with further interaction. Both graph-cut and our algorithm by acting on the pixel likelihoods overcome such problem.

A first example of interactive segmentation on a high-resolution image is shown in fig. 1. The flower image is $\sim 20Mpix$ in size and the segmentation is updated on our machine in only **121 ms** for changes to $\boldsymbol{\theta}$ and **810 ms** for changes to the user strokes. This is to be compared with a graph-cut segmentation time of ∼13.3s.[8] As in [Rother et al. 2004] the color models are updated iteratively (typically 2 iterations suffice) to achieve accurate segmentation with economical interaction. Further results on relatively simple images are shown in fig. 7. More results are in fig. 9 and the accompanying video.[9]

5.1.1 *Comparison with Bai et al..* Our technique takes inspiration from [Bai and Sapiro 2007] and extends that work in many ways. In this section we compare the two approaches in terms of both efficiency and accuracy.

5.1.1.1 *Computational efficiency*. Figure 8(Left) compares the run times achieved by the recent FMM method in [Yatziv et al. 2006] (employed in [Bai and Sapiro 2007]) with those achieved by our technique, as a function of the image size N. The task is that of image segmentation and it was run on similar-spec. machines. As discussed earlier, despite both algorithms being $\mathcal{O}(N)$ in complexity, our technique achieves lower run-times in practice, thanks to its contiguous memory access.

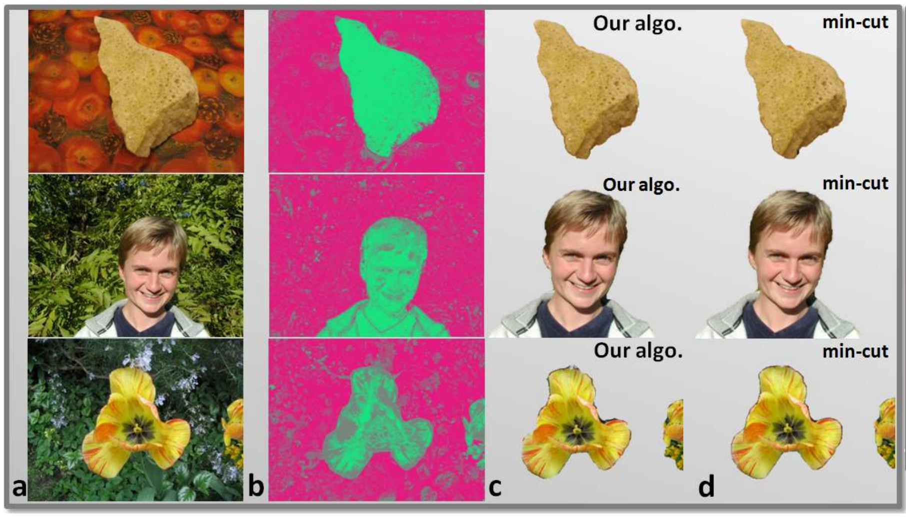
5.1.1.2 *Topological differences.* In Bai's work GDTs are computed from binary, user-entered strokes. This implies that each output, segmented region needs to be connected to at least one such stroke.[10] This effect is due to the algorithm's implicit "connectivity prior", which can often turn out to be useful in practice. In contrast, our GSF operator is applied to real-valued pixel likelihoods. This removes any topological restrictions and extends the applicability of our algorithm to automatic segmentation tasks. Figure 8(Right) illustrates those points by comparing segmentation results on a standard test image. Another example of robustness to complex topology is shown in fig. 7f. Next we compare our results with those obtained by graph-cut based techniques.

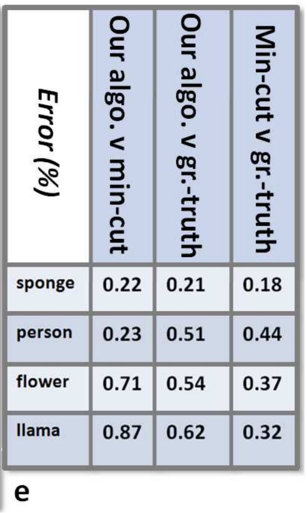
5.1.2 *Comparison with min-cut.* Figure 9 presents a comparison with min-cut [Boykov and Jolly 2001], obtained on standard test images. Our algorithm achieves similar segmentations to min-cut, but about two orders of magnitude faster. Furthermore, while our algorithm's runtimes are linear with the image area (*cf.* fig. 8, blue curve), for min-cut we have observed a slightly super-linear behaviour. Fig. 10 presents another comparison with min-cut, indicating similar (if not better) segmentation quality at a fraction of the computational cost. Here, the unaries were fixed and the parameters optimized (manually) individually for each algorithm. In this specific example our algorithm seems to be more robust to the "shrinking bias" problem.

5.1.3 *Comparison with multi-resolution techniques.* In order to run on high resolution images both graph-cut [Boykov and Jolly 2001] and random walker [Sinop and Grady 2007] require a multi-resolution approach.[11] The unavoidable loss of details is illustrated

[8]With our own implementation of graph-cut, optimized for regular grids and thus about 1/3 faster than publicly available C implementations.
[9]`http://research.microsoft.com/apps/pubs/default.aspx?id=81528`

[10]*e.g.* segmenting the image of a chess board would require at least $8 \times 8 = 64$ user strokes.
[11]In random walker the time-consuming step is the inversion of an $N \times N$ matrix, where N is the number of pixels.





Error (%)	Our algo. v min-cut	Our algo. v gr.-truth	Min-cut v gr.-truth
sponge	0.22	0.21	0.18
person	0.23	0.51	0.44
flower	0.71	0.54	0.37
llama	0.87	0.62	0.32

e

Fig. 9. **Comparison with min-cut. (a)** Input images. The input images *and* the associated input user strokes come from the standard GrabCut dataset which was used for comparisons in [Szeliski et al. 2006]. **(b)** Our segmentation results. **(c)** Min-cut results. **(d)** Segmentation errors, measured as the percentage of differently classified pixels. The segmentation results are extremely similar, with our technique being much faster. The segmentation of the "llama" test image is shown in the accompanying video.

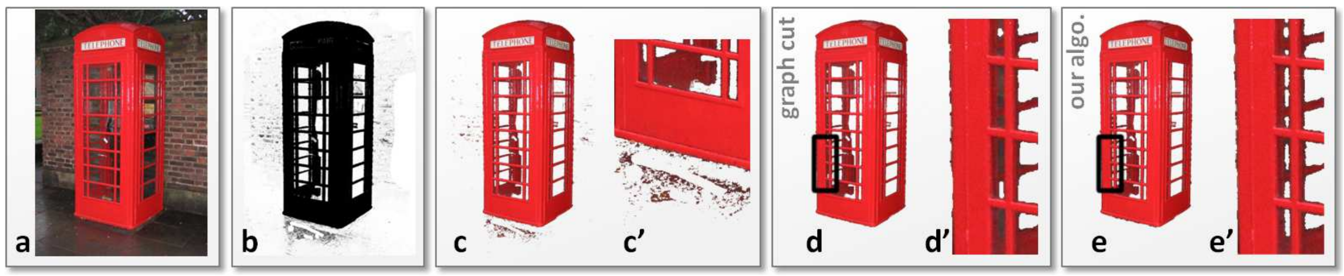


Fig. 10. **Further comparisons with min-cut. (a)** Original image (2048×1536 pix). **(b)** The likelihood signal used for all segmentation results (dark for Fg). **(c)** Simply thresholding the unaries is very fast but produces significant segmentation artefacts. This is due to the lack of spatial smoothness priors. **(c')** Zooming in on those artefacts. **(d, d')** The segmentation produced by min-cut, at full resolution in **1126 ms**. Isolated pixels have disappeared at the price of high computational cost. **(e, e')** The segmentation produced by our geodesic algorithm in **26 ms**. The segmentations in (d) and (e) are similar in quality. Both encourage connected regions separated by strong image edges. In this example, our algorithm is more than 40 times faster than graph-cut.

for graph-cut in fig. 11b. The more recent work in [Grady and Sinop 2008] achieves run-times of the order of seconds on small images, which is much slower than our approach.

5.2 Segmenting n-dimensional data

Geodesic distances are easily defined in an n-dimensional space, with $n > 2$. Hence, our algorithm is not restricted to 2D image data and can easily be extended to n-D data such as videos or medical image datasets (typically 3D or even 4D). In fig. 12 we show an example of bilayer video segmentation where the whole video cube ($709 \times 540 \times 120$ voxels) is segmented at once as opposed to frame-by-frame. Batch video processing of this kind minimzes temporal instability.

5.3 Other segmentation-based applications

5.3.1 *Panoramic image stitching.* As a further application of our technique we present results on panoramic image stitching. This task can be interpreted as a version of segmentation. In fact, given two registered, overlapping images of a scene, we wish to find the cut through the stitching map such that the two output regions are smooth and their interface aligned with strong image edges.

In this case we define the log-likelihood map L as $L \in \{-\infty, 0, \infty\}$ with $L = 0$ in the overlapping region. Then, our GSF operator encourages the separating cut to lie in the overlap region and follow strong image edges. Figure 13 shows stitching results on two 1650×1500 pre-registered photos of Rome's Piazza Navona. The parameters $\boldsymbol{\theta}$ and γ where set interactively. High quality stitching results are achieved with our algorithm in only $\sim 10ms$ on 2 cores and $\sim 4ms$ on 4 cores.

5.3.2 *Applications in Computer Aided Diagnosis.* As illustrated in fig. 14 our technique may be used to segment anatomical structures within 3D medical images. In the figure the patient's skull has been segmented in 3D from the noisy input CT slices with a few brush strokes. More strokes are then used to segment the teeth which were then highlighted with a different colour and opacity. Figure 15 shows another example, where the patient's aorta has

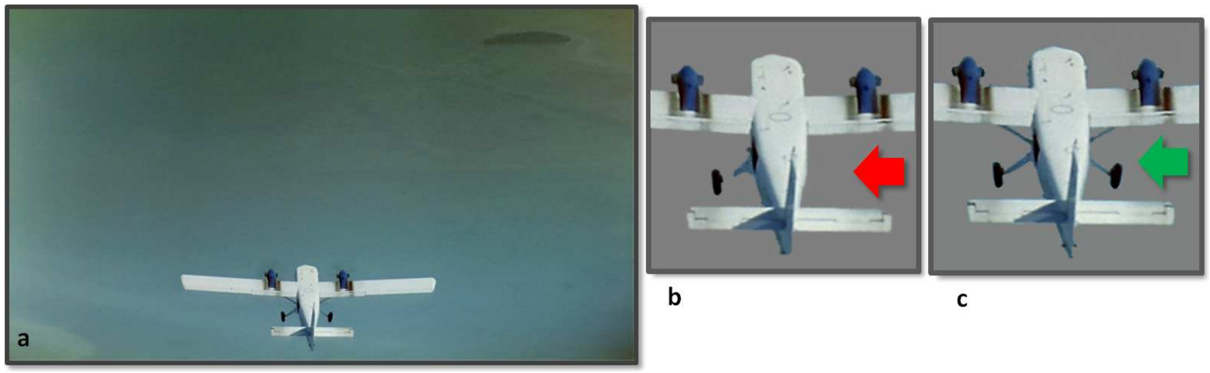


Fig. 11. **Retaining fine details. (a)** A 13Mpix image of an aeroplane. **(b)** Min-cut segmentation. **(c)** Segmentation obtained by our algorithm. The multi-resolution approach necessary for min-cut misses the thin structures of the aeroplane. This is in contrast to our algorithm which runs on the original resolution.

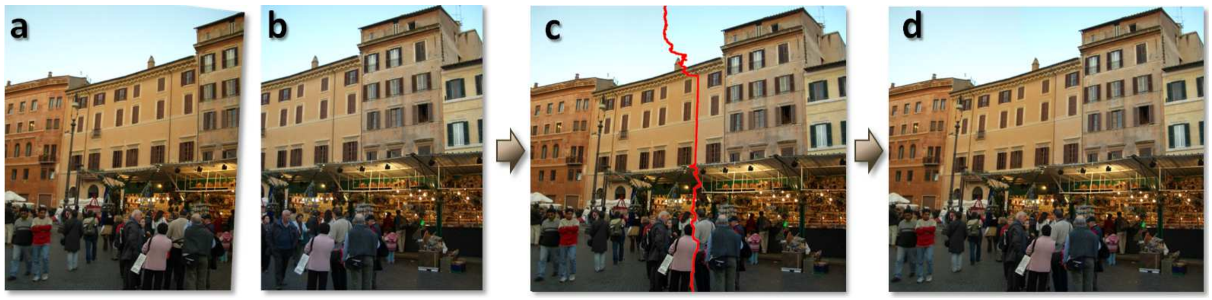


Fig. 13. **Panoramic image stitching. (a,b)** Two registered 1650×1500 images from a rotating camera with people moving between shots. **(c,d)** The seamless stitched panorama obtained by our algorithm in only $\sim 10ms$. **(c)** ...with separating cut superimposed in red.

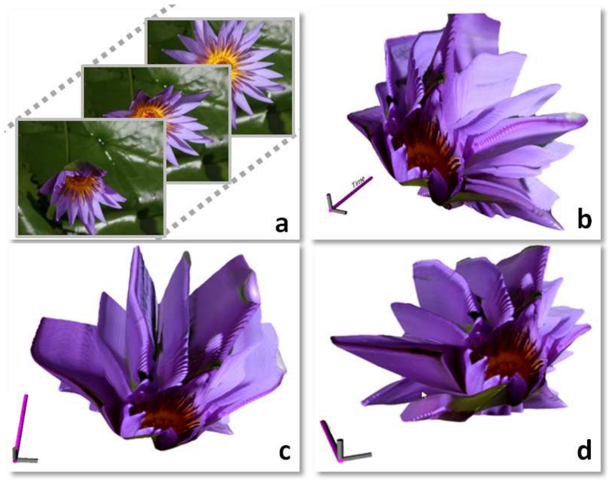


Fig. 12. **Batch segmentation of video**. **(a)** A few frames from a time-lapse video of a growing flower. **(b,c,d)** 3D snapshots from the segmented video. Segmentation was performed directly in the 3D space-time volume. Only two user brush strokes were sufficient.

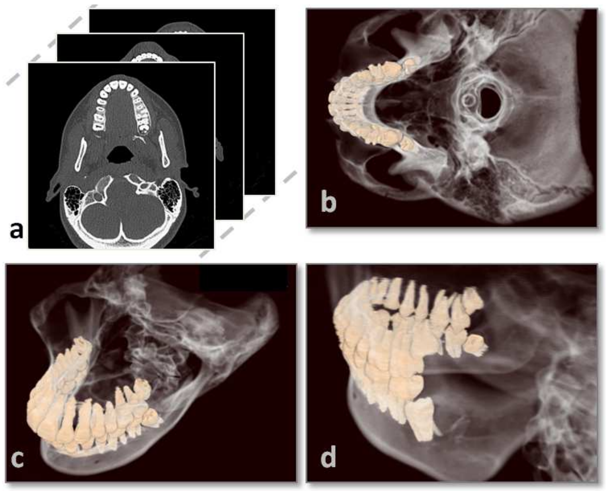


Fig. 14. **Segmentation of anatomical structures in 3D medical images**. **(a)** Input noisy Computed Tomography images of a patient's head. **(b,c,d,)** Segmented and colorized 3D rendered views. The teeth region and the rest of the skull are assigned different colours and opacities to aid medical diagnosis.

been accurately segmented and highlighted. Notice the thin vessels connecting the main artery to the vertebrae.

5.3.3 *Colorization via soft segmentation.* Soft segmentation of an object is also useful for a number of image editing tasks, such as cutout with transparency (matting) or color and tone adjustments. To this end, specific techniques such as border matting [Rother et al. 2004] or geodesic matting [Bai and Sapiro 2007] can be ap-

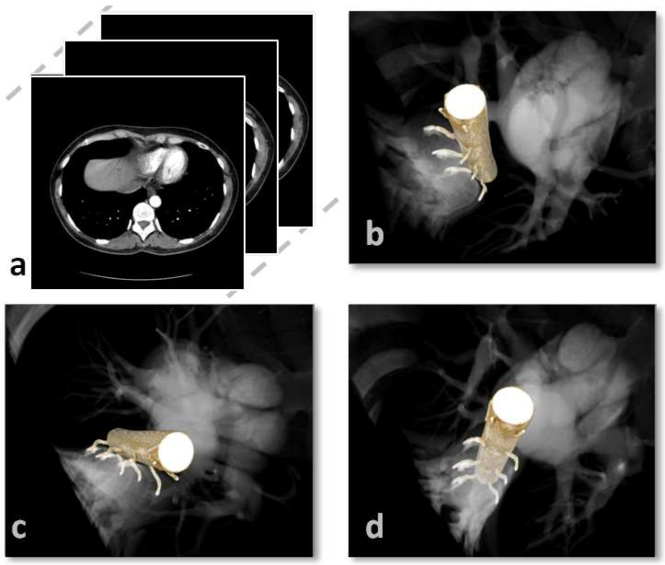


Fig. 15. **Segmentation of anatomical structures in 3D medical images**. **(a)** Input noisy Computed Tomography images of a patient's thorax. **(b,c,d,)** Segmented and colorized 3D rendered views. The aorta has been segmented and highlighted. The opacity of all remaining organs has been reduced though not set to zero (to provide visual context). Notice the thin vessels connecting the aorta to the spine.

plied. Alternatively, the symmetric signed distance $D_s^s(., M, \nabla I)$ to a soft or binary mask M can also be used, with no thresholding, as a soft segmentation map. Figure 16 illustrates an example where: 1) the user has drawn a couple of foreground and background brushes (not shown), 2) foreground and background pixel-wise likelihoods have been estimated, 3) the soft segmentation map D_s^s has been computed (see eq. 12) and 4) used to weight the amount of per-pixel colorization where the target colour (yellow in this case) has been manually selected. Notice that this is a very different task than the one in [Levin et al. 2004]. Here re-colouring is achieved as a soft segmentation task. Also, in contrast to [Yatziv and Sapiro 2006] here the user did not need to touch every petal in order to get a convincing colorization. These results are achieved with only 4 GGDT computations. Finally the fact that the GSF filter imposes spatial smoothness helps avoid colour bleeding effects (*cf.* section 4.1. Larger values of $\boldsymbol{\theta}$ produce larger robustness to noise).

5.4 Limitations

Like all segmentation algorithms the quality of the results depends on how diverse the foreground and background appearance statistics are. When the two layers look similar, like in the case of camouflage, more user interaction is required to obtain a good segmentation (*e.g.*, see the challenging "llama" image in the accompanying video). As shown in fig. 6, unlike [Bai and Sapiro 2007] our technique does encourage smoothness. At this stage, however, it is not clear whether imposing smoothness via GSF filtering is sufficient in general, and when/if a full MRF energy model is more appropriate.

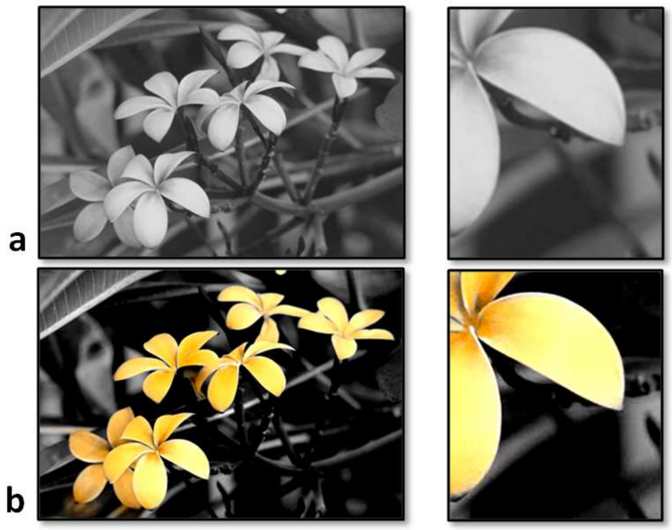


Fig. 16. **Foreground colorization**. (a) Original image; (b) Colorized output. Only a couple of user brush strokes were sufficient to change the colours of all flowers.

6. APPLICATIONS OF GGDT-BASED FLATTENING WITH COMPARISONS

We show in this section how denoising, flattening and cartooning/abstraction applications can be effectively addressed using the same edge-preserving smoothing engine based on Generalized GDT. Different tasks are characterized by small variations in terms of the parameters used and the additional presence of boundary strokes in the case of image tooning.

6.1 Image denoising

Accurate and efficient denoising algorithms find wide application for example when dealing with camera-phone videos, old video footage, ultrasound scans and astronomical imaging.

Image denoising is implemented here via the edge-sensitive smoothing algorithm described in Section 4.2. Figure 17 shows quantitative comparison with respect to the following state of the art algorithms: i) "Non-local Means"' (`nlm`) [Buades et al. 2005], ii) "Fields of Experts" (`foe`) [Roth and Black 2005], and iii) the "Basis Rotation Algorithm" (`brfoe`) [Weiss and Freeman 2007]. The figure shows PSNR curves computed for all algorithms applied to the standard test "peppers" image in fig. 1, for varying levels of added Gaussian noise. The `nlm`, `foe` and `brfoe` results were obtained by using publicly available matlab implementations from the original authors. The parameters of each algorithm were optimized for each algorithm to achieve the highest possible PSNR, and then kept fixed for all test images.

Our algorithm's PSNR curve extensively overlaps the best performing technique (`nlm`) and is better than the fields of experts-based algorithms. Furthermore, avoiding the patch search necessary in `nlm` ensures lower running times (in the order of hundreds of ms). Extensive tests have been run on seven other standard test images (*e.g.*, "Lena", "House", "Boats" etc.) with similar results. In this case $k = \tau$ and $\phi = 10$ were used. At this point we would like to remind the reader that in a parallel implementation each intensity level may be processed by a separate thread/core. In a machine with k cores denoising an image will take the time of flattening a single

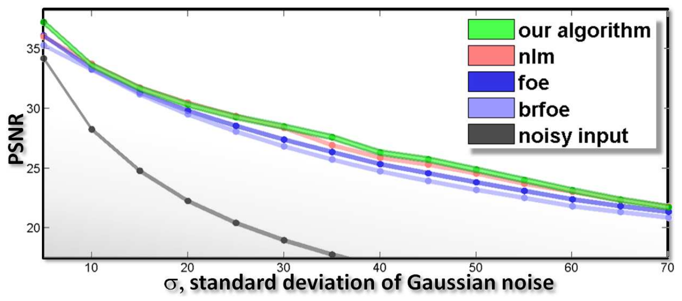


Fig. 17. **Image de-noising comparisons**. Signal-to-noise curves as a function of noise level for four different algorithms. The proposed algorithm (in green) achieves accuracy similar to the best performing, state-of-the-art technique (non-local means); and in some cases it achieves better PSNR. See text for details.

intensity level. A more thorough analysis of denoising properties goes beyond the scope of this paper.

6.2 Texture flattening

Detail-preserving texture flattening is achieved with exactly the same algorithm as the one used for denoising, but with typically larger values of ϕ in (14) (*e.g.*, $\phi = 100$) to allow a larger basin of influence of each pixel and thus a longer range "diffusion" effect. For this kind of abstraction results a small number (*e.g.*, $k = 16$) of intensity clusters suffices. An example is shown in fig. 1. Notice how fine details in the lady's face are preserved (*e.g.*, the ear rings), while the skin texture is effectively smoothed. The effect is visually related to that of bilateral filtering [Chen et al. 2007; Tomasi and Manduchi 1998], but is achieved in real-time on large images with our CPU-based, parallel algorithm. Also, we have observed that the quality of the final output seems slightly better that the one obtained by bilateral filtering. As an example fig. 18 shows a comparison between the flattening results obtained with our approach and those obtained via bilateral filtering.[12] With bilateral filtering it is often difficult to select a value of the range variance σ_r which simultaneously produces enough flattening while avoiding blurring of important details.

6.3 Image and video tooning

To apply a cartoon effect to an image, we first perform edge-preserving texture flattening. We then overlay ink strokes using a mask computed as the contrast-enhanced gradient magnitude of the flattened image, similar to [Winnemoller et al. 2006]. Computing the gradient map on the flattened image rather than the original one ensures longer, visually pleasing strokes. Example tooning results are shown in fig. 19.

The video tooning work in [Wang et al. 2004] used a mean-shift based approach to segment the video into flat regions. However, such hard segmentation technique is likely to produce large temporal instability visible as disturbing flicker. Expensive application of mean-shift to spatio-temporal volumes reduces this effect. In contrast, our technique avoids hard commitment and retains smoothly-varying gradients where necessary, thus reducing temporal flicker. However, large amounts of input noise will still introduce flicker artefacts. This is a problem for old videotape footage and less for

[12]We used the implementation of [Chen et al. 2007] publically available at `http://people.csail.mit.edu/sparis/bf/`

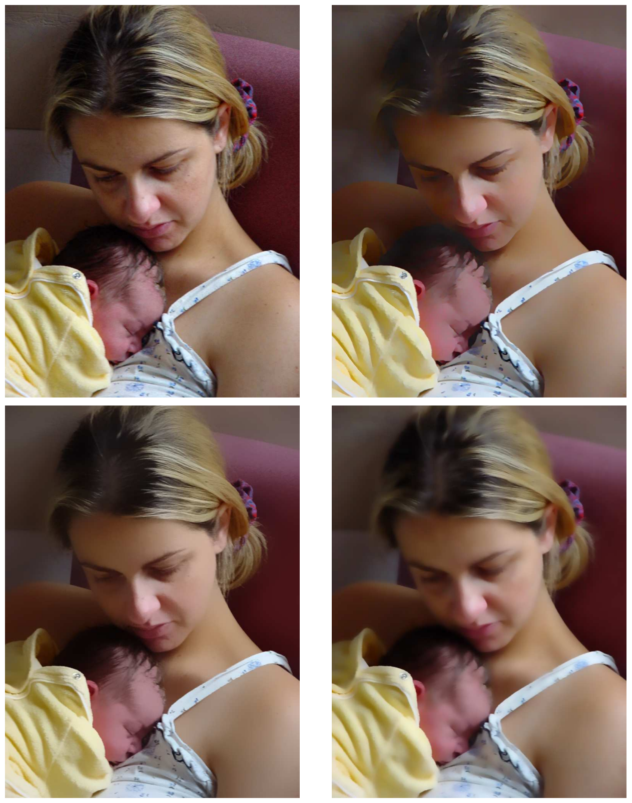


Fig. 18. **Comparing our geodesic-based texture flattening with bilateral filter.** Figure best viewed on screen. **(a)** Original image. **(b)** Geodesic flattening results. Strong flattening of the skin is achieved without blurring facial details such as the eye region. **(c)** Bilateral filter results, with $\sigma_s = 5$ and $\sigma_r = 0.1$. The flattening effect is similar to the one in (b) on parts of the image but much less noticeable on other parts, such as the face of the mum. **(d)** Increasing the variance of the bilateral filter in the range domain (to $\sigma_r = 0.2$) to try and flatten the skin texture results in over-smoothing important facial details.

modern digital cameras. This problem could be corrected by increasing the number of intensity levels but at the cost of lower efficiency. The accompanying video demonstrates the quality of our video tooning results.

6.4 Further discussion of GGDT-based flattening

6.4.1 *Robustness to quantization.* Figure 20 demonstrates the robustness of our flattening algorithm to quantization of the input image levels. Strong quantization ratios (small k/τ) can effectively increase computational efficiency without affecting the visual quality significantly. In fact, in the algorithm described in section 4.2 the use of "soft" intensity quantization and weighted reconstruction (15) allows us to minimize banding artefacts, typical of simpler quantization techniques. A value of k between $k = 8$ and $k = 32$ (tested on many images with diverse colour palettes) represents a good operating setting for most images.

6.4.2 *More on computational efficiency.* On our test machine, the time taken to run a single GGDT on a VGA-sized image (640X480) is $t = 0.9ms$. Then the whole smoothing algorithm takes $T \approx tk/N_c$ ms for a gray image (or a single, luma channel); with k the number of image levels and N_c the number of CPU

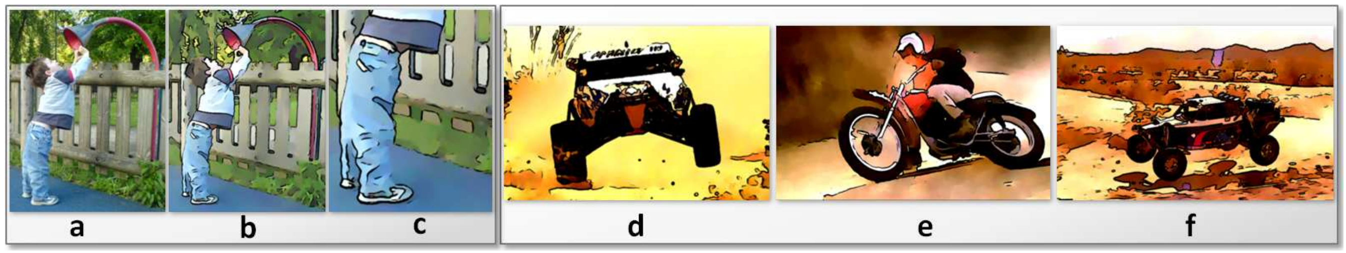
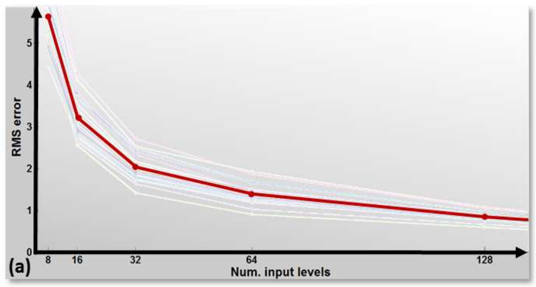


Fig. 19. **Examples of image and video tooning**. **(Left panel)** *Image tooning.* **(a)** Original photo. **(b)** Tooned results. **(c)** Enlarged version of (b), to show details. **(Right panel)** *Video tooning.* From the movie "Dust to Glory". **(d,e,f)** Different frames from the cartooned video. Please see the submitted video to appreciate temporal consistency.



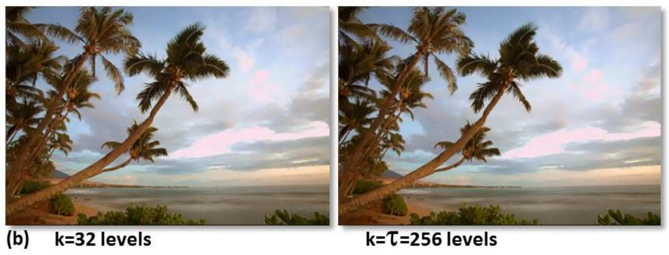


Fig. 20. **Robustness to quantization of input image levels**. **(a)** Twenty different test photographs have been flattened for varying values of $k \in \{8, 16, 32, 64, 128, 256\}$. The difference with respect to the $k = 256$ images have been computed as RMS errors and plotted in different (bright) colors. The mean RMS curve is plotted in red. As expected larger values of k produce lower errors but at a computational cost. **(b)** Two flattened version of a test image obtained for different values of k. There is no perceivable visual difference between the two images, this is often true even for smaller values of k. Achieving good flattening with such hard intensity quantization yields great advantages in terms of speed.

cores.[13] The following table shows the time T as a function of k with $N_c = 4$ cores:

k	8	16	32	64	128	256
T (ms)	1.8	3.6	7.2	14.4	28.8	57.6

To our knowledge ours is the fastest flattening algorithm which does not necessitate resolution reduction, while avoiding banding artefacts.

6.4.3 *Comparison with state of the art.* The work in [Durand and Dorsey 2002; Paris and Durand 2009] has addressed very well the issue of accelerating the bilateral filter. This is achieved in [Paris and Durand 2009] by subsampling *both* in the spatial and intensity domain. The best performance that they report is 38ms on a VGA-sized image. While avoiding spatial subsampling, which is important for preserving thin structures (see fig. 11), we can process an image of the same size in 28ms on a single core CPU (with $N_c = 1$ and $k = 32$) and much quicker if more cores are available. We have also implemented the core GGDT on the GPU following [Weber et al. 2008]. In this case the same texture flattening operation takes less than 4ms. Hence, relying on our generic geodesic machinery (which has been demonstrated on many other tasks) we perform texture flattening faster than in [Paris and Durand 2009] and obtain results which tend to better preserve fine details (see fig. 18).

The best-performing algorithms for median and bilateral filtering have complexity $\mathcal{O}(N \log r)$ [Weiss 2006], with r the radius of the spatial filter. The complexity of our algorithm is linear in the number of pixels and *independent* of the spatial extent of the filter.

The intensity clustering employed in this work is related to channel smoothing [Felsberg et al. 2006]. In [Felsberg et al. 2006] the authors employ quadratic B-Splines for the smoothing of the individual channels. The authors also compare their technique to non-linear diffusion, bilateral filtering and mean-shift. In contrast, here we achieve channel smoothing by means of efficient geodesic transforms.

6.5 Limitations

Exploiting the full potential of our flattening algorithms requires some experience in parallel programming. For instance, each thread may be assigned the task of flattening a single channel. This is a little more involved than writing a single-threaded program but with great efficiency benefits. Finally, reducing the number of levels k too much (*e.g.*, $k = 4$) may result in unsatisfactory results.

7. CONCLUSION

The main contribution of this paper is in having presented a single, efficient algorithm for handling a variety of image and video editing tasks: n-dimensional segmentation, edge-aware denoising and flattening, cartooning, soft selection and panoramic stitching.

At the core is the fast generalized geodesic distance (GGDT) and the geodesic and symmetric operator (GSF) which is built upon it. The algorithm's contiguous memory access and parallelism account for its high efficiency. In turn, this enables very high-resolution images to be processed at interactive rates on the CPU or on the GPU without the need for spatial subsampling.

Quantitative and qualitative experiments on high resolution images and videos, and comparisons with state of the art algorithms have demonstrated the validity of the proposed framework. We believe our technique can be extended to address further tasks such

[13]Our algorithm benefits straight away from architectures with $N_c >> 2$ cores.

as tone mapping and compression, and can have a high impact on future image and video editing applications.

REFERENCES

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUKER, A., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. In *ACM SIGGRAPH*.

BAI, X. AND SAPIRO, G. 2007. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE Intl. Conference on Computer Vision*.

BORGEFORS, G. 1986. Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*.

BOUSSEAU, A., NEYRET, F., THOLLOT, J., AND SALESIN, D. 2007. Video watercolorization using bidirectional texture advection. In *ACM SIGGRAPH*.

BOYKOV, J. AND JOLLY, M.-P. 2001. Interactive graph cuts for optimal boundary and region segmentation of objects in n-D images. In *IEEE Intl. Conference on Computer Vision*.

BROWN, M., SZELISKI, R., AND WINDER, S. 2005. Multi-image matching using multi-scale oriented patches. In *IEEE Computer Vision and Pattern Recognition*.

BUADES, A., COLL, B., AND MOREL, J.-M. 2005. A non-local algorithm for image denoising. In *IEEE Computer Vision and Pattern Recognition*.

CHEN, J., PARIS, J., AND DURAND, F. 2007. Real-time edge-aware image processing with the bilateral grid. In *ACM SIGGRAPH*.

COUPRIE, C., GRADY, L. AMD NAJMAN, L., AND TALBOT, H. 2009. Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In *IEEE Intl. Conference on Computer Vision*. Kyoto, Japan.

CRIMINISI, A., CROSS, G., BLAKE, A., AND KOLMOGOROV, V. 2006. Bilayer segmentation of live video. In *IEEE Computer Vision and Pattern Recognition*.

CRIMINISI, A., SHARP, T., AND BLAKE, A. 2008. GeoS: Geodesic image segmentation. In *European Conference on Computer Vision*. Marseille, France.

DIJKSTRA, E. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik 1,* 269–271.

DURAND, F. AND DORSEY, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. In *ACM SIGGRAPH*.

FABBRI, R., COSTA, L., TORRELLI, J., AND BRUNO, O. 2008. 2D euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys 40,* 1.

FELSBERG, M., FORSSEN, P.-E., AND SCHARR, H. 2006. Efficient robust smoothing of low-level signal features. *IEEE Trans. Pattern Analysis and Machine Intelligence*.

FELZENSZWALB, P. AND HUTTENLOCHER, D. P. 2004. Efficient belief propagation for early vision. *Int. Journal of Computer Vision*.

GRADY, L. 2006. Random walks for image segmentation. *IEEE Trans. PAMI 28,* 11.

GRADY, L. AND SINOP, A. K. 2008. Fast approximate random walker segmentation using eigenvector precomputation. In *IEEE Computer Vision and Pattern Recognition*.

HEIJMANS, H. J. A. M. 1995. Mathematical morphology: A modern approach in image processing based on algebra and geometry. *SIAM Review 37,* 1, 1–36.

JONES, M., BAERENTZEN, J., AND SRAMEK, M. 2006. 3D distance fields: a survey of techniques and applications. *IEEE Trans. on Visualization and Computer Graphics 12*.

JUAN, O. AND BOYKOV, J. 2006. Active graph cuts. In *IEEE Computer Vision and Pattern Recognition*.

KOHLI, P. AND TORR, P. H. S. 2007. Dynamic graph cuts for efficient inference in Markov Random Fields. *IEEE Trans. Pattern Analysis and Machine Intelligence*.

KOLMOGOROV, V., CRIMINISI, A., BLAKE, A., CROSS, G., AND ROTHER, C. 2005. Bilayer segmentation of binocular stereo video. In *IEEE Computer Vision and Pattern Recognition*.

KOLMOGOROV, V. AND ZABIH, R. 2004. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Analysis and Machine Intelligence 26,* 2.

KOPF, J., , COHEN, M., LISCHINSKI, D., AND UYTTENDAELE, M. 2007. Joint bilateral upsampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26,* 3.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2004. Colorization using optimization. *ACM Trans. on Graphics*.

LI, Y., SUN, J., TANG, C.-K., AND H.-Y., S. 2004. Lazy snapping. *ACM Trans. on Graphics*.

LISCHINSKI, D., FARBMAN, Z., UYTTENDAELE, M., AND SZELISKI, R. 2006. Interactive local adjustment of tonal values. *ACM Trans. on Graphics*.

LIU, J., SUN, J., AND H.-Y., S. 2009. Paint selection. *ACM Trans. on Graphics*.

LOMBAERT, H., SUN, Y., GRADY, L., AND XU, C. 2005. A multilevel banded graph cuts method for fast image segmentation. In *IEEE Intl. Conference on Computer Vision*. Bejing, China.

LUAN, Q., WEN, F., COHEN-OR, D., LIANG, L., XU, Y. Q., AND SHUM, H. Y. 2007. Natural image colorization. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*, J. Kautz and S. Pattanaik, Eds. Eurographics.

PARIS, S. AND DURAND, F. 2009. A fast approximation of the bilateral filter. *Intl. Journal of Computer Vision*.

PERONA, P. AND MALIK, J. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Analysis and Machine Intelligence 12,* 7.

ROTH, S. AND BLACK, M. 2005. Fields of experts: A framework for learning image priors. In *IEEE Computer Vision and Pattern Recognition*.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. GrabCut: Interactive foreground extraction using iterated graph cuts. In *ACM Trans. on Graphics (SIGGRAPH)*.

SETHIAN, J. A. 1999. Fast marching methods. *SIAM Rev. 41,* 2.

SHOTTON, J., WINN, J., ROTHER, C., AND CRIMINISI, A. 2007. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling appearance, shape and context. *Intl. Journal on Computer Vision*.

SINOP, A. AND GRADY, L. 2007. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *IEEE Intl. Conference on Computer Vision*. Rio de Janeiro, Brazil.

SOILLE, P. 1999. *Morphological Image Analysis*. Springer.

SZELISKI, R. 2006. Locally adapted hierarchical basis preconditioning. *ACM Trans. on Graphics*.

SZELISKI, R., ZABIH, R., SCHARSTEIN, D., VEKSLER, O., KOLMOGOROV, V., AGARWALA, A., TAPPEN, M., AND ROTHER, C. 2007. A comparative study of energy minimization methods for Markov Random Fields with smoothness-based priors. In *Intl. Journal of Computer Vision*.

TOIVANEN, P. J. 1996. New geodesic distance transforms for gray-scale images. *Pattern Recognition Letters 17,* 5, 437–450.

TOMASI, C. AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *IEEE Intl. Conference on Computer Vision*. 839–846.

WANG, J., BHAT, P., COLBURN, R. A., AGRAWALA, M., AND COHEN, M. F. 2005. Interactive video cut out. *ACM Trans. on Graphics*.

WANG, J., XU, Y., SHUM, H.-Y., AND COHEN, M. 2004. Video tooning. In *ACM SIGGRAPH*.

WEBER, O., DEVIR, Y. S., BRONSTEIN, A. M., BRONSTEIN, M. M., AND KIMMEL, R. 2008. Parallel algorithms for approximation of distance maps on parametric surfaces. In *ACM SIGGRAPH*.

WEISS, B. 2006. Fast median and bilateral filtering. In *ACM SIGGRAPH*.

WEISS, Y. AND FREEMAN, W. T. 2007. What makes a good model of natural images? In *IEEE Computer Vision and Pattern Recognition*.

WINNEMOLLER, H., OLSEN, S. C., AND GOOCH, B. 2006. Real time video abstraction. In *ACM SIGGRAPH*.

YATZIV, L., BARTESAGHI, A., AND SAPIRO, G. 2006. O(n) implementation of the fast marching algorithm. *Journal of Computational Physics 212*, 393–399.

YATZIV, L. AND SAPIRO, G. 2006. Fast image and video colorization using chrominance blending. *IEEE Trans. on Image Processing 15*, 5.

Received ; accepted