Geodesic Forests for Image Editing
A. Criminisi†, T. Sharp†, P. Pérez‡
† Microsoft Research Ltd., Cambridge, UK
‡ Technicolor Research and Innovation, Cesson-Sévigné, France

Abstract
A Geodesic Forest is a new representation of digital color images which yields flexible and efficient editing algorithms.

In this paper an image is decomposed into a collection of trees (a forest) whose branches follow directions of minimum variation. This representation enables expensive, 2D, edge-aware processing to be cast as efficient one-dimensional operations along the tree branches. Existing and novel contrast-sensitive editing tasks can now be achieved by simple and effective algorithms acting on the same tree-based image decomposition.

The contribution of this paper is three-fold: i) We introduce the Geodesic Forests image representation which unifies a number of previously diverse editing techniques; ii) We present a GPU-CUDA algorithm for the efficient decomposition of an image into a complete set of disjoint geodesic trees; iii) We describe a number of simple algorithms to generate existing and new edge-aware image and video effects.

The effectiveness of our algorithms is demonstrated with a number of applications such as: texture flattening, ink painting, data-aware resizing, diffusive painting and geodesic plotting. The high level of parallelism of our algorithms enables them to be applied interactively to high-resolution images (≈ 15Mpixel), and video data.


Keywords: geodesic distance transform, parallel distance transform, forest transform, image editing, bilateral filtering, texture flattening, image resizing, non-photorealistic rendering.

1 Introduction
This paper introduces a new representation of color images which turns expensive, edge-aware, 2D processing into a set of efficient 1D operations. Existing and new image effects are enabled by such unified representation.

Nowadays high resolution images and videos are ubiquitous. Yet, algorithms for editing such data are not fast enough to allow real-time processing; the reason being that many modern editing algorithms require spatially variant processing. For instance, one of the most sought-after requirements is edge-sensitivity [Chen et al. 2007; Rother et al. 2004]; i.e. the ability of the algorithm to change its behavior depending on the local image contrast. Such dependence on the data content limits the speed of current algorithms.

This paper shifts the burden of spatial inhomogeneity from each specific algorithm to the underlying image representation. This is achieved by reorganizing the image content so as to hierarchically cluster together pixels which are similar to one another and spatially close to one another. Once such a decomposition is com-
puted, many diverse editing tasks may be applied directly to the 1D neighborhoods defined by the branches of our tree-based structure.

GPU processing and the intrinsic parallelism of tree-based processing allow a number of very diverse effects to be achieved efficiently as variants of the same underlying algorithm.

1.1 Literature survey

This work is inspired by and builds upon the following four papers [Falcao et al. 2004; Chen et al. 2007; Criminisi et al. 2008; Weber et al. 2008].

Geodesic Forests are related to the Image Foresting Transform (IFT) [Falcao et al. 2004], with the following differences: i) Geodesic Forests use probabilistic soft seed masks as opposed to binary seed masks. This captures data uncertainty and reduces computation (details later). ii) Here the seed masks and the tree roots are computed automatically as well as selected manually. iii) The Geodesic Forest decomposition is achieved via an efficient GPU-based, linear-time algorithm. iv) We extend the range of editing applications from morphological-like operations [Falcao et al. 2004] to: texture flattening, diffusive painting, seam carving, ink painting and many others.

The work in [Chen et al. 2007] has introduced the “Bilateral Grid”; i.e. a clever data structures which combines spatial and range dimensions into a single, coarse resolution 3D array. This structure enables efficient edge-aware GPU-based editing. Geodesic Forests are a very different representation which turns an n-dimensional regular grid into a set of 1D paths and enables GPU processing without compromising resolution or image detail. Our representation relates to physical phenomena such as diffusion and fluid dynamics and provides new insights into geodesic distance transforms and their uses.

The work in [Criminisi et al. 2008] has introduced generalized geodesic distances (GGDT), i.e. geodesic distances applied to real-valued, probabilistic seed masks and demonstrated their application in segmentation. GGDTs are the primary tool used in our tree-based image decomposition algorithm.

The Geodesic Forest representation builds upon the vast literature on efficient geodesic distance transforms. A complete list of relevant papers is beyond the scope of this paper; with some of the most relevant work being: [Sethian 1999] for the Fast Marching Method; [Tsitsiklis 1995] for comparisons between wave-front and raster-scan techniques; [Breu. et al. 1995; Fischer and Gotsman 2006] for the efficient computation of Voronoi diagrams; [Surazhsky et al. 2005; Sigg et al. 2003; Sud et al. ] for CPU and GPU algorithms applied to triangle meshes; and finally [Weber et al. 2008] for an efficient GDT algorithm.

Currently, the fastest reported GDT algorithm is the GPU-CUDA implementation in [Weber et al. 2008]. In this paper we describe two modifications of the Weber algorithm: i) we adapt it to compute generalized geodesic distances and, ii) we store the minimum cost paths as well as the distances, with little loss of efficiency.

The need for edge-aware processing. Edge-aware segmentation has been achieved recently using a variety of image models such as Markov and Conditional Random Fields [Szeliski et al. 2007; Boykov and Jolly 2001; Kolmogorov and Zabih 2004; Li et al. 2004; Rother et al. 2004; Wang et al. 2005], Random Walker [Grady and Sinop 2008; Sinop and Grady 2007], Total Variation [Unger et al. 2008] and Geodesic models [Bai and Sapiro 2007]. Although impressive results have been obtained, the energy minimization techniques utilized are, in general, not sufficiently efficient (on large images), nor easy to parallelize.

Related work on edge-aware image includes (and is not limited to): anisotropic diffusion [Perona and Malik 1990], image denoising [Szczechanski et al. 2003; Buades et al. 2005], bilateral filtering [Chen et al. 2007; Tomasi and Manduchi 1998; Weiss 2006], non-photorealistic rendering [Boussau et al. 2007; Wang et al. 2004; Winnemoller et al. 2006], image 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 their popularity geodesic distance transforms have so far found only a modest use in the field of image editing. Much of their use has been limited to skeletonization and morphological operations [Falcao et al. 2004], 3D mesh editing and manipulation [Bendels et al. 2003], surface interpolation [Cohen-Or et al. 1998] and shape images [Weber et al. 2008]. In Computer Vision single or multiple (non geodesic) trees have been used for energy minimization in applications such as stereo correspondence [Veksler 2005; Kolmogorov 2006]. In this paper we extend the range of applications considerably.

2 Geodesic Forests

This section describes background on generalized geodesic distances and introduces the Geodesic Forest decomposition.

2.1 Background on geodesic distance transforms

For completeness, we begin by providing a formal definition of the geodesic distance and then define the recently introduced generalized version.

2.1.1 Geodesic Distance Transform

Given a color image \( I(x) : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \), a binary mask \( M(x) \in \{0,1\}^\mathbb{R}^2 \) and a “seed” region (or “object” region) \( \Omega \) with \( x \in \Omega \Leftrightarrow M(x) = 0 \), the unsigned geodesic distance of each pixel \( x \) from \( \Omega \) is defined as:

\[
D(x; M, I) = \min_{(x', M(x')=0)} d(x, x'), \quad \text{with} \quad (1)
\]

\[
d(a, b) = \inf_{\Gamma \in \mathcal{P}_{a,b}} \int_0^1 \sqrt{\left| \Gamma' (s) \right|^2 + \gamma^2 (\nabla I(x) \cdot \Gamma'(s))^2} \, ds. \quad (2)
\]

with \( \mathcal{P}_{a,b} \) the set of all possible paths between the points \( a \) and \( b \) and \( \Gamma(s) : \mathbb{R} \rightarrow \mathbb{R}^2 \) indicating one such path, parameterized by the scalar \( s \in [0,1] \). The spatial derivative \( \Gamma'(s) = \partial \Gamma(s)/\partial s \) represents a vector tangent to the direction of the path. The dot-product in (2) ensures maximum influence for the gradient \( \nabla I \) when it is parallel to the direction of the path \( \Gamma \). The geodesic factor \( \gamma \) weighs the contribution of the image gradient versus the spatial distances. Furthermore, the integral in (2) is the Euclidean length of the 3D path \( \Gamma \) that \( \Gamma \) defines on the \( (x,y,I) \) surface: \( \Gamma(s) = [\Gamma(x(s); \gamma I(\Gamma(x(s)))]. \) Also, for \( \gamma = 0 \), eq. (2) reduces to the Euclidean length of the path \( \Gamma \).

An efficient GPU algorithm for computing geodesic distances is presented in [Weber et al. 2008], while other linear-time CPU algorithms are discussed in [Yatziv et al. 2006; Criminisi et al. 2008] and references therein.

2.1.2 Generalized Geodesic Distance Transform

This section explains the generalized GDT (GGDT) introduced in [Criminisi et al. 2008]. The key difference between the GDT and the GGDT is the fact that in the latter the input seed map \( M \) is more generally a soft, real-valued matrix. Given a map \( M(x) \in [0,1] \) the Generalized Geodesic Distance is defined as follows:

\[
D(x; M, I) = \min_{\psi} \left( d(x, x') + \nu M(x') \right). \quad (3)
\]

with \( d(\cdot) \) as in (2). Mathematically, this is a small change. However, the fact that eq. (3) uses the soft belief of a pixel belonging
to the object of interest means that the latter can be defined probabilistically. This is achieved more economically than having to compute the fully optimized binary segmentation. The parameter \( \nu \) in eq. (3) establishes the mapping between the beliefs \( M \) and the spatial distances. Figure 2 shows an explanatory diagram in the 1D case. Alternatives to eq. (3) (e.g. minimizing the distance \( \sqrt{d^2(x, x') + \nu M^2(x')} \)) may also be considered.

Figure 3 further clarifies these points with a further, explanatory 2D example. Given an image of a flower, the user may use different brush strokes to quickly indicate a foreground object [Li et al. 2004]. In this case the user strokes are used only to compute the foreground (\( F_g \)) and background (\( B_g \)) color models, which are represented as the histograms \( h_{F_g}, h_{B_g} \) over the 3D RGB domain. At this point, for each pixel \( x \) in the image we can compute the log-likelihood ratio as:

\[
L(x) = \log \left( \frac{h_{F_g}(c(x))}{h_{B_g}(c(x))} \right)
\]  

(4)

where \( c(x) \) represents the RGB color for the pixel in \( x \) as a 3-vector. Now, the real-valued seed map \( M \) is obtained via a sigmoid transformation of the log-likelihood ratio as

\[
M(x) = \sigma(L(x)), \quad \text{with} \quad \sigma(L) = \frac{1}{1 + \exp(-L/\mu)}.
\]  

(5)

This operation ensures that \( M(x) \in [0, 1] \). Figure 3c shows the computed seed map in our example. The corresponding GGDT is shown in Figure 3d.

Note that in this example the seed region does correspond to an actual object (a flower) and is determined interactively. Later we will show examples of seed maps computed entirely automatically, and their applications.

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

### 2.2 Implementing the Generalized DGT on the GPU

Our GPU implementation uses nVidia’s CUDA programming environment and builds upon the work of [Weber et al. 2008]. Weber et al describe an algorithm they name the Parallel Marching Method (PMM), which is a CUDA-friendly parallelization of an approximate 4-pass raster scan technique originally introduced in [Danielsson 1980]. They apply their method to multi-chart geometry images to compute geodesic distances on a curved surface, whereas we have adapted it to compute the generalized GGDT and forest decomposition of a natural image.

The original Parallel Marching Method achieves a degree of parallelization by organizing the raster scans so that a block of threads may proceed in parallel. Multiple blocks are overlapped and may also proceed in parallel. The data is organized to use texture memory for the immutable image \( I \) and global memory accessed coherently for the distance map \( D \). Four passes (up, down, left and right) are completed for each iteration of the algorithm. Two copies of the distance map are used: one for the up/down passes, and a transposed version for the left/right passes. For further details, we refer the reader to [Weber et al. 2008].

To adapt this procedure for our generalized GGDT we first initialize the distance map from the soft seed mask as follows: \( D(x) = \nu M(x) \). We then execute the raster scans, so that each pixel is replaced by the minimum of four values:

\[
D(x) = \min \left\{ \begin{array}{l}
D(x) \\
D(x + a_1) + \sqrt{2\rho^2 + \gamma^2[I(x) - I(x + a_1)]^2} \\
D(x + a_2) + \sqrt{2\rho^2 + \gamma^2[I(x) - I(x + a_2)]^2} \\
D(x + a_3) + \sqrt{2\rho^2 + \gamma^2[I(x) - I(x + a_3)]^2}
\end{array} \right. 
\]  

(6)

The three offset vectors \( a_i \) define the local kernel and vary for each scan direction. For example, in the downward pass (fig. 4a) \( a_1 = (-1, -1), a_2 = (0, -1), a_3 = (+1, -1) \). The parameter \( \rho \) represents the spatial scale in the image plane and is fixed to \( \rho = 1 \). This implementation is augmented with an additional output of one byte per pixel, to store the incoming direction from which the minimum distance in eq. (6) is achieved. This back-links map \( B(x) \) is initialized to zero and for each pass, one of three possible direction indices is written at each pixel, depending on which term in eq. (6) gives rise to a minimum. For example in the left pass, direction indices of 3-5 are used (fig. 4b). In each pass, if the distance stored at a pixel remains unchanged by the minimum operation, then the
back-link value is not altered either. As with the distance map, we maintain two mutually transposed versions of the back-links map for memory access coherence. After executing all four passes, the values in $B$ trace minimum cost paths for each position $x$ (fig. 4c and fig.3f).

**Comparison with state of the art** We timed our implementation on an nVidia GeForce 280 GTX. Each timing represents a complete iteration of four consecutive passes. As shown in fig. 5 our modification of Weber’s algorithm achieves similar timings to those reported in [Weber et al. 2008] (i.e. $\sim$ 40ms on a 9 Mpixel grid). Interestingly, no extra processing time is required to use soft seed maps as opposed to binary ones. Storing back-links at the same time as distances is achieved with only about an additional 20% of the execution time. This is a very reasonable cost to pay for the added advantage of having access to all minimum cost paths as well as the associated distances. For added clarity our timings are also reported in the table below. Interestingly, the percentage difference due to the back-links decreases with larger images.

<table>
<thead>
<tr>
<th>Image size (Mpix)</th>
<th>Gener. GDT only (ms)</th>
<th>Distance and back-links (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>7.5</td>
<td>9.5</td>
</tr>
<tr>
<td>2.1</td>
<td>18.1</td>
<td>22.1</td>
</tr>
<tr>
<td>5.5</td>
<td>37.7</td>
<td>46.3</td>
</tr>
<tr>
<td>12.6</td>
<td>69.8</td>
<td>85.9</td>
</tr>
<tr>
<td>20</td>
<td>84.2</td>
<td>99.3</td>
</tr>
</tbody>
</table>

**2.3 Geodesic trees and forests**

Given the distance $D(x)$ and the computed back links (fig. 3d,e), for each point in the image we can follow the chain of back links and reconstruct the associated minimum cost path. As illustrated in fig. 3f such paths terminate at a small set of points. Those automatically determined points constitute the roots of a set of trees whose union covers the entire image (each pixel belongs to at least one tree) and which are disjoint (each pixels belongs to only one tree). We call this set of trees a Geodesic Forest.

Figure 6 further clarifies such a decomposition, where each of the four images represents a different forest, produced by different values of the geodesic factor $\gamma$. Larger values yield fewer trees with bendier and more convoluted branches. In contrast to the IFT algorithm [Falcao et al. 2004] here the tree roots are computed automatically from a probabilistic seed map. The number of trees in the forest is automatically selected by our algorithm. The behavior shown in fig. 3f is somewhat reminiscent of the hill climbing Mean Shift algorithm in [Comaniciu and Meer 2002].

Having described our image representation and the associated computation algorithm, we next discuss its applications.

**2.3.1 Simulating diffusion processes**

The branches of a Geodesic Forest are, by construction, paths of minimum cost and may be interpreted as paths of least resistance through an inhomogeneous, physical medium. Therefore, Geodesic Forests may be used to simulate physical phenomena such as electrical conduction, fluid propagation, magnetism and diffusion.

Figure 7 illustrates a simulation where some green paint is diffused from a seed region through a carved physical medium. In this example the seed mask $M$ is defined as a blank image with a single blob of paint placed in the middle of the carved region (fig. 7b). The gradient field $\nabla I$ is defined as the gradient of the carved surface.

The Geodesic Forest decomposition produces a number of trees with roots in the initial paint location. Diffusing paint along the tree branches, from the root towards the leaves amount simply to computing the weight

$$W(x) = Z e^{-\frac{\rho^2(x)}{\sigma^2}}$$

(7)
with $Z$ a normalization constant which ensures that $W \in [0, 1]$, and $\phi$ a user-defined diffusion parameter. The quantity $W(x)$ is defined as a function of the amount of paint which reaches position $x$. Then the diffused image (left column in fig. 7) is constructed as

$$I(x) = W(x) S(x) + (1 - W(x)) C$$

with $S(x)$ the original surface color (uniform white in this toy example) and the constant $C$ the color of the paint (green here). The diffusion parameter $\phi$ controls the spatial extent of the diffusion process, with larger values of $\phi$ yielding spatially larger diffusion effects. Paint thinning can also be simulated by multiplying $W$ by a function of the area affected by diffusion.

Being able to diffuse information from one image region to other regions anisotropically and efficiently is the basis of the editing applications described next.

### 3 Applications to image and video editing

This section presents numerous applications of Geodesic Forests in image and video editing. The reader is kindly invited to view the associated video for further results and animations.

#### 3.0.2 Ink painting

Figure 8 shows a photograph of the Notre Dame cathedral in Paris turned into an ink brush painting, with added pen strokes. This effect is obtained by computing the soft seed mask $M$ from the image content itself as

$$M(x) = Z|\nabla Y(x)|,$$

with $Y$ the luma channel and $Z$ a normalization factor to ensure that $M \in [0, 1]$. The result in fig 8b is simply a visualization of the diffusion weight $W(x)$ as defined in eq. (7). As discussed in section 2.3.1 this effect can be interpreted as the diffusion of the ink from regions of high gradient in the image towards the flatter areas. The only interaction necessary here is in choosing the amount of diffusion by setting the value of $\phi$. The final composition (fig. 8c) is obtained by adding automatic pen strokes. Following [Winnemoller et al. 2006], pen strokes are computed as the contrast-enhanced gradient magnitude of the original image.

As demonstrated in fig. 9, ink painting on a rough, textured surface can also be easily simulated. This is achieved simply by replacing the gradient field $\nabla I$ in the computation of the GGDHT (eq. 3) with $\max(\nabla I, \nabla J)$; where $\nabla J$ is a different gradient field. For instance, in the two examples in fig. 9 $J$ was chosen to be an image with diagonally oriented texture. This operation allows ink to diffuse into the crevasses of the rough surface.

#### 3.0.3 Geodesic texture flattening

Edge-aware texture flattening is obtained here by: i) decomposing the image into a small number of components, ii) applying geodesic
In the case of an inhomogeneous support the ink diffuses along directions of minimum resistance within the support surface. This is simulated in our algorithm simply by modifying the gradient field $\nabla I$.

In detail, given a color image $I$ and its luma $Y$ all pixel intensities are clustered via conventional K-means into $K$ cluster centres, with mean luma values denoted $\mu_i$ and corresponding standard deviations denoted $\sigma_i$ ($i$ is a cluster index).

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

$$M_i(x) = 1 - \lambda_i e^{-\frac{1}{2} \left( \frac{Y(x) - \mu_i}{\sigma_i} \right)^2}$$  \hspace{1cm} (9)

with $\lambda_i$ normalization constants (see fig. 10b for an illustration).

For each mask $M_i$ we then compute the corresponding distance $D_i(x)$ as in (3), and thus the diffusion weights $W_i(x)$ following eq. (7). The flattened luma is obtained as a weighted average of the cluster colors:

$$Y'(x) = \sum_i^K \mu_i W_i(x).$$  \hspace{1cm} (10)

Combining the flattened luma $Y'$ with the unprocessed chromaticity channels yields the output color image (fig. 10d). This operation may be interpreted in physical terms as constructing an image by pouring $K$ paint shades onto a canvas in different regions and letting the paint diffuse and mix. Please note that although the minimum cost paths are not used explicitly here, the geodesic distances are still defined by the underlying forest structure.

In this application the soft maps are computed automatically and the user is only required to select the amount of diffusion $\phi$ and the number of clusters $K$. In this paper no more than four clusters have been used thus making the whole process extremely efficient. The reason why a very small number of cluster suffices (even on images with complex luma statistics) is due to the probabilistic nature of the GGDT's, which can take into account the uncertainty of pixels which do not fully belong to a single cluster and whose contribution is spread across multiple clusters.

The bottleneck of our flattening algorithm is in the computation of the $K$ distance maps. However, note that those can be computed completely independently from one another and in parallel. Also, our GPU-based GGDT algorithm is itself parallel and extremely efficient. As an example, 1 Mpix images are flattened in about 30ms, and 10 Mpix ones in about 160 ms. Notice that here no data subsampling has been necessary (thus avoiding possible loss of fine details, such as thin structures). Compared to the work in [Chen et al. 2007] our representation requires relatively small memory use (even for high res. images), and no large speed reduction is observed for larger images. Figure 11 shows some typical results. As shown in the accompanying video, our GPU-based texture flattening technique allows abstraction and cartoonization of videos in real time.

### 3.0.4 Painterly effects

The work in [Bousseau et al. 2007] achieved impressive results on video watercolorization. In that work image simplification was achieved by simple morphological operations and considerable emphasis was placed on being able to see the paper texture in the final product. The flattening procedure described in the previous section achieves image simplification for us. Furthermore, as in section 3.0.2, we can use a combination of image gradient and the gradient of the paper texture to drive the diffusion of the paint. This produces watercolor-like effects such as those in fig. 12 where pa-
per with two different texture types have been used.

Furthermore, multiplying the ink painting image by the luma of an input color image produce the effect shown in fig. 13, with some dramatic “burning” effects combined with little Voronoi-like cells which enhance the tree foliage.

3.0.5 Geodesic image plotting

Interesting animations may be generated by letting the trees grow slowly from their roots, and paint an image as they do so. Figure 14 shows four frames of such an animation. The picture of the house is being built progressively starting from the tree roots, lengthening the branches of the forest until each pixel has been visited. Natural looking intricate patterns are drawn at different stages of the animation. In this example only one root node has been selected on the top of the farther chimney. Two more example images are shown in fig. 1. The reader is kindly invited to view the accompanying video to fully appreciate the effect.

3.0.6 Diffusive painting

Figure 15 shows two examples of diffusive painting. In fig. 15a pixels in the sky of the original photograph are diffused along the forest branches, from the leaves towards the roots. This produces an effect where the clouds look like they are being attracted towards the large dishes of the radio-telescope. In fig. 15b pixel colors are diffused the other way around, from the root nodes towards the leaves. This yields an effect akin to electricity discharges. More examples and animations are shown in fig. 1 as well as the accompanying video.

3.0.7 2D image resizer

Our Geodesic Forest representation generalizes the single curve, seam carving Dynamic Programming approach in [Avidan and Shamir 2007]. In this section we show how content-driven image resizing can be achieved via Geodesic Forests, where the width and the height of the image are jointly decreased (or increased) by one pixel at each iteration.

Interactive resizing. A first technique relies on the interactive selection of one seed point in a region where image material can be
safely discarded (e.g. textureless regions). In this case the matrix $M$ is set to 0 on such seed point and 1 everywhere else. The (degenerate) geodesic forest consists in the single tree rooted at the selected point. Then, we select the four-branch star subtree which has exactly one leaf per side of the image frame. For each of the 4 sides the leaf node corresponds to the pixel with the minimal geodesic distance from the root (cf. fig. 16-right). This defines two pixel chains joining the two opposite sides of the image frame. Removing all the pixels in those chains results in a new image with reduced size. If several seed points are specified, they can be either visited in a fixed arbitrary order, or randomly.

Automatic resizing. A fully automatic version of this resizing technique consists in automatically selecting the forest root nodes within flat image regions (e.g. locations of low gradient magnitude). Results of such technique are presented in fig. 16 and in the accompanying video. It can also be shown that Geodesic Forests generalize intelligent scissors operations [Mortensen and Barrett 1995] as well as panoramic image stitching. Space considerations prevent us from describing details.

4 Limitations
The algorithms presented in this paper work well for the great majority of images, however, some difficulties may arise for more intricate images. For instance, texture flattening complex images, rich in thin structures, may lead to less interesting results. This is probably due to the difference between what humans perceive to be important object contours and the computed image gradients.

In rare occasions more iterations of our GGDT algorithm may be necessary to achieve better distance accuracy; with consequently slower run times. Although usually 2 iterations suffice, in extreme examples (such as fig. 7) up to 6 iterations may be necessary.

5 Conclusion
This paper has presented Geodesic Forests, a new representation of digital color images. The key idea is to shift the burden of data dependent processing from the image editing algorithms to the core image representation. This enables casting edge-sensitive 2D operations as a set of efficient 1D transformations.

The proposed image decomposition is achieved via an efficient GPU-based algorithm which enables processing high resolution images in tens of milliseconds.

Geodesic Forests have also been demonstrated to be a unified framework for the implementation of a number of existing and novel image and video editing effects as well as physics-based simulations. Extensions to 3D, tomographic images and video cubes are relatively straightforward. Further possible applications include exact energy minimization via belief propagation on Geodesic Forest structures.

References


Boykov, J., and Jolly, M.-P. 2001. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In ICCV.


Figure 16: 2D, content-aware image resizing. (Left) Versions of the same image with different sizes (original, input image from Avidan et al.). Please note that the size of the girl and the foreground bird remain constant throughout while removing redundant pixels. In this example the geodesic forest has been computed completely automatically. (Right) One of the automatically computed geodesic trees used for resizing.


Sud, A., Govindaraju, N., Gayle, R., and Manocha, D.

