Image-Based Modeling and Photo Editing

Byong Mok Oh Max Chen Julie Dorsey Frédo Durand

Laboratory for Computer Science
Massachusetts Institute of Technology *

Abstract

We present an image-based modeling and editing system that takes a single photo as input. We represent a scene as a layered collection of depth images, where each pixel encodes both color and depth. Starting from an input image, we employ a suite of user-assisted techniques, based on a painting metaphor, to assign depths and extract layers. We introduce two specific editing operations. The first, a "clone brushing tool," permits the distortion-free copying of parts of a picture, by using a parameterization optimization technique. The second, a "texture-illuminance decoupling filter," discounts the effect of illumination on uniformly textured areas, by decoupling large- and small-scale features via bilateral filtering. Our system enables editing from different viewpoints, extracting and grouping of image-based objects, and modifying the shape, color, and illumination of these objects.

1 Introduction

Despite recent advances in photogrammetry and 3D scanning technology, creating photorealistic 3D models remains a tedious and time consuming task. Many real-world objects, such as trees or people, have complex shapes that cannot easily be described by the polygonal representations commonly used in computer graphics. Image-based representations, which use photographs as a starting point, are becoming increasingly popular because they allow users to explore objects and scenes captured from the real world. While considerable attention has been devoted to using photographs to build 3D models, or to rendering new views from photographs, little work has been done to address the problem of manipulating or modifying these representations. This paper describes an interactive modeling and editing system that uses an image-based representation for the entire 3D authoring process. It takes a single photograph as input, provides tools to extract layers and assign depths, and facilitates various editing operations, such as painting, copy-pasting, and relighting.

Our work was inspired, in part, by the simplicity and versatility of popular photo-editing packages, such as *Adobe Photoshop*. Such tools afford a powerful means of altering the appearance of an image via simple and intuitive editing operations. A photo-montage, where the color of objects has been changed, people have been removed, added or duplicated, still remains convincing and fully "photorealistic." The process involves almost no automation and is entirely driven by the user. However, because of this absence of automation, the user has direct access to the image data, both conceptually and practically. A series of specialized interactive tools complement one another. Unfortunately, the lack of 3D information sometimes imposes restrictions or makes editing more tedious. In this work, we overcome some of these limitations and introduce two new tools that take advantage of the 3D information: a new "distortion-free clone brush" and a "texture-illuminance decoupling filter."

Clone brushing (a.k.a. rubberstamping) is one of the most powerful tools for the seamless alteration of pictures. It interactively copies a region of the image using a brush interface. It is often used to remove undesirable portions of an image, such as blemishes or distracting objects in the background. The user chooses a source region of the image and then paints over the destination region using a brush that copies from the source to the destination region. However, clone brushing has its limitations when object shape or perspective causes texture foreshortening: Only parts of the image with similar orientation and distance can be clone brushed. Artifacts also appear when the intensity of the target and source regions do not match.

The existing illumination also limits image editing. Lighting design can be done by painting the effects of new light sources using semi-transparent layers. However, discounting the existing illumination is often difficult. Painting "negative" light usually results in artifacts, especially at shadow boundaries. This affects copy-pasting between images with different illumination conditions, relighting applications and, as mentioned above, clone brushing.

In this paper, we extend photo editing to 3D. We describe a system for interactively editing an image-based scene represented as a layered collection of depth images, where a pixel encodes both color and depth. Our system provides the means to change scene structure, appearance, and illumination via a simple collection of editing operations, which overcome a number of limitations of 2D photo editing.

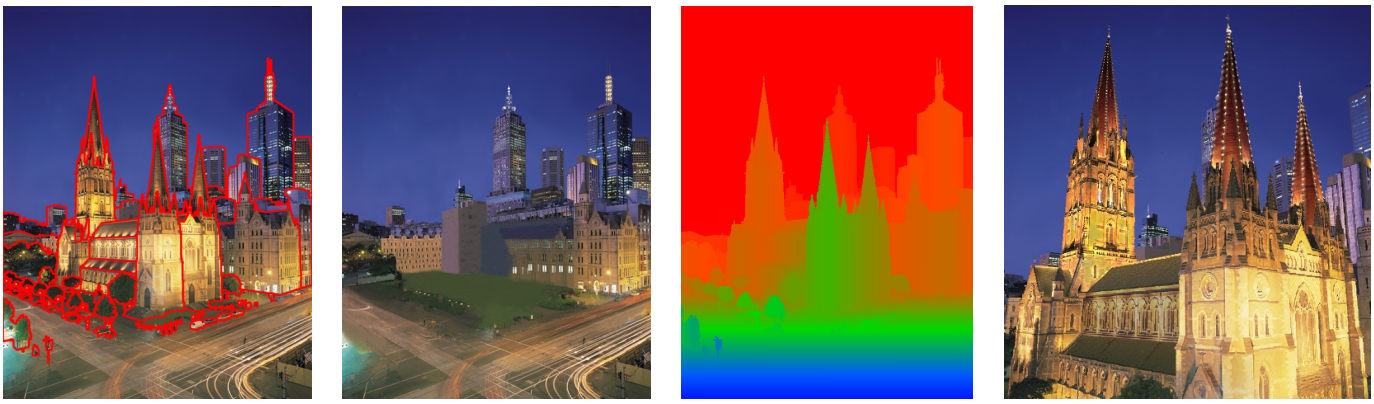
Many processes involving the editing of real images, for aesthetic, design or illustration purposes, can benefit from a system such as ours: designing a new building in an existing context, changing the layout and lighting of a room, designing a virtual TV set from a real location, or producing special effects. Some of these applications already obtain impressive results with 2D image-editing tools by segmenting the image into *layers* to permit separate editing of different entities. A particular case is cell animation, which can take immediate and great advantage of our system.

We will see that once this segmentation is performed, an image-based representation can be efficiently built, relying on the ability of the user to infer the spatial organization of the scene depicted in the image. By incorporating depth, powerful additional editing is possible, as well as changing the camera viewpoint (Fig. 1).

One of the major advantages of image-based representations is their ability to represent arbitrary geometry. Our system can be used without any editing, simply to perform 3D navigation inside a 2D image, in the spirit of the *Tour into the Picture* system [HAA97], but with no restriction on the scene geometry.

*http://graphics.lcs.mit.edu/

ACM SIGGRAPH 2001, 12-17 August 2001, Los Angeles, CA, USA



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

Figure 1: St Paul's Cathedral in Melbourne. (a) Image segmented into layers (boundaries in red). (b) Hidden parts manually clone brushed by the user. (c) False-color rendering of the depth of each pixel. (d) New viewpoint and relighting of the roof and towers.

1.1 Previous work

We make a distinction between two classes of image-based techniques. The first is based on sampling. View warping is a typical example that uses depth or disparity per pixel [CW93, LF94, MB95, SGHS98]. Higher dimensional approaches exist [LH96, GGSC96], but they are still too costly to be practical here. The representation is purely independent of the geometry, but for real images, depth or disparity must be recovered, typically using stereo matching. Closer to our approach, Kang proposes to leave the depth assignment task to the user via a painting metaphor [Kan98], and Williams uses level sets from silhouettes and image grey levels [Wil98].

The second class concerns image-based modeling systems that take images as input to build a more traditional geometric representation [Pho, Can, DTM96, FLR+95, LCZ99, POF98, Rea]. Using photogrammetry techniques and recovering textures from the photographs, these systems can construct photorealistic models that can be readily used with widely-available 3D packages. However, the use of traditional geometric primitives limits the geometry of the scene, and the optimization techniques often cause instabilities.

Our goal is to bring these two classes of approaches together. We wish to build a flexible image-based representation from a single photograph, which places no constraints on the geometry and is suitable for editing.

The work closest to ours is the plenoptic editing approach of Seitz et al. [SK98]. Their goal is also to build and edit an image-based representation. However, their approach differs from ours in that their system operates on multiple views of the same part of the scene and propagates modifications among different images, allowing a better handling of view-dependent effects. Unfortunately, the quality of their results is limited by the volumetric representation that they use. Moreover, they need multiple images of the same object viewed from the outside in.

We will see that some of our depth acquisition tools and results can be seen as a generalization of the *Tour Into the Picture* approach, where central perspective and user-defined billboards are used to 3D-navigate inside a 2D image [HAA97]. Our work, however, imposes no restrictions on the scene geometry, provides a broader range of depth acquisition tools, and supports a variety of editing operations.

Our work is also related to 3D painting, which is an adaptation of popular 2D image-editing systems to the painting of textures and other attributes directly on 3D models [HH90, Met]. This approach is, however, tightly constrained by the input geometry and the texture parameterization.

1.2 Overview

This paper makes the following contributions:

- An image-based system that is based on a depth image representation organized into layers. This representation is simple, easy to render, and permits direct control. It is related to layered depth images (LDIs) [SGHS98], but offers a more meaningful organization of the scene. We demonstrate our system on high-resolution images (megapixels, as output by current high-end digital cameras).
- A new set of tools for the assignment of depth to a single photograph based on a 2D painting metaphor. These tools provide an intuitive interface to assign relevant depth and permit direct control of the modeling.
- A non-distorted clone brushing operator that permits the duplication of portions of the image using a brush tool, but without the distortions due to foreshortening in the classical 2D version. This tool is crucial for filling in gaps, due to occlusions in the input image, that appear when the viewpoint changes.
- A filter to decouple texture and illuminance components in images of uniformly textured objects. It factors the image into two channels, one containing the large-scale features (assumed to be from illumination) and one containing only the small-scale features. This filter works even in the presence of sharp illumination variations, but cannot discount shadows of small objects. Since it results in uniform textures, it is crucial for clone brushing or for relighting applications.
- Our system permits editing from different viewpoints, e.g. painting, copy-pasting, moving objects in 3D, and adding new light sources.

2 System overview

2.1 Layers of images with depth

All elements of our system operate on the same simple data structure: images with depth [CW93]. This permits the use of standard image-based rendering techniques [CW93, MB95, SGHS98, MMB97]. Depth is defined up to a global scale factor.

The representation is organized into layers (Fig. 1(a) and 2), in the spirit of traditional image-editing software and as proposed in

```
layer {
   reference camera  : transformation matrix
   color channels    : array of floats
   alpha channel     : array of floats
   depth channel     : array of floats
   optional channels : arrays of floats
}
```

Figure 2: Basic layer data structures.

computer vision by Wang and Adelson [WA94]. An alpha channel is used to handle transparency and object masks. This permits the treatment of semi-transparent objects and fuzzy contours, such as trees or hair. Due to the similarity of data structures, our system offers an import/export interface with the Adobe Photoshop format that can be read by most 2D image-editing programs.

The image is manually segmented into different layers, using selection, alpha masks, and traditional image-editing tools (Fig. 1(a)). This is typically the most time-consuming task. The parts of the scene hidden in the input image need to be manually painted using clone brushing. This can more easily be done after depth has been assigned, using our depth-corrected version of clone brushing.

A layer has a reference camera that describes its world-to-image projection matrix. Initially, all layers have the same reference camera, which is arbitrarily set to the default OpenGL matrix (i.e. identity). We assume that the camera is a perfect pinhole camera, and unless other information is available, that the optical center is the center of the image. Then, only the field of view needs to be specified. It can be entered by the user, or a default value can be used if accuracy is not critical. Standard vision techniques can also be used if parallelism and orthogonality are present in the image (see Section 3). Note that changing the reference camera is equivalent to moving the objects depicted in the layer in 3D space. Throughout the paper we will deal with two kinds of images: *reference images* that correspond to the main data structure, and *interactive images* that are displayed from different viewpoints to ease user interaction. The degree to which the viewpoint can be altered, without artifacts, is dependent on the particular scene, assigned depth, and occluded regions.

Our organization into layers of depth images is related to the LDIs [SGHS98], with a major difference: In an LDI, the layering is done at the pixel level, while in our case it is done at a higher level (objects or object parts). LDIs may be better suited for rendering, but our representation is more amenable to editing, where it nicely organizes the scene into different higher-level entities.

Additional channels, such as texture, illuminance, and normal (normals are computed for each pixel using the depth of the 4 neighboring pixels), may be used for specific applications (relighting in particular).

2.2 System architecture

The architecture of our system is simple, since it consists of a set of tools organized around a common data structure (Fig. 3). It is thus easy to add new functionality. Although we present the features of our system sequentially, all processes are naturally interleaved. Editing can start even before depth is acquired, and the representation can be refined while the editing proceeds.

Selection, like channels, is represented as an array corresponding to the reference image. Each pixel of each layer has a selection value, which can be any value between 0 and 1 to permit feathering. Selection is used not only for copy-pasting, but also for restricting the action of the tools to relevant areas.

The interactive display is performed using triangles [McM97, MMB97] and hardware projective texture mapping [SKvW+92]. The segmentation of the scene into layers greatly eliminates rubber-sheet triangle problems. Obviously, any other image-based-rendering technique such as splatting could be used [CW93, SGHS98, MB95].

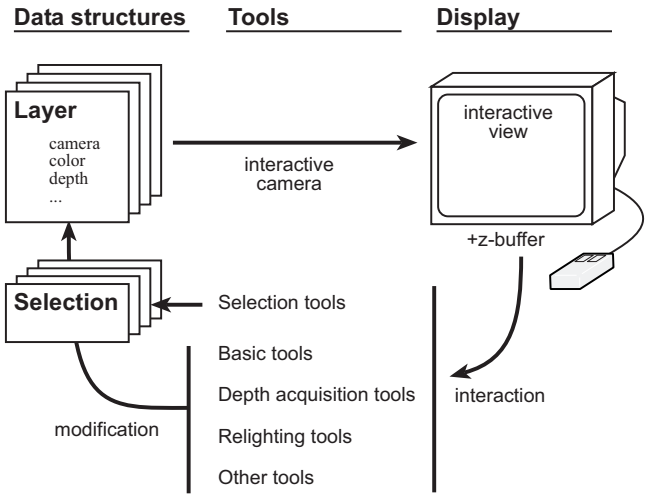


Figure 3: Architecture of our system.

The tools, such as depth assignment, selection or painting, can be used from any interactive viewpoint. The z-buffer of the interactive view is read, and standard view-warping [CW93, McM97] transforms screen coordinates into 3D points or into pixel indices in the reference image. The texture parameter buffers of Hanrahan and Haeberli could also be used [HH90].

3 Depth Assignment

We have developed depth assignment tools to take advantage of the versatility of our representation. The underlying metaphor is to paint and draw depth like colors are painted, in the spirit of Kang [Kan98]. This provides complete user control, but it also relies on the user's ability to comprehend the layout of the scene. The level of detail and accuracy of depth, which can be refined at any time, depend on the target application and intended viewpoint variation.

However, even if a user can easily infer the spatial organization and shapes depicted in the image, it is not always easy to directly paint the corresponding depth. Hence we have also developed hybrid tools that use pre-defined shapes to aid in painting accurate depth. In the development of our interface, we have emphasized 2D, rather than 3D interaction, the direct use of cues present in the image, and the use of previously-assigned depth as a reference.

Depth can be edited from any interactive viewpoint, which is important in evaluating the effects of current manipulations. Multiple views can also be used [Kan98]. We will see that some tools are easier to use in the reference view, where image cues are more clearly visible, while for others, interactive views permit a better judgment of the shape being modeled.

The use of selection also permits us to restrict the effect of a tool to a specific part of the image, providing flexibility and finer control. And since our selections are real-valued, the effect of depth tools can be attenuated at the selection boundary to obtain smoother shapes. In our implementation, we use the selection value to interpolate linearly between the unedited and edited values. Smoother functions, such as a cosine, could also be used.

In contrast to optimization-based photogrammetry systems [Can, DTM96, FLR+95], the field of view of the reference camera must be specified as a first step (as aforementioned, we assume a perfect

pinhole camera). If enough information is available in the image, the field of view can be calculated (Section 3.2). The user can also set the focal length manually. Otherwise, the focal length is set to a default value (50mm in practice).

3.1 Depth painting

The user can directly paint depth using a brush, either setting the absolute depth value or adding or subtracting to the current value (chiseling). Absolute depth can be specified using a tool similar to the color picker, by clicking on a point of the image to read its depth. The relative brush tool is particularly useful for refining already-assigned depth (Fig. 5(b)). The size and softness of the brush can be interactively varied.

The whole selected region can also be interactively translated. Translation is performed along lines of sight with respect to the reference camera: The depth of each selected pixel is incremented or decremented (Fig. 4). However, it is desirable that planar objects remain planar under this transformation. We do not add or subtract a constant value, but instead multiply depth by a constant value. Depth-translating planar objects therefore results in parallel planar objects.

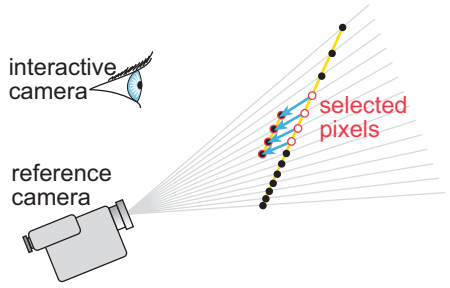
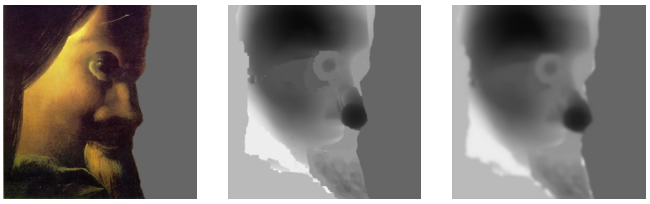


Figure 4: Depth translation is performed along lines of sight with respect to the reference camera.

In the spirit of classical interactive image-editing tools, we have developed local blurring and sharpening tools that filter the depth channel under the pointer (Fig. 5(c)). Blurring smooths the shape, while sharpening accentuates relief. Local blurring can be used to "zip" along depth discontinuities, as described by Kang [Kan98]. A global filtering is also possible, in particular blurring to smooth the 3D shape, noise to add complexity, and median filtering to remove outliers.



(a) (b) (c)

Figure 5: (a) Face. (b) Chiseled depth. (c) Blurred depth.

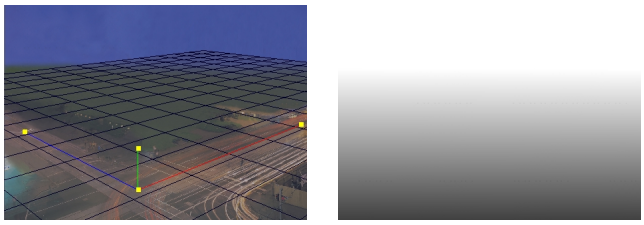
Similar to Kang's [Kan98] and William's [Wil98] methods, the user can use the rgb channels to assign depth. This is motivated by cases where darker values correspond to more distant pixels (such as trees) or by atmospheric perspective making distant objects bluer. The user specifies the z_{min} and z_{max} depth values, and the vector specifying color direction $\vec{C}$ (e.g. dark to light or amount of blue), and the effect can be applied absolutely or relatively. In the absolute case, for example, depth is then specified from the color at each pixel $\vec{c}(x,y)$ from:

$$z(x,y) = z_{min} + (z_{max} - z_{min}) * \vec{C} \cdot \vec{c}(x,y).$$

3.2 Ground plane and reference depth

The tools presented so far work best if some initial depth has been assigned, or if a reference is provided for depth assignment. Similar to the perspective technique used since the Renaissance, and to the spidery mesh by Horry et al. [HAA97], we have found that the use of a reference ground plane greatly simplifies depth acquisition and improves accuracy dramatically, since it provides an intuitive reference. The position with respect to the ground plane has actually been shown to be a very effective depth cue [Pal99]. Specifying a ground plane is typically the first step of depth assignment.

The ground plane tool can be seen as the application of a gradient on the depth channel (Fig. 6). However, an arbitrary gradient may not correspond to a planar surface. In our system, the user specifies the horizon line in the reference image, which constrains two degrees of freedom, corresponding to a set of parallel planes. The remaining degree of freedom corresponds to the arbitrary scale factor on depth. We can thus arbitrarily set the height of the observer to 1, or the user can enter a value.



(a) (b)

Figure 6: (a) Ground plane. (b) Depth map.

We have also implemented the method by Liebowitz et al. [LCZ99] that allows the acquisition of architectural models and camera parameters from one image using parallelism and orthogonality constraints. This provides both the camera parameters and an accurate reference ground plane. This also allows us to compute the position of the optical axis if it is not in the center of the image.

Depth picking and depth painting can then easily be used to depth-paint billboards parallel to the image plane. Since most objects in a scene touch the ground, or their projection on the ground can be inferred by the user, this proves to be very efficient. This is similar to the placement of billboards on the "spidery mesh" of Horry et al. [HAA97]. However, their system is limited to central perspective and polygonal orthogonal objects, while we can refine our representation and obtain arbitrary shapes.

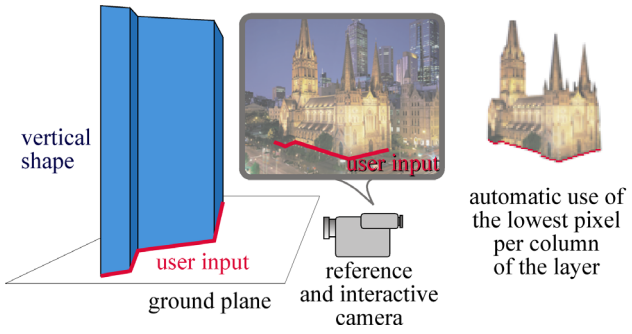


Figure 7: Vertical tool. The user draws the contact of the vertical geometry with the ground plane.

We have extended the notion of billboards to allow the user to paint the depth of arbitrary vertical objects. This works best when the interactive camera is set to the reference camera: The contact or projection of the object on the ground plane is drawn, and a vertical depth is extruded (Fig. 7). In practice, the contact drawn by the user is represented as a polyline, the corresponding vertical polygons are rendered using OpenGL, and the z-buffer is read to update the selected pixels.

We are not limited to the use of a planar ground reference. The 3D locations of the points of the contact polyline drawn by the user are read from the interactive z-buffer. This means that if the ground has been chiseled or translated for better terrain modeling, vertical objects will be extruded accordingly.

We have also implemented an automatic version that processes the whole selection or layer at once. It assumes that the layer or selection is in contact with the ground reference at its lowest pixels. Each column of pixels in the reference image is assigned the depth corresponding to its lowest visible or selected pixel.

3.3 Geometric primitives

Some geometric shapes, such as boxes, spheres, or cylinders, are hard to depth-paint accurately. We therefore utilize geometric primitives that can be drawn transparently as 2D objects. For example, the user draws a circle or clicks on three points to assign spherical depth. We use similar interfaces for cylinders (the user draws the edges), boxes (the user draws three edges of a corner), and pyramids (the user draws the base and apex). These tools work best when used from the reference camera.

The primitive is rendered from the reference camera using OpenGL, and the z-buffer is read to assign depth. This requires the use of a depth buffer at the resolution of the reference image. Since our system treats images that can be larger than the screen, we use tiling.

Once the image projection of the primitive has been provided, its distance must be specified. The user can use an arbitrary distance and then refine it with the translation tool. He can also use the already-assigned depth as a reference by clicking on one point. By default, the first point clicked by the user is used as a reference depth (e.g. corner of a box).

To improve the quality of depth when a ground plane has been assigned, the user can use *primitive snapping* to enforce the verticality of boxes, cylinders or pyramids, or to constrain them along the normal of a given pixel. A least-square error minimization is then run to optimize the 3D shape position.

3.4 Organic shapes

In the spirit of Williams [Wil98] and related to the Teddy fast-modeling system [IMT99], we propose a tool that assigns depth using level sets. It is well suited to giving organic shapes a bulgy appearance, by specifying more distant depth at the boundary and closer depth in the center (Fig. 8(a) and (b)). This tool is relative, and the range r of depth addition is specified by the user.

We compute the level sets using an erosion technique (e.g. [SD95]). The initial active interior of the object is defined as the set of pixels of the layer or selection with non-null alpha. The distance to the boundary d_{bound} is initialized to 0, and we iteratively "erode." For each iteration, we discard pixels that have a non-active neighbor, and increment the distance of active pixels by 1.

We use the normalized distance to the centroid $d' = 1 - \frac{d_{bound}}{d_{bound}max}$ and update depth according to $z = z + r\sqrt{1-d'^2}$. This formula was chosen because it assigns a spherical shape to a disk under orthographic projection.

3.5 Faces

The specific case of human faces is important. The output of the impressive morphable model of Blanz et al. [BV99] could be used to retrieve accurate depth. However, this technique is not easy to implement since it requires a large database of 3D face scans, and unfortunately, it takes tens of minutes to acquire the geometry of a face from a photograph.

We have developed a simpler method that trades accuracy and user intervention for speed and simplicity. This method could be further generalized to a broader class of template shapes. We use a generic arbitrary 3D face model, optimize its 3D position to match the photograph, and then use 2D morphing to refine the match (Fig. 8).

The user specifies correspondence points between the image and the 3D model. These points are used to find the rotation, scale, and position of the 3D model using Levenberg-Marquardt optimization [PSVF92]. Rotation and scale are optimized independently to avoid shear in the resulting transformation. The 3D face is rendered, and the z-buffer is read back. We then use the same correspondence points to morph the z-buffer and obtain a better match with the image using triangle-based morphing and linear interpolation [GDCV98].

4 Non-distorted clone brushing

Standard clone brushing has its limitations when perspective causes texture foreshortening (Fig. 9(a)). In practice, only parts with similar orientation and distance to the camera can be clone brushed. Moreover, only regions of similar intensity can be copied. The former problems will be treated in this section, while the latter will be addressed in the next section.

Since our system has information about depth, we can correct distortion due to both perspective and surface shape. In the general case of arbitrary geometry, the problem is similar to low-distortion texture mapping: We want to map the source region of the image-based representation to the destination, with as little distortion as possible. Our idea is to compute a (u,v) texture parameterization for both the source and destination regions, and use this mapping for the clone brush.

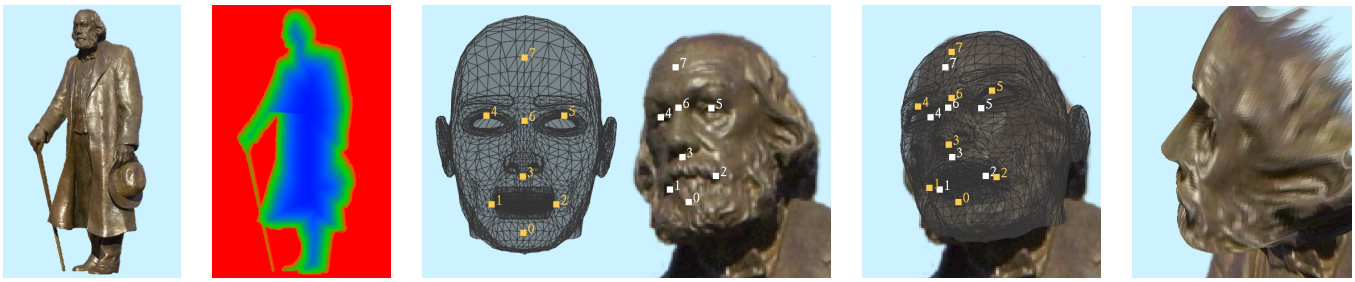
4.1 Non-distorted parameterization

Our parameterization optimization is based on the work by Levy et al. [LM98, Mal89], with three important differences: Our approach is local around the clone-brushed regions, has no boundary condition, and needs to run in real-time. We first quickly review the key points of their method in order to describe the specifics of our flood-fill adaptation. This overview is presented in our specific context, where each pixel is seen as a vertex connected to its 4-neighbors. We refer the reader to their article for details [LM98, Mal89].

Levy et al. propose to minimize two classes of distortions: angular (preserve orthogonal angles) and iso-parametric distance (make isolines equidistant). The former requires that the gradient of u and v be orthogonal, and the latter requires constant magnitude for their gradients. Different weights can be assigned to emphasize one constraint over another.

They use *discrete smooth interpolation* [Mal89], where each discrete value (u or v at a pixel in our case) is smoothed using a linear combination of its neighbors. The combinations minimize a *roughness* function to obtain a valid mapping, and quadratic constraints to minimize distortion. Smoothing steps on u and v are interleaved and iterated.

In our case, the roughness R for a pixel p depends on its 4-neighbors $N(p)$:



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

Figure 8: Organic and face tools. (a) Initial statue image. (b) False-color depth map after level set tool. (c) User-specified correspondence points. (d) Transformation after optimization. (e) Face from new viewpoint after depth is applied.

$$R(u,v) = \sum_p \left(-4u_p + \sum_{p' \in N(p)} u_{p'} \right)^2 + \left(-4v_p + \sum_{p' \in N(p)} v_{p'} \right)^2 . \quad (1)$$

The minimum of R is reached when its partial derivatives $\frac{\partial R}{\partial u}$ and $\frac{\partial R}{\partial v}$ are null for each pixel p. This yields an expression of u and v at a pixel as a linear combination of the values of the neighbors of its neighbors:

$$\begin{aligned} u_p &= \sum_{p' \in N(N(p))} a^u_{p,p'} u_{p'} \\ v_p &= \sum_{p' \in N(N(p))} a^v_{p,p'} v_{p'}, \end{aligned} \quad (2)$$

where the a^i_j are given coefficients.

Distortions are minimized by introducing a set of linear constraints C depending on the gradients of u and v. Levy et al. use a least-square approach, which defines a generalized roughness $R^\star$:

$$R^\star(u,v) = R(u,v) + \sum_{c \in C} \left\{ \left(\sum_p A^{c,u}_p u_p \right) - b^u_c \right\}^2 + \left\{ \left(\sum_p A^{c,v}_p v_p \right) - b^u_c \right\}^2 , \quad (3)$$

where the A^i_j and b^i_j are given coefficients *when one parameter, u or v, is assumed constant*. The smoothing on u and v are interleaved, which means that the values of $A^i_{c,u}$ depend on v, and vice-versa.

This generalized roughness $R^\star$ reaches its minimum when its partial derivatives, $\frac{\partial R^\star}{\partial u}$ and $\frac{\partial R^\star}{\partial v}$, are null for each pixel p. This can be expressed as a set of linear equations:

$$\begin{aligned} u_p &= \sum_{p' \in N(N(p))} a^u_{p,p'} u_{p'} + \alpha^u_{p,p'} u_{p'} \\ v_p &= \sum_{p' \in N(N(p))} a^v_{p,p'} v_{p'} + \alpha^v_{p,p'} v_{p'}, \end{aligned} \quad (4)$$

where α^u_j are coefficients that depend on v, and α^v_j depend on u.

This method is, however, too slow to optimize over an entire layer in our interactive context. Moreover, it requires boundary conditions. We therefore adapt it using two strategies: a flood-fill approach and a simple but effective initialization.

4.2 Flood-fill parameterization

We adapt the discrete smooth interpolation method in a "flood-fill" manner to optimize the parameterization around the current position of the clone brush. We compute the parameterization for only a subset of pixels, called *active pixels*. This subset is expanded as time progresses and the user drags the brush. We describe the method for a single layer, however, it runs concurrently for both the source and destination layers. We interleave *optimization* steps, where coordinates are refined, and *expansion* steps, where new pixels are declared active and initialized. We moreover *freeze* the coordinates of already-brushed pixels.

To initialize the process, we use the first point clicked by the user as a seed, assign it the coordinates $(0,0)$, and set the gradient of u, $\vec{\nabla} u$ orthogonal to the vertical direction and to the pixel normal. $\vec{\nabla} v$ is then orthogonal to $\vec{\nabla} u$.

The set of pixels at the boundary of the active region is called the *active front*. More formally, a pixel is declared in the active front if it is active and if one of its 4-neighbors has an inactive 4-neighbor (Fig. 9(c)). Intuitively, the active front corresponds to pixels that lack neighbors necessary to compute smoothing in Eqs. (2) and (4). Active pixels not in the active front are said to be *fully active*.

Optimization

The optimization steps follow the lines of the original discrete smooth interpolation. Optimizations on u and v are interleaved, and active pixels are treated as vertices of a mesh and smoothed accordingly using linear operations.

The active front requires special care. Due to the absence of some neighbors, Eq. (4) cannot be directly used. If these neighbors are simply discarded from the formula, the parameterization will "shrink," because the roughness term R from Eq. (1) is no longer balanced. We thus only optimize the gradient constraints for these pixels and omit the mapping of the roughness term (the a^i_j in Eq. (4)). A good initial value for the active-front pixels is then the key to the stability of the process.

Expansion and initialization

An expansion step extends the active region by one pixel in the direction of the current mouse location. The active front is accordingly updated, and each new active pixel receives initial coordinate values. This is done according to its active neighbors, by using a local planar approximation of the geometry. For each neighbor, the coordinates (u',v') of the new pixel are computed using the current gradients of an active neighbor, $\vec{\nabla} u$ and $\vec{\nabla} v$, and the object-space vector $\vec{d}$ between the two pixels (Fig. 9(c)):

$$(u',v') = (u + \vec{d} \cdot \vec{\nabla} u, v + \vec{d} \cdot \vec{\nabla} v).$$

(a) (b) (c)

Figure 9: (a) Classical clone brushing. The user first clicks on i_0 to specify the source region and then paint brushes starting at b_0, which assigns the translation. b is then a simple copy of i. (b) Perspective-corrected clone brushing. The column geometry has been changed to a cylinder, and the carpet has been removed and clone brushed onto the ceiling. (c) Flood-fill parameterization. Fully active pixels are in red. The active front is in yellow. The green pixel is set active, and its initial (u', v') parameters are computed using the gradient of its active neighbors.

The average of the values computed from the active neighbors is used. This formula results in an optimal initial value, provided the geometry is planar.

Freezing and clone brushing

The parameterization proceeds as the user interactively clone brushes. It must be faster than the speed of the brush to ensure smooth interaction. In practice, we have found that subsampling the layer was necessary in order to obtain real-time feedback. We compute (u,v) values every 4×4 pixels and interpolate bilinearly. This process does not take into account the local bumps in the geometry, but fits the global shape.

As soon as a pixel has been clone brushed, its (u,v) coordinates must be *frozen* to avoid artifacts that would occur if the same pixel were re-used with different coordinate values due to subsequent optimization iterations.

Clone brushing a destination pixel with coordinate (u,v) involves inverting the mapping of the source image. Note that in the general case, no pixel will have the exact $(u,v,)$ coordinates. We thus use the four pixels with the nearest coordinates and perform bilinear interpolation. To find these pixels, we use a marching method. Since a brush corresponds to a set of contiguous pixels, we only need to compute a seed value and march from it along the gradient to find the inverse mapping of subsequent pixels.

Our optimization is clearly not as accurate as the method of Levy and Mallet [LM98, Mal89]. However, it provides an exact solution in the case of planar geometry and has worked well in practice for curved geometry. This is because our case is simpler than the general mesh-parameterization problem. Our data is a height field transformed by a perspective matrix, which greatly decreases potential distortions. Moreover, our layers are segmented by the user into different spatial objects that prevent strong discontinuities.

5 Texture-illuminance decoupling filter

We now present a filter that factors the image into a texture component and an illumination component. This is useful both for relighting and clone brushing, since the decoupled texture channel has a uniform level of illumination.

Most previous relighting work relies on a light transport simulation to remove the effect of existing lighting [FGR93, DRB97, Deb98, YDMH99, LFD+99, LDR00]. Loscos et al. use texture synthesis to remove artifacts along shadow boundaries, but still require an initial physical simulation [LDR00]. In contrast, our approach is not physically based. It is an image-processing filter that removes lighting effects from uniformly textured objects.

A related approach was introduced by Nayar and Bolle [NB93]. Our approach differs from theirs in that they deal with non-textured regions and focus on the segmentation and computation of reflectance ratios, while we deal with texture extraction.

5.1 Large- and small-scale feature separation

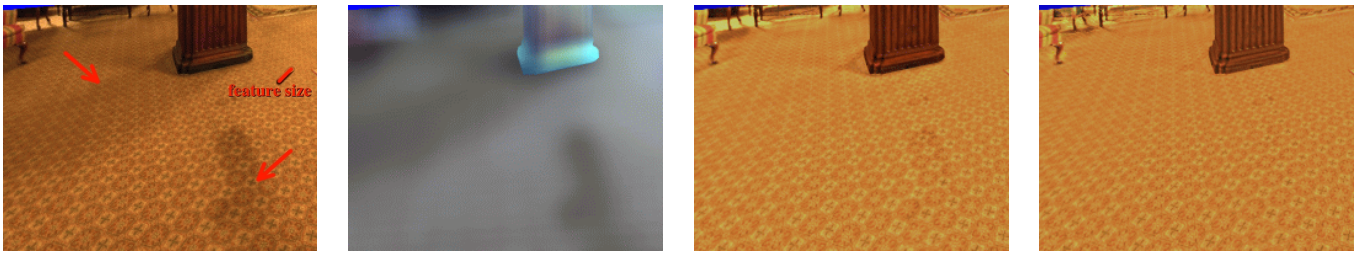
We make the following simple assumption: Large-scale luminance variations are due to the lighting, while small-scale details are due to the texture. In practice, this means that large stains will be treated as illuminance variations (which is actually desirable in most practical cases), while shadows of small objects will not be handled correctly. Small detailed shadows are the main limitation of our technique.

We have developed a non-linear filter that factors an image into a texture channel and an illuminance channel respecting the above assumption. We do not claim that these are the true texture and illuminance, but we will use these terms for simplicity. This problem is related to image denoising, but the "noise" in this case is the texture information that we want to retrieve.

To begin, the user specifies a *feature size* of the texture by dragging a line segment over a pattern. The basic idea is to blur the image with a low-pass Gaussian filter G, specified by the feature size (in practice we use $\sigma_{spatial} = feature\ size$). If I_0 is the input image, and p and p' are pixel locations, we have:

$$I_1(p) = \frac{\Sigma_{p'} G(p,p',\sigma_{spatial}) I_0(p')}{\Sigma_{p'} G(p,p',\sigma_{spatial})}. \quad (5)$$

Only large-scale illuminance variations remain. We moreover assume that the average color comes from the texture component, so we divide the illuminance obtained by the normalized average color value. We then divide the initial image by this blurred version to compute a uniform texture component (Fig. 10(b) and (c)).



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

Figure 10: Texture-illuminance decoupling. (a) Input image. (b) Initial illuminance estimation using simple Gaussian filtering. (c) Initial texture estimation, note the artifacts corresponding to shadow boundaries. (d) Texture computed using bilateral filtering.

This approach works well for slowly varying illumination, but fails at shadow boundaries, where haloing effects occur (see Fig. 10(c) and 11). This simply means that shadow boundaries introduce frequencies that are in the range of the feature size, and that they are treated as texture frequencies by the Gaussian blur. In addition, texture foreshortening needs to be treated to make consistent use of the feature size.

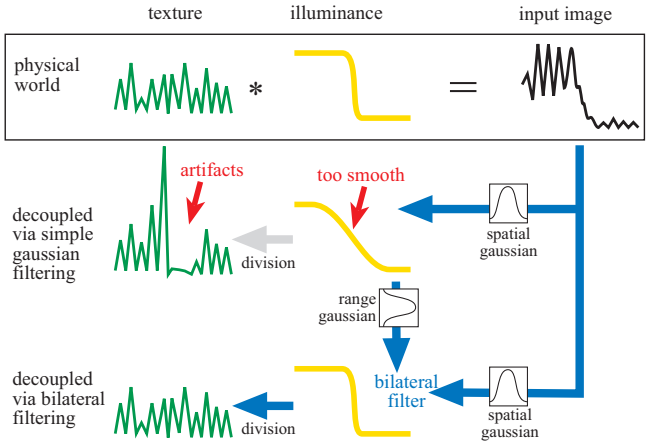


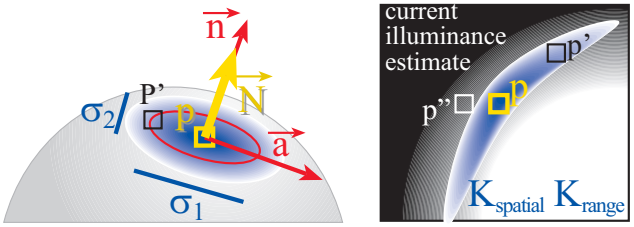
Figure 11: Texture-illuminance decoupling in 1D. The example exhibits sharp illumination variations, which cannot be captured using simple Gaussian filtering. This first pass is, however, used for the bilateral filtering pass, to average only pixels with similar illumination, due to the use of an additional range Gaussian.

5.2 Depth correction

Due to foreshortening and surface orientation, the feature size is not constant in image space. We therefore use the depth channel to compensate for foreshortening, and normals for orientation. The user specifies feature size at a reference pixel p_{ref}. For other pixels, the spatial kernel is scaled by $\frac{z_{ref}}{z}$. To compensate for orientation, we use a local planar approximation of the surface: Gaussian ellipses oriented along the scene surfaces (Fig. 12(a)).

Let $\vec{N}$ be the unit normal and $\vec{E}$ the unit viewing direction. The small axis of the ellipse is along $\vec{n}$, the unit image projection of $\vec{N}$. We note $\vec{a}$ the long axis of the ellipse, which is a unit vector orthogonal to $\vec{n}$. The small/large ratio $\frac{\sigma_2}{\sigma_1}$ is given by the dot product $\vec{N}\cdot\vec{E}$, where $\sigma_1 = \frac{z_{ref}}{z(p)}\sigma_{spatial}$ as described above. We then have:

$$K_{spatial}(p', p, \sigma_{spatial}) = G(\vec{pp'}\cdot\vec{n}, \sigma_2)\, G(\vec{pp'}\cdot\vec{a}, \sigma_1). \quad (6)$$



(a) (b)

Figure 12: (a) Orientation correction of the spatial Gaussian. (b) The kernel filter for pixel P is the product of the spatial and range Gaussians: The weight of point p'' is lower than the weight of p' although it is closer, because the estimated illuminance of p' is more similar to that of p.

5.3 Bilateral filtering

To handle discontinuities, we use a non-linear edge-preserving filter. Anisotropic diffusion could be an option [PM90], but since we have information about the feature size, we prefer the use of bilateral filtering [TM98], which is more controllable. The principle is to use an additional Gaussian in the range domain to average only pixels of similar intensity. The total kernel is the product of the spatial Gaussian (Eq. (5)) and this range Gaussian (Fig. 12(b)).

We adapt this method to our particular case. It is iterative, and the current estimate of the illuminance is used to drive the range Gaussian:

$$I_{i+1}(P) = \frac{\Sigma_{p'} K(p', p) I_0(p')}{\Sigma_{p'} K(p', p)}$$

$$K(p', p) = K_{spatial}(p', p, \sigma_{spatial}) G_{range}(I_i(p), I_i(p'), \sigma_{range}).$$

The main difference between this approach and standard bilateral filtering is that we always filter the initial image I_0. Unlike denoising approaches, we are interested in factoring the initial image, not in removing noise. Because the kernel averages only pixels of similar estimated illuminance, the filter captures shadow boundaries (Fig. 10(d) and 11). The process converges quickly, and we use I_3 as our final illuminance estimate. The only hard-coded parameter is the variance of the range Gaussian. In practice, we have found that $\sigma_{range} = 0.01 max(I_1)$ gives good results.

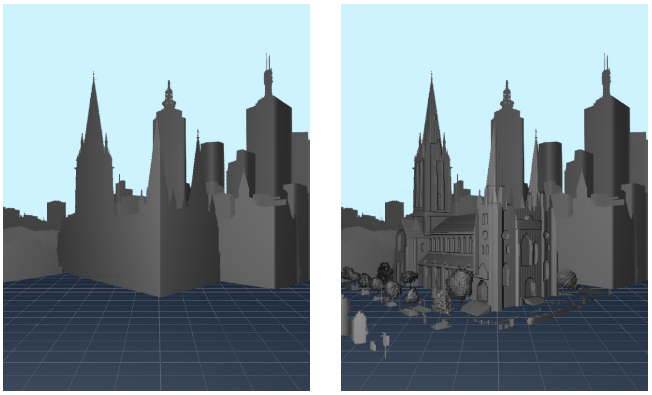
This filter is very effective on high-dynamic range images [DM97]. For 24-bit images, it works best if the illumination does not vary too dramatically. Otherwise, very dark or very bright regions can lead to artifacts, because the texture information has been destroyed by color quantization. In this case, texture synthesis is the only solution to resynthesize the lost information.

6 Implementation and Results

Our system has been implemented on SGI workstations using the QT GUI library and the graphics API OpenGL. The accompanying video demonstrates a variety of results obtained with the system. In this section, we first describe in detail how the church example was acquired and then describe other examples and editing operations.

6.1 Church example

For the church example (Fig. 1), we used a single 24-bit 1000x1280 scanned input photograph. The most time-consuming part of the acquisition was the manual segmentation of the input image into layers. Because our system uses only one input photograph, the area behind the church was manually clone brushed (Fig. 1(b)). The segmentation and clone brushing took about 10 hours. 52 different layers were extracted: Each tree is represented as a separate layer, and the church itself is decomposed into 13 layers for easier depth assignment and to allow for occlusion and parallax effects (Fig. 1(a)).



(a) (b)

Figure 13: St Paul's Cathedral acquired geometry. (a) Coarse depth after the use of the vertical tool. (b) Refined depth. Note the chiseling on the façades and the shapes of the trees, obtained using a combination of level sets and depth-from-rgb.

Depth acquisition took an additional three hours. We first defined the ground plane and camera parameters using the method by Liebowitz et al. [LCZ99]. Three orthogonal pairs of parallel lines were specified. Each layer was given a coarse billboard depth by utilizing the automatic vertical plane tool. For those layers without a clear point of contact with the ground, we used a user-specified vertical plane. Fig. 13(a) shows this initial coarse depth. We then used the level-set method to provide the trees and bushes with a bulgy appearance, which we refined using the depth-from-rgb tool. The depths for the cars and street furniture were refined with the box tool.

The church depth was acquired using several tools. The coarse depth from the vertical tool provided a starting point. We used the push/pull tool to add relief to the façade windows and buttresses. We then used the pyramid tool for the tower and turrets, and the plane tool for the roofs (Fig. 13(b)).

6.2 Editing and other examples

Various editing operations are possible including painting, filtering, and clone brushing. The ability to paint from different viewpoints makes it easy to edit foreshortened surfaces and obtain correct perspective. Copy-pasting, 3D scaling, translation, and rotation of objects are also possible. The range of rotation, like changes in viewpoint, are ultimately limited by the geometry of the scene and disocclusion artifacts.

We also demonstrate relighting applications. Texture and illuminance are decoupled using our new filter. Then, the illuminance channel is modified and multiplied by the texture channel to obtain the new image. The illuminance can be edited either by specifying 3D light sources, or by directly painting light on the channel. The latter solution often provides simpler interaction. This is due to the difficulty of specifying 3D positions and anticipating the resulting lighting effects. Sketching approaches, e.g. [PRJ97] could be helpful in this context.

We also use cubical panoramas as input, representing them as a collection of six images (Fig. 14). This provides a dramatic immersive experience and could greatly improve techniques such as Quicktime VR [Che95] at a reasonable cost.

7 Conclusion and Future Work

We have presented an image-based modeling and editing system that takes a single photo as input, and allows a user to build a representation consisting of layers of images with depth. Interactive views and a variety of editing operations permit the definition and manipulation of image-based scenes. The system has been demonstrated on several examples, including relighting.

Future work includes the merging of this approach with 3D painting programs, or with alternative image-editing approaches [EG01]. Many other techniques could be incorporated into our system. For example, depth assignment could benefit from shape from shading or from multiple images via stereo correspondence and epipolar geometry. In addition, handling multiple images of the same portion of a scene would permit the inclusion of view-dependent effects.

We believe that our clone brushing and texture-illuminance decoupling tools have application beyond the scope of this system. For example, a simpler version of clone brushing could be adapted to the 2D case for simple configurations. Our decoupling filter could be useful in a variety of contexts, including enabling classical image-based modeling to retrieve uniform texture or to preprocess the input of texture generation algorithms.

Acknowledgments

We would like to thank Aparna Das, Rujira Hongladaromp, and Sini Kamppari for experimenting with the system, providing feedback, and working on the examples. Thanks to Steven Gortler, George Drettakis, Nicolas Tsingos, Leonard McMillan, and Neel Master for encouragement and helpful discussions. Barry Webb and Associates, Lighting and Technology Consultants in Sydney, Australia, kindly provided the photograph of St. Paul's Cathedral in Melbourne. This work was supported by an NSF CISE Research Infrastructure Award (EIA-9802220) and a gift from Pixar Animation Studios.

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

Figure 14: Panoramic view of a hotel lobby. (b) is the original viewpoint, and (a),(c) are synthetic viewpoints. (d) visualizes the representation from a birds-eye view. The red arrow shows the original acquisition point and direction of the panorama. Although the geometry is coarse, the immersive experience within the room is very convincing.

References

[BV99] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. *Proc. of SIGGRAPH*, 1999.

[Can] Canoma. http://www.canoma.com.

[Che95] E. Chen. Quicktime VR - an image-based approach to virtual environment navigation. *Proc. of SIGGRAPH*, 1995.

[CW93] E. Chen and L. Williams. View interpolation for image synthesis. In *Proc. of SIGGRAPH*, 1993.

[Deb98] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proc. of SIGGRAPH*, 1998.

[DM97] P. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. *Proc. of SIGGRAPH*, 1997.

[DRB97] G. Drettakis, L. Robert, and S. Bougnoux. Interactive common illumination for computer augmented reality. *Eurographics Rendering Workshop*, 1997.

[DTM96] P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proc. of SIGGRAPH 96*, 1996.

[EG01] J. Elder and R. Goldberg. Image editing in the contour domain. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(3), 2001.

[FGR93] A. Fournier, A. Gunawan, and C. Romanzin. Common illumination between real and computer generated scenes. *Graphics Interface*, 1993.

[FLR+95] O. Faugeras, S. Laveau, L. Robert, G. Csurka, and C. Zeller. 3-d reconstruction of urban scenes from sequences of images. In A. Gruen, O. Kuebler, and P. Agouris, editors, *Automatic Extraction of Man-Made Objects from Aerial and Space Images*. Birkhauser, 1995.

[GDCV98] J. Gomes, L. Darsa, B. Costa, and L. Velho. *Warping And Morphing Of Graphical Objects*. Morgan Kaufman, 1998.

[GGSC96] S.. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Proc. of SIGGRAPH*, 1996.

[HAA97] Y. Horry, K. Anjyo, and K. Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *Proc. of SIGGRAPH 97*, 1997.

[HH90] P. Hanrahan and P. Haeberli. Direct wysiwyg painting and texturing on 3d shapes. *Proc. of SIGGRAPH*, 1990.

[IMT99] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3D freeform design. In *Siggraph*, Los Angeles, 1999.

[Kan98] S. Kang. Depth painting for image-based rendering applications. Tech. report, CRL, Compaq Cambridge Research Lab, 1998. http://www.research.microsoft.com/Users/sbkang/publications/index.html.

[LCZ99] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In *Proc. of Eurographics*, 1999.

[LDR00] C. Loscos, G. Drettakis, and L. Robert. Interactive virtual relighting of real scenes. *IEEE Trans. on Visualization and Computer Graphics*, 6(3), 2000.

[LF94] S. Laveau and O. Faugeras. 3-D scene representation as a collection of images and fundamental matrices. In *Proc. of 12th Int. Conf. on Pattern Recognition*, volume 1, pages 689–691, 1994.

[LFD+99] C. Loscos, M.C. Frasson, G. Drettakis, B. Walter, X. Granier, and P. Poulin. Interactive virtual relighting and remodeling of real scenes. *Eurographics Rendering Workshop*, 1999.

[LH96] M. Levoy and P. Hanrahan. Light field rendering. In *Proc. of SIGGRAPH*, 1996.

[LM98] B. Lévy and JL Mallet. Non-distorted texture mapping for sheared triangulated meshes. In *Proc. of SIGGRAPH*, 1998.

[Mal89] JL Mallet. Discrete smooth interpolation. *ACM Trans. on Graphics*, 8(2):121–144, 1989.

[MB95] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proc. of SIGGRAPH*, 1995.

[McM97] L. McMillan. *An Image-based Approach to Three-Dimensional Computer Graphics*. PhD thesis, U. of North Carolina, Chapel Hill, 1997.

[Met] MetaCreations. http://www.metacreations.com.

[MMB97] W. Mark, L. McMillan, and G. Bishop. Post-rendering 3D warping. In *ACM Symp. on Interactive 3D Graphics*, 1997.

[NB93] S. K. Nayar and R. M. Bolle. Computing reflectance ratios from an image. *Pattern recognition*, 7, 1993.

[Pal99] S. Palmer. *Vision Science : Photons to Phenomenology*. MIT Press, 1999.

[Pho] Photomodeler. http://www.photomodeler.com.

[PM90] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.

[POF98] P. Poulin, M. Ouimet, and M.C. Frasson. Interactively modeling with photogrammetry. In *Eurographics Workshop on Rendering*, 1998.

[PRJ97] P. Poulin, K. Ratib, and M. Jacques. Sketching shadows and highlights to position lights. In *Proc. of Computer Graphics International 97*, 1997.

[PSVF92] W. Press, S.Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes*. Cambridge Univ. Pr., 2nd edition, 1992.

[Rea] Realviz. Image modeler. http://www.realviz.com.

[SD95] F. Sillion and G. Drettakis. Feature-based control of visibility error: A multi-resolution clustering algorithm for global illumination. In *Proc. SIGGRAPH*, 1995.

[SGHS98] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *Proc. of SIGGRAPH*, 1998.

[SK98] S. Seitz and K. Kutulakos. Plenoptic image editing. In *Proc. 5th Int. Conf.on Computer Vision*, 1998.

[SKvW+92] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadows and lighting effects using texture mapping. *Proc. of SIGGRAPH*, 1992.

[TM98] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *IEEE Int. Conf. on Computer Vision*, 1998.

[WA94] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Trans. on Image Processing*, 3(5):625–638, 1994.

[Wil98] L. Williams. Image jets, level sets and silhouettes. Workshop on Image-Based Modeling and Renderingt, http://www-graphics.stanford.edu/workshops/ibr98/, March 1998.

[YDMH99] Y. Yu, P. Debevec, J. Malik, and T. Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. *Proc. of SIGGRAPH*, 1999.

Fast Bilateral Filtering for the Display of High-Dynamic-Range Images

Frédo Durand and Julie Dorsey
Laboratory for Computer Science, Massachusetts Institute of Technology

Abstract

We present a new technique for the display of high-dynamic-range images, which reduces the contrast while preserving detail. It is based on a two-scale decomposition of the image into a base layer, encoding large-scale variations, and a detail layer. Only the base layer has its contrast reduced, thereby preserving detail. The base layer is obtained using an edge-preserving filter called the *bilateral filter*. This is a non-linear filter, where the weight of each pixel is computed using a Gaussian in the spatial domain multiplied by an influence function in the intensity domain that decreases the weight of pixels with large intensity differences. We express bilateral filtering in the framework of robust statistics and show how it relates to anisotropic diffusion. We then accelerate bilateral filtering by using a piecewise-linear approximation in the intensity domain and appropriate subsampling. This results in a speed-up of two orders of magnitude. The method is fast and requires no parameter setting.

CR Categories: I.3.3 [Computer Graphics]: Picture/image generation—Display algorithms; I.4.1 [Image Processing and Computer Vision]: Enhancement—Digitization and image capture

Keywords: image processing, tone mapping, contrast reduction, edge-preserving filtering,weird maths

1 Introduction

As the availability of high-dynamic-range images grows due to advances in lighting simulation, e.g. [Ward 1994], multiple-exposure photography [Debevec and Malik 1997; Madden 1993] and new sensor technologies [Mitsunaga and Nayar 2000; Schechner and Nayar 2001; Yang et al. 1999], there is a growing demand to be able to display these images on low-dynamic-range media. Our visual system can cope with such high-contrast scenes because most of the adaptation mechanisms are *local* on the retina.

There is a tremendous need for contrast reduction in applications such as image-processing, medical imaging, realistic rendering, and digital photography. Consider photography for example. A major aspect of the art and craft concerns the management of contrast via e.g. exposure, lighting, printing, or local dodging and burning [Adams 1995; Rudman 2001]. In fact, poor management of light – under- or over-exposed areas, light behind the main character, etc. – is the single most-commonly-cited reason for rejecting photographs. This is why camera manufacturers have developed sophisticated exposure-metering systems. Unfortunately, exposure only operates via *global* contrast management – that is, it recenters the intensity window on the most relevant range. If the range of intensity is too large, the photo will contain under- and over-exposed areas (Fig. 1, rightmost part).

Figure 1: High-dynamic-range photography. No single global exposure can preserve both the colors of the sky and the details of the landscape, as shown on the rightmost images. In contrast, our spatially-varying display operator (large image) can bring out all details of the scene. Total clock time for this 700x480 image is 1.4 seconds on a 700Mhz PentiumIII. Radiance map courtesy of Paul Debevec, USC. [Debevec and Malik 1997]

Base Detail Color

Figure 2: Principle of our two-scale decomposition of the input intensity. Color is treated separately using simple ratios. Only the base scale has its contrast reduced.

Our work is motivated by the idea that the use of high-dynamic-range cameras and relevant display operators can address these issues. Digital photography has inherited many of the strengths of film photography. However it also has the potential to overcome its limitations. Ideally, the photography process should be decomposed into a measurement phase (with a high-dynamic-range output), and a post-process phase that, among other things, manages the contrast. This post-process could be automatic or user-controlled, as part of the camera or on a computer, but it should take advantage of the wide range of available intensity to perform appropriate contrast reduction.

In this paper, we introduce a fast and robust operator that takes a high-dynamic-range image as input, and compresses the contrast while preserving the details of the original image, as introduced by Tumblin [1999]. Our operator is based on a two-scale decomposition of the image into a *base* layer (large-scale features) and a *detail*

layer (Fig. 2). Only the base layer has its contrast reduced, thereby preserving the detail. In order to perform a fast decomposition into these two layers, and to avoid halo artifacts, we present a fast and robust edge-preserving filter.

1.1 Overview

The primary focus of this paper is the development of a fast and robust edge-preserving filter – that is, a filter that blurs the small variations of a signal (noise or texture detail) but preserves the large discontinuities (edges). Our application is unusual however, in that the noise (detail) is the important information in the signal and must therefore be preserved.

We build on *bilateral filtering*, a non-linear filter introduced by Tomasi et al. [1998]. It derives from Gaussian blur, but it prevents blurring across edges by decreasing the weight of pixels when the intensity difference is too large. As it is a fast alternative to the use of anisotropic diffusion, which has proven to be a valuable tool in a variety of areas of computer graphics, e.g. [McCool 1999; Desbrun et al. 2000], the potential applications of this technique extend beyond the scope of contrast reduction.

This paper makes the following contributions:

Bilateral filtering and robust statistics: We recast bilateral filtering in the framework of robust statistics, which is concerned with estimators that are insensitive to outliers. Bilateral filtering is an estimator that considers values across edges to be outliers. This allows us to provide a wide theoretical context for bilateral filtering, and to relate it to anisotropic diffusion.

Fast bilateral filtering: We present two acceleration techniques: we linearize bilateral filtering, which allows us to use FFT and fast convolution, and we downsample the key operations.

Uncertainty: We compute the uncertainty of the output of the filter, which permits the correction of doubtful values.

Contrast reduction: We use bilateral filtering for the display of high-dynamic-range images. The method is fast, stable, and requires no setting of parameters.

2 Review of local tone mapping

Tone mapping operators can be classified into *global* and *local* techniques [Tumblin 1999; Ferwerda 1998; DiCarlo and Wandell 2000]. Because they use the same mapping function for all pixels, most *global* techniques do not directly address contrast reduction. A limited solution is proposed by Schlick [1994] and Tumblin et al. [1999], who use S-shaped functions inspired from photography, thus preserving some details in the highlights and shadows. Unfortunately, contrast is severely reduced in these areas. Some authors propose to interactively vary the mapping according to the region of interest attended by the user [Tumblin et al. 1999], potentially using graphics hardware [Cohen et al. 2001].

A notable exception is the global histogram adjustment by Ward-Larson et al. [1997]. They disregard the empty portions of the histogram, which results in efficient contrast reduction. However, the limitations due to the global nature of the technique become obvious when the input exhibits a uniform histogram (see e.g. the example by DiCarlo and Wandell [2000]).

In contrast, *local* operators use a mapping that varies spatially depending on the neighborhood of a pixel. This exploits the fact that human vision is sensitive mainly to local contrast.

Most local tone-mapping techniques use a decomposition of the image into different layers or scales (with the exception of Socolinsky, who uses a variational technique [2000]). The contrast is reduced differently for each scale, and the final image is a recomposition of the various scales after contrast reduction. The major pitfall of local methods is the presence of haloing artifacts. When dealing with high-dynamic-range images, haloing issues become even more critical. In 8-bit images, the contrast at the edges is limited to roughly two orders of magnitude, which directly limits the strength of halos.

Chiu et al. vary a gain according to a low-pass version of the image [1993], which results in pronounced halos. Schlick had similar problems when he tried to vary his mapping spatially [1994]. Jobson et al. reduce halos by applying a similar technique at multiple scales [1997]. Pattanaik et al. use a multiscale decomposition of the image according to comprehensive psychophysically-derived filter banks [1998]. To date, this method seems to be the most faithful to human vision, however, it may still present halos.

DiCarlo et al. propose to use robust statistical estimators to improve current techniques [2000], although they do not provide a detailed description. Our method follows in the same spirit and focuses on the development of a fast and practical method.

Tumblin et al. [1999] propose an operator for synthetic images that takes advantage of the ability of the human visual system to decompose a scene into intrinsic "layers", such as reflectance and illumination [Barrow and Tenenbaum 1978]. Because vision is sensitive mainly to the reflectance layers, they reduce contrast only in the illumination layer. This technique is unfortunately applicable only when the characteristics of the 3D scene are known. As we will see, our work can be seen as an extension to photographs. Our two-scale decomposition is very related to the texture-illuminance decoupling technique by Oh et al. [2001].

Recently, Tumblin and Turk built on anisotropic diffusion to decompose an image using a new low-curvature image simplifier (LCIS) [Tumblin 1999; Tumblin and Turk 1999]. Their method can extract exquisite details from high-contrast images. Unfortunately, the solution of their partial differential equation is a slow iterative process. Moreover, the coefficients of their diffusion equation must be adapted to each image, which makes this method more difficult to use, and the extension to animated sequences unclear. We build upon a different edge-preserving filter that is easier to control and more amenable to acceleration. We will also deal with two problems mentioned by Tumblin et al.: the small remaining halos localized around the edges, and the need for a "leakage fixer" to completely stop diffusion at discontinuities.

3 Edge-preserving filtering

In this section, we review important edge-preserving-smoothing techniques, e.g. [Saint-Marc et al. 1991].

3.1 Anisotropic diffusion

Anisotropic diffusion [Perona and Malik 1990] is inspired by an interpretation of Gaussian blur as a heat conduction partial differential equation (PDE): $\frac{\partial I}{\partial t} = -\Delta I$. That is, the intensity I of each pixel is seen as heat and is propagated over time to its 4 neighbors according to the heat spatial variation.

Perona and Malik introduced an *edge-stopping* function g that varies the *conductance* according to the image gradient. This prevents heat flow across edges:

$$\frac{\partial I}{\partial t} = div[g(||\nabla I||)\,\nabla I]. \tag{1}$$

They propose two expressions for the edge-stopping function g(x):

$$g_1(x) = \frac{1}{1+\frac{x^2}{\sigma^2}} \quad \text{and} \quad g_2(x) = e^{-(x^2/\sigma^2)}, \tag{2}$$

where σ is a scale parameter in the intensity domain that specifies what gradient intensity should stop diffusion.

The discrete Perona-Malik diffusion equation governing the value I_s at pixel s is then

$$I_s^{t+1} = I_s^t + \frac{\lambda}{4} \sum_{p \in neighb_4(s)} g(I_p^t - I_s^t) \quad (I_p^t - I_s^t), \tag{3}$$

where t describes discrete time steps, and $neighb_4(s)$ is the 4-neighborhood of pixel s. λ is a scalar that determines the rate of diffusion.

Although anisotropic diffusion is a popular tool for edge-preserving filtering, its discrete diffusion nature makes it a slow process. Moreover, the results depend on the stopping time, since the diffusion converges to a uniform image.

3.2 Robust anisotropic diffusion

Black et al. [1998] recast anisotropic diffusion in the framework of robust statistics. Our analysis of bilateral filtering is inspired by their work. The field of robust statistics develops estimators that are robust to outliers or deviation to the theoretical distribution [Huber 1981; Hampel et al. 1986].

Black et al. [1998] show that anisotropic diffusion can be seen as the estimate of a value I_s at each pixel s that is an estimate of its 4-neighbors, which minimizes an energy over the whole image:

$$\min \sum_{s \in \Omega} \sum_{p \in neighb_4(s)} \rho(I_p - I_s), \tag{4}$$

where Ω is the whole image, and ρ is an error norm (e.g. quadratic). Eq. 4 can be solved by gradient descent for each pixel:

$$I_s^{t+1} = I_s^t + \frac{\lambda}{4} \sum_{p \in neighb_4(s)} \psi(I_p - I_s), \tag{5}$$

where ψ is the derivative of ρ, and t is a discrete time variable. ψ is proportional to the so-called *influence function* that characterizes the influence of a sample on the estimate.

For example, a least-square estimate is obtained by using $\rho(x) = x^2$, and the corresponding influence function is linear, thus resulting in the mean estimator (Fig. 4, left). As a result, values far from the mean have a considerable influence on the estimate. In contrast, an influence function such as the *Lorentzian* error norm, given in Fig. 3 and plotted in Fig. 4, gives much less weight to outliers and is therefore more robust. In the plot of ψ, we see that the influence function is *redescending* [Black et al. 1998; Huber 1981][1]. Robust norms and influence functions depend on a parameter σ that provides the notion of scale in the intensity domain, and controls where the influence function becomes redescending, and thus which values are considered outliers.

Black et al. note that Eq. 5 is similar to Eq. 3 governing anisotropic diffusion, and that by defining $g(x) = \psi(x)/x$, anisotropic diffusion is reduced to a robust estimator. They also show that the g_1 function proposed by Perona et al. is equivalent to the Lorentzian error norm plotted in Fig. 4 and given in Fig. 3.

This analogy allows them to discuss desirable properties of edge-stopping functions. In particular, they show that *Tukey's biweight function* (Fig. 3) yields more robust results, because it completely stops diffusion across edges: The influence of outliers is null, as shown in Fig. 5, as opposed to the Lorentzian error norm that slowly goes to zero towards infinity. This also solves the termination problem, since diffusion then converges to a *piecewise*-uniform image.

[1] Some authors reserve the term redescending for function that vanish after a certain value [Hampel et al. 1986].

Huber	Lorentz
$g_\sigma(x) = \begin{cases} \frac{1}{\sigma} & \lvert x\rvert \le \sigma \\ \frac{1}{\lvert x\rvert}, & \text{otherwise} \end{cases}$ σ	$g_\sigma(x) = \frac{2}{2+\frac{x^2}{\sigma^2}}$ $\sigma/\sqrt{2}$
Tukey	**Gauss**
$g_\sigma(x) = \begin{cases} \frac{1}{2}[1-(x/\sigma)^2]^2 & \lvert x\rvert \le \sigma \\ 0, & \text{otherwise} \end{cases}$ $\sigma * \sqrt{5}$	$g_\sigma(x) = e^{-\frac{x^2}{2\sigma^2}}$ σ

Figure 3: Robust edge-stopping functions. Note that ψ can be found by multiplying g by x, and ρ by integration of ψ. The value of σ has to be modified accordingly to use a consistent scale across estimators, as indicated below the Lorentz and Tukey functions.

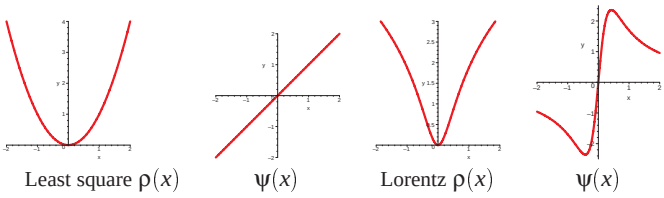


Figure 4: Least-square vs. Lorentzian error norm (after [Black et al. 1998]).

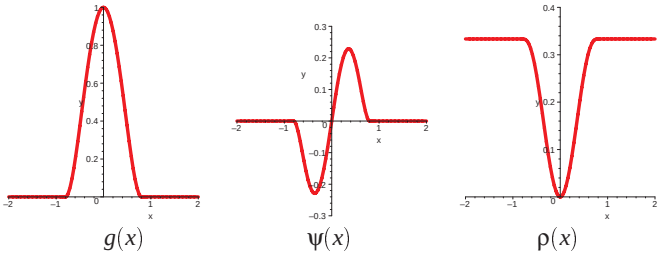


Figure 5: Tukey's biweight (after [Black et al. 1998]).

3.3 Bilateral filtering

Bilateral filtering was developed by Tomasi and Manduchi as an alternative to anisotropic diffusion [1998]. It is a non-linear filter where the output is a weighted average of the input. They start with standard Gaussian filtering with a spatial kernel f (Fig. 6). However, the weight of a pixel depends also on a function g in the intensity domain, which decreases the weight of pixels with large intensity differences. We note that g is an edge-stopping function similar to that of Perona et al. [1990]. The output of the bilateral filter for a pixel s is then:

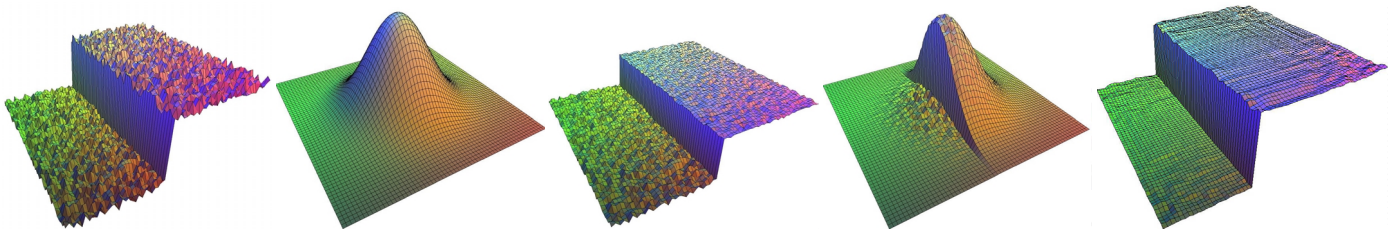
$$J_s = \frac{1}{k(s)} \sum_{p \in \Omega} f(p-s) \quad g(I_p - I_s) \quad I_p, \tag{6}$$

where $k(s)$ is a normalization term:

$$k(s) = \sum_{p \in \Omega} f(p-s) \quad g(I_p - I_s). \tag{7}$$

In practice, they use a Gaussian for f in the spatial domain, and a Gaussian for g in the intensity domain. Therefore, the value at a pixel s is influenced mainly by pixel that are close spatially and that have a similar intensity (Fig. 6). This is easy to extend to color images, and any metric g on pixels can be used (e.g. CIE-LAB).

Barash proposes a link between anisotropic diffusion and bilateral filtering [2001]. He uses an extended definition of intensity that includes spatial coordinates. This permits the extension of bilateral filtering to perform feature enhancement. Unfortunately,



input | spatial kernel f | influence g in the intensity domain for the central pixel | weight $f \times g$ for the central pixel | output

Figure 6: Bilateral filtering. Colors are used only to convey shape.

the extended definition of intensity is not quite natural. Elad also discusses the relation between bilateral filtering, anisotropic diffusion, and robust statistics, but he address the question from a linear-algebra point of view [to appear]. In this paper, we propose a different unified viewpoint based on robust statistics that extends the work by Black et al. [1998].

4 Edge-preserving smoothing as robust statistical estimation

In their paper, Tomasi et al. only outlined the principle of bilateral filters, and they then focused on the results obtained using two Gaussians. In this section, we provide a principled study of the properties of this family of filters. In particular, we show that bilateral filtering is a robust statistical estimator, which allows us to put empirical results into a wider theoretical context.

4.1 A unified viewpoint on bilateral filtering and 0-order anisotropic diffusion

In order to establish a link to bilateral filtering, we present a different interpretation of discrete anisotropic filtering. In Eq. 3, $I_p^t - I_s^t$ is used as the derivative of I^t in one direction. However, this can also be seen simply as the *0-order* difference between the two pixel intensities. The edge-stopping function can thus be seen as preventing diffusion between pixels with large intensity differences. The two formulations are equivalent from a practical standpoint, but Black et al.'s variational interpretation [1998] is more faithful to Perona and Malik's diffusion analogy, while our 0-order interpretation is more natural in terms of robust statistics.

In particular, we can extend the 0-order anisotropic diffusion to a larger spatial support:

$$I_s^{t+1} = I_s^t + \lambda \sum_{p \in \Omega} f(p-s)\, g(I_p^t - I_s^t)\ \ (I_p^t - I_s^t), \tag{8}$$

where f is a spatial weighting function (typically a Gaussian), Ω is the whole image,and t is still a discrete time variable. The anisotropic diffusion of Perona et al., which we now call *local diffusion*, corresponds to an f that is zero except at the 4 neighbors. Eq. 8 defines a robust statistical estimator of the class of *M-estimators* (generalized maximum likelihood estimator) [Hampel et al. 1986; Huber 1981].

In the case where the conductance g is uniform (isotropic filtering) and where f is a Gaussian, Eq. 8 performs a Gaussian blur for each iteration, which is equivalent to several iterations of the heat-flow simulation. It can thus be seen as a way to trade the number of iterations for a larger spatial support. However, in the case of anisotropic diffusion, it has the additional property of propagating heat across ridges. Indeed, if the image is white with a black line in the middle, local anisotropic diffusion does not propagate energy between the two connected components, while extended diffusion does. Depending on the application, this property will be either beneficial or deleterious. In the case of tone mapping, for example, the notion of connectedness is not important, as only spatial neighborhoods matter.

We now come to the robust statistical interpretation of bilateral filtering. Eq. 6 defines an estimator based on a weighted average of the data. It is therefore a *W-estimator* [Hampel et al. 1986]. The iterative formulation is an instance of *iteratively reweighted least squares*. This taxonomy is extremely important because it was shown that M-estimators and W-estimators are essentially equivalent and solve the same energy minimization problem [Hampel et al. 1986], p. 116:

$$\min \sum_{s \in \Omega} \sum_{p \in \Omega} \rho(I_s - I_p) \tag{9}$$

or for each pixel s:

$$\sum_{p \in \Omega} \psi(I_s - I_p) = 0, \tag{10}$$

where ψ is the derivative of ρ. As shown by Black et al. [1998] for anisotropic diffusion, and as is true also for bilateral filtering, it suffices to define $\psi(x) = g(x) * x$ to find the original formulations. In fact the second edge-stopping function g_2 in Eq. 2 defined by Perona et al. [1990] corresponds to the Gaussian influence function used for bilateral filtering [Tomasi and Manduchi 1998]. As a consequence of this unified viewpoint, all the studies on edge-stopping functions for anisotropic diffusion can be applied to bilateral filtering.

Eqs. 9 and 10 are not strictly equivalent because of local minima of the energy. Depending on the application, this can be desirable or undesirable. In the former case, the use of a very robust estimator, such as the median, to initialize an iterative process is recommended. In the case of tone mapping or texture-illuminance decoupling, however, we want to find the local minimum closest to the initial pixel value.

It was noted by Tomasi et al. [1998] that bilateral filtering usually requires only one iteration. Hence it belongs to the class of *one-step W-estimators*, or *w-estimators*, which have been shown to be particularly efficient. The existence of local minima is however a very important issue, and the use of an initial median estimator is highly recommended. In contrast, Oh. et al. use a simple Gaussian blur [2001], which deserves further study.

Now that we have shown that 0-order anisotropic diffusion and bilateral filtering belong to the same family of estimators, we can compare them. They both respect *causality*: No maximum or minimum can be created, only removed. However, anisotropic diffusion is *adiabatic* (energy-preserving), while bilateral filtering is not. To see this, consider the energy exchange between two pixels p and s. In the diffusion case, the energy $\lambda f(p-s)g(I_p^t - I_s^t)(I_p^t - I_s^t)$ flowing from p to s is the opposite of the energy from s to p because the expression is symmetric (provided that g and f are symmetric). In contrast, in bilateral filtering, the normalization factor $1/k$

is different for the two pixels, resulting in an asymmetric energy flow. Energy preservation can be crucial for some applications, e.g. [Rushmeier and Ward 1994], but it is not for tone mapping or reflectance extraction.

In contrast to anisotropic diffusion, bilateral filtering does not rely on shock formation, so it is not prone to stairstepping artifacts. The output of bilateral filtering on a gradient input is smooth. This point is mostly due to the non-iterative nature of the filter and deserves further exploration.

4.2 Robust estimators

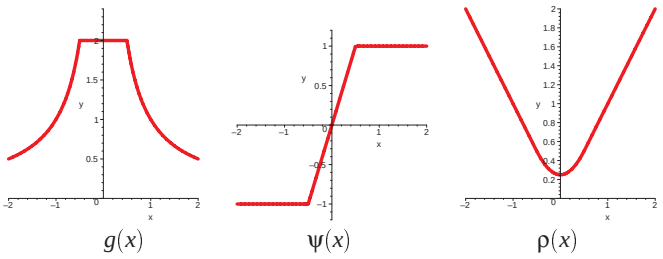


Figure 7: Huber's minimax (after [Black et al. 1998]).

Fig. 8 plots a variety of robust influence functions, and their Formulas are given in Fig. 3. When the influence function is monotonic, there is no local minimum problem, and estimators always converge to a global maximum. Most robust estimators have a shape as shown on the left: The function increases, then decreases, and potentially goes to zero if it has a finite rejection point.

These plots can be very helpful in understanding how an estimator deals with outliers. For example, we can see that the Huber minimax gives constant influence to outliers, and that the Lorentz estimator gives them more importance than, say, the Gaussian estimator. The Tukey biweight is the only purely *redescending* function we show. Outliers are thus completely ignored.

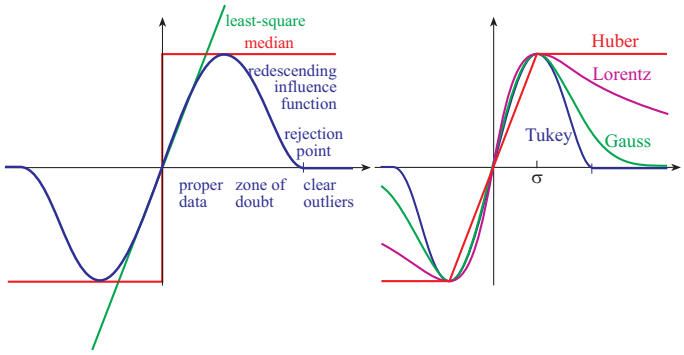


Figure 8: Comparison of influence functions.

We anticipate the results of our technique and show in Fig. 9 the output of a robust bilateral filter using these different ψ functions (or their g equivalent in Eq. 6). We can see that larger influences of outliers result in estimates that are more blurred and further from the input pixels. In what follows, we use the Gaussian or Tukey influence function, because they are more robust to outliers and better preserve edges.

5 Efficient Bilateral Filtering

Now that we have provided a theoretical framework for bilateral filtering, we will next deal with its speed. A direct implementation of

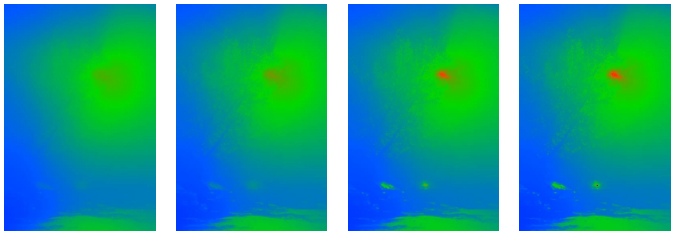


Figure 9: Comparison of the 4 estimators for the log of intensity of the foggy scene of Fig 15. The false-colored output is normalized to the log of the min and max of the input.

bilateral filtering might require $O(n^2)$ time, where n is the number of pixels in the image. In this section, we dramatically accelerate bilateral filtering using two strategies: a piecewise-linear approximation in the intensity domain, and a sub-sampling in the spatial domain. We then present a technique that detects and fixes pixels where the bilateral filter cannot obtain a good estimate due to lack of data.

5.1 Piecewise-linear bilateral filtering

A convolution such as Gaussian filtering can be greatly accelerated using Fast Fourier Transform. A $O(n^2)$ convolution in the primal becomes a $O(n)$ multiplication in the frequency domain. Since the discrete FFT and its inverse have cost $O(n \log n)$, there is a gain of one order of magnitude.

Unfortunately, this strategy cannot be applied directly to bilateral filtering, because it is not a convolution: The filter is signal-dependent because of the edge-stopping function $g(I_p - I_s)$. However consider Eq. 6 *for a fixed pixel* s. It is equivalent to the convolution of the function $H^{I_s} : p \to g(I_p - I_s) I_p$ by the kernel f. Similarly, the normalization factor k is the convolution of $G^{I_s} : p \to g(I_p - I_s)$ by f. That is, the only dependency on pixel s is the value I_s in g.

Our acceleration strategy is thus as follows: We discretize the set of possible signal intensities into `NB_SEGMENT` values $\{i^j\}$, and compute a linear filter for each such value:

$$\begin{aligned} J_s^j &= \frac{1}{k^j(s)} \sum_{p \in \Omega} f(p-s) \;\; g(I_p - i^j) \;\; I_p \\ &= \frac{1}{k^j(s)} \sum_{p \in \Omega} f(p-s) \;\; H_p^j \end{aligned} \tag{11}$$

and

$$\begin{aligned} k^j(s) &= \sum_{p \in \Omega} f(p-s) \;\; g(I_p - i^j) \\ &= \sum_{p \in \Omega} f(p-s) \;\; G^j(p). \end{aligned} \tag{12}$$

The final output of the filter for a pixel s is then a linear interpolation between the output J_s^j of the two closest values i^j of I_s. This corresponds to a piecewise-linear approximation of the original bilateral filter (note however that it is a linearization of the whole functional, not of the influence function). The pseudocode is given in Fig. 10.

Fig. 11 shows the speed-up we obtain depending on the size of the spatial kernel. Quickly, the piecewise-linear version outperforms the brute-force implementation, due to the use of FFT convolution. The formal analysis of error remains to be performed, but no artifact was noticeable for segments up to the size of the scale σ_r.

This could be further accelerated when the distribution of intensities is not uniform spatially. We can subdivide the image into sub-images, and if the difference between the max and min of the

PiecewiseBilateral
(Image I, spatial kernel f_{σ_s}, intensity influence g_{σ_r})
J=0 /* *set the output to zero* */
for j=0..NB_SEGMENTS
i^j= minI+j × (max(I)-min(I))/NB_SEGMENTS
$G^j=g_{\sigma_r}$ (I - i^j) /* evaluate g_{σ_r} at each pixel */
$K^j=G^j \otimes f_{\sigma_s}$ /* normalization factor */
$H^j=G^j \times$ I /* compute H for each pixel */
$H^{*j}=H^j \otimes f_{\sigma_s}$
$J^j=H^{*j}/K^j$ /* normalize */
J=J+J^j × InterpolationWeight(I, i^j)

Figure 10: Pseudo code of the piecewise-linear acceleration of bilateral filtering. Operations with upper cases such as $G^j=g_{\sigma_r}$ (I, i^j) denote computation on all pixels of the image. $\otimes$ denotes the convolution, while $\times$ is simply the per-pixel multiplication. InterpolationWeight is the "hat" interpolation weight for linear interpolation. In practice, we use NB_SEGMENT=(max(I)-min(I))/σ_r.

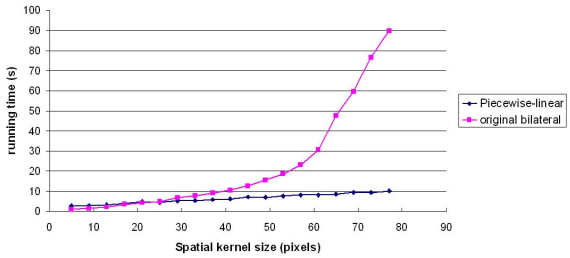


Figure 11: Speed-up of the piecewise-linear acceleration for 17 segments and a 576x768 image.

FastBilateral
(Image I, spatial kernel f_{σ_s}, intensity influence g_{σ_r}, downsampling factor z)
J=0 /**set the full-scale output to zero* */
I'=downsample (I, z)
$f'_{\sigma_s/z}$**=downsample (** f_{σ_s}**, z)**
for j=0..NB_SEGMENTS
i^j= minI+j × (max(I)-min(I))/NB_SEGMENTS
$G'^j=g_{\sigma_r}$ (I'-i^j) /* evaluate g_{σ_r} at each pixel */
$K'^j=G'^j \otimes f'_{\sigma_s/z}$ /* normalization factor */
$H'^j=G'^j \times$ I' /* compute H for each pixel */
$H'^{*j}=H'^j \otimes f'_{\sigma_s/z}$
$J'^j=H'^{*j}/K'^j$ /* normalize */
J^j**=upsample(**J'^j**, z)**
J=J+J^j × InterpolationWeight(I, i^j)

Figure 12: Pseudo code of the downsampled piecewise-linear acceleration of bilateral filtering. Parts at the full resolution are in green, while downsampled operations are in blue, and downsampled images are denoted with a prime.

intensity is more reduced in the sub-images than in the whole image, fewer segments can be used. This solution has however not been implemented yet.

5.2 Subsampling

To further accelerate bilateral filtering, we note that all operations in Fig. 10 except the final interpolation aim at low-pass filtering. We can thus safely use a downsampled version of the image with little quality loss. However, the final interpolation must be performed using the full-scale image, otherwise edges would not be respected, resulting in visible artifacts. Fig. 12 shows the new algorithm.

We use nearest-neighbor downsampling, because it does not modify the histogram. The acceleration we obtain is plotted in Fig. 13 for an example. While a formal study of error/acceleration remains to be done, we did not notice any visible artifact up to downsampling factor of 10 to 25. At this resolution, the cost of the upsampling and linear interpolation outweighs the filtering operations, and no further acceleration is gained by more aggressive downsampling.

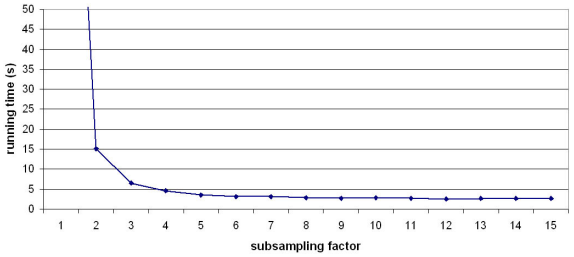


Figure 13: Speed-up due to downsampling for 17 segments and a 576x768 image. The value for the full-scale filtering is 173 sec.

5.3 Uncertainty

As noted by Tumblin et al. [Tumblin 1999; Tumblin and Turk 1999], edge-preserving contrast reduction can still encounter small halo artifacts for antialiased edges or due to flare around high-contrast edges. We noticed similar problems on some synthetic as well as real images. We propose an explanation in terms of signal/noise ratio. These small halos correspond to pixels where there is not enough information in the neighborhood to decouple the large-scale and the small-scale features. Indeed, the values at the edges span the whole range between the upper and the lower values, and there are very few pixels in the zone of proper data of the influence function. We thus compute a statistical estimator with very little data, and the variance is quite high.

Fortunately, bilateral filtering provides a direct measure of this uncertainty: The normalization factor k in Eq. 6 is the sum of the influence of each pixel. We can therefore use it to detect dubious pixels that need to be fixed. In practice, we use the log of this value because it better extracts uncertain pixels.

The fixing strategy we use is then simple. We compute a low-pass version $\tilde{J}$ of the output J of the bilateral filter, using a small Gaussian kernel (2 pixels in practice), and we assign to a pixel the value of a linear interpolation between J and $\tilde{J}$ depending on the log of the uncertainty k.

6 Contrast reduction

We now describe how bilateral filtering can be used for contrast reduction. We note that our method is not strictly a tone reproduction operator, in the sense of Tumblin and Rushmeier's [1993], since it does not attempt to imitate human vision.

Building on previous approaches, our contrast reduction is based on a multiscale decomposition e.g. [Jobson et al. 1997; Pattanaik et al. 1998; Tumblin and Turk 1999]. However, we only use a two-scale decomposition, where the "base" image is computed using bilateral filtering, and the detail layer is the division of the input intensity by the base layer. Fig. 2 illustrates the general approach. The base layer has its contrast reduced, while the magnitude of the detail layer is unchanged, thus preserving detail.

Following Tumblin et al. [Tumblin 1999; Tumblin and Turk 1999], we compress the range of the base layer using a scale factor in the log domain. We compute this scale factor such that the whole

range of the base layer is compressed to a user-controllable *base contrast*. In practice, a base contrast of 5 worked well for all our examples, but in some situations where lights sources are visible, one might want to vary this setting.

Our treatment of color is simple. We perform contrast reduction on the intensity of pixels and recompose color after contrast reduction [Schlick 1994; Tumblin 1999; Tumblin and Turk 1999]. We perform our calculations on the logs of pixel intensities, because pixel differences then correspond directly to contrast, and because it yields a more uniform treatment of the whole range.

Our approach is faithful to the original idea by Chiu et al. [1993], albeit using a robust filter instead of their low-pass filter. It can also be viewed as the decomposition of the image into intrinsic layers of reflectance and illuminance [Oh et al. 2001], followed by an appropriate contrast reduction of the illuminance (or base) layer [Tumblin et al. 1999].

For the filtering phase, we experimented with the various influence functions discussed in Section 4.2. As expected, the Huber minimax estimator decreases the strength of halos compared to standard Gaussian blur, but does not eliminate them. Moreover, the results vary with the size of the spatial kernel. The Lorentz function performed better, but only the Gaussian and Tukey's biweight were able to accurately decompose the image. With both functions, the scale σ_s of the spatial kernel had little influence on the result. This is important since it allows us to keep σ_s constant to a value of 2% of the image size.

The value $\sigma_r = 0.4$ performed consistently well for all our experiments. Again, this property is quite important because the user does not have to set a complex parameter. The significance of this value might come from two complementary origins, which are still areas of future research. First, it might be due to characteristics of the local sensitivity of the human visual system. Perhaps beyond this value, we notice no difference. Second, it might be related to the physical range of possible reflectance values, between a perfect reflector and a black material.

As a conclusion, the only user-controlled parameters of our method are the overall brightness and the base contrast. While the automatic values perform very well, we found it useful to provide these intuitive degrees of freedom to allow the user a control over the "look" of the image. The base contrast provides a very intuitive alternative to the contrast/brightness setting of image-editing software. It controls the overall appearance of the image, while still preserving the fine details.

6.1 Implementation and results

We have implemented our technique using a floating point representation of images, and the Intel image processing library for the convolutions. We have tested it on a variety of synthetic and real images, as shown in the color plates. All the examples reproduced in the paper use the Gaussian influence function, but the results with Tukey's biweight are not different. The technique is extremely fast, as can be seen in Fig. 14. We have tested it on an upsampled 10Mpixel image with contrast of more than 1:100,000, and the computation took only 6s on a 2GHz Pentium 4. In particular, due to our acceleration techniques, the running time grows sub-linearly. This is a dramatic speed-up compared to previous methods.

Our technique can address some of the most challenging photographic situations, such as interior lighting or sunset photos, and produces very compelling images. In our experiments, Tumblin and Turk's operator [1999] appears to better preserve fine details, while our technique better preserves the overall photorealistic appearance (Figs. 21 and 22).

Image	resolution	# segments	z	timing (s)
Grove D	710 * 480	15	4	0.33
Memorial	512 * 768	11	4	0.31
Hotel room	750 * 487	13	4	0.31
Vine	710 * 480	10	4	0.23
Fog	1130 * 751	12	8	0.45
Grove C	709 * 480	14	4	0.30
Window	2K*1.3K	10	16	2.73
Interior	2K*1.3K	19	16	2.19
Interior*2	2.6K * 4K	19	24	6.03

Figure 14: Results of our new technique. Timings on a 2GHz P4.

7 Discussion

This paper opens several avenues of future research related to edge-preserving filtering and contrast reduction. The unified viewpoint on bilateral filtering and anisotropic diffusion offers some interesting possibilities. The robust statistical framework we have introduced suggests the application of bilateral filtering to a variety of graphics areas where energy preservation is not a major concern.

The treatment of uncertainty deserves more attention. The correction scheme based on a Gaussian blur by a small kernel works well in the cases we have tested, but a more formal analysis is needed. Other approaches might involve the use of a different range scale σ_r.

In terms of contrast reduction, future work includes the development of a more principled fixing method for uncertain values, and the use of a more elaborate compression function for the base layer, e.g. [Tumblin et al. 1999; Larson et al. 1997]. White balance is an important issue for indoor scenes that also exhibit outdoor portions, as can be seen in Fig. 23. A strategy similar to Pattanaik et al.'s operator [Pattanaik et al. 1998] should be developed. The inclusion of perceptual aspects is a logical step. The main difficulty stems from the complex interaction between local adaptation and gaze movements. The extension to animated sequences is an exciting topic. Initial experiments are very encouraging.

Finally, contrast reduction is only one example of *pictorial techniques* to cope with the *limitations of the medium* [Durand 2002]. We believe that these techniques are crucial aspects of the digital photography and video revolution, and will facilitate the creation of effective and compelling pictures.

Acknowledgments

We would like to thank Byong Mok Oh for his help with the radiance maps and the bibliography; he and Ray Jones also provided crucial proofreading. Thanks to Paul Debevec and Jack Tumblin for allowing us to use their radiance maps. Thanks to the reviewers for their careful comments. This research was supported by NSF grants CCR-0072690 and EIA-9802220, and by a gift from Pixar Animation Studios.

References

ADAMS, A. 1995. *The Camera+The Negative+The Print*. Little Brown and Co.

BARASH, D. 2001. A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. *IEEE PAMI*. in press.

BARROW, H., AND TENENBAUM, J. 1978. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*. Academic Press, New York, 3–26.

BLACK, M., SAPIRO, G., MARIMONT, D., AND HEEGER, D. 1998. Robust anisotropic diffusion. *IEEE Trans. Image Processing 7*, 3 (Mar.), 421–432.

CHIU, K., HERF, M., SHIRLEY, P., SWAMY, S., WANG, C., AND ZIMMERMAN, K. 1993. Spatially nonuniform scaling functions for high contrast images. In *Proc. Graphics Interface*, 245–253.

COHEN, J., TCHOU, C., HAWKINS, T., AND DEBEVEC, P. 2001. Real-time high-dynamic range texture mapping. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, Eurographics, 313–320.

DEBEVEC, P. E., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH 97*, ACM SIGGRAPH / Addison Wesley, Los Angeles, California, Computer Graphics Proceedings, Annual Conference Series, 369–378.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 2000. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Graphics Interface*, 145–152.

DICARLO, J., AND WANDELL, B. 2000. Rendering high dynamic range images. *Proceedings of the SPIE: Image Sensors 3965*, 392–401.

DURAND, F. 2002. An invitation to discuss computer depiction. In *Proc. NPAR'02*.

ELAD, M. to appear. On the bilateral filter and ways to improve it. *IEEE Trans. on Image Processing*.

FERWERDA, J. 1998. Fundamentals of spatial vision. In *Applications of visual perception in computer graphics*. Siggraph '98 Course Notes.

HAMPEL, F. R., RONCHETTI, E. M., ROUSSEEUW, P. J., AND STAHEL, W. A. 1986. *Robust Statistics: The Approach Based on Influence Functions*. Wiley, New York.

HUBER, P. J. 1981. *Robust Statistics*. John Wiley and Sons, New York.

JOBSON, RAHMAN, AND WOODELL. 1997. A multi-scale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Trans. on Image Processing: Special Issue on Color Processing 6* (July), 965–976.

LARSON, G. W., RUSHMEIER, H., AND PIATKO, C. 1997. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics 3*, 4 (October - December), 291–306.

MADDEN, B. 1993. Extended intensity range imaging. Tech. rep., U. of Pennsylvania, GRASP Laboratory.

MCCOOL, M. 1999. Anisotropic diffusion for monte carlo noise reduction. *ACM Trans. on Graphics 18*, 2, 171–194.

MITSUNAGA, T., AND NAYAR, S. K. 2000. High dynamic range imaging: Spatially varying pixel exposures. In *IEEE CVPR*, 472–479.

OH, B. M., CHEN, M., DORSEY, J., AND DURAND, F. 2001. Image-based modeling and photo editing. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 433–442.

PATTANAIK, S. N., FERWERDA, J. A., FAIRCHILD, M. D., AND GREENBERG, D. P. 1998. A multiscale model of adaptation and spatial vision for realistic image display. In *Proceedings of SIGGRAPH 98*, ACM SIGGRAPH / Addison Wesley, Orlando, Florida, Computer Graphics Proceedings, Annual Conference Series, 287–298.

PERONA, P., AND MALIK, J. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE PAMI 12*, 7, 629–639.

RUDMAN, T. 2001. *The Photographer's Master Printing Course*. Focal Press.

RUSHMEIER, H. E., AND WARD, G. J. 1994. Energy preserving non-linear filters. In *Proceedings of SIGGRAPH 94*, ACM SIGGRAPH / ACM Press, Orlando, Florida, Computer Graphics Proceedings, Annual Conference Series, 131–138.

SAINT-MARC, P., CHEN, J., AND MEDIONI, G., 1991. Adaptive smoothing: a general tool for early vision.

SCHECHNER, Y. Y., AND NAYAR, S. K. 2001. Generalized mosaicing. In *Proc. IEEE CVPR*, 17–24.

SCHLICK, C. 1994. Quantization techniques for visualization of high dynamic range pictures. *5th Eurographics Workshop on Rendering*, 7–20.

SOCOLINSKY, D. 2000. Dynamic range constraints in image fusion and visualization. In *Proc. Signal and Image Processing*.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proc. IEEE Int. Conf. on Computer Vision*, 836–846.

TUMBLIN, J., AND RUSHMEIER, H. 1993. Tone reproduction for realistic images. *IEEE Comp. Graphics & Applications 13*, 6, 42–48.

TUMBLIN, J., AND TURK, G. 1999. Lcis: A boundary hierarchy for detail-preserving contrast reduction. In *Proceedings of SIGGRAPH 99*, ACM SIGGRAPH / Addison Wesley Longman, Los Angeles, California, Computer Graphics Proceedings, Annual Conference Series, 83–90.

TUMBLIN, J., HODGINS, J., AND GUENTER, B. 1999. Two methods for display of high contrast images. *ACM Trans. on Graphics 18*, 1, 56–94.

TUMBLIN, J. 1999. *Three methods of detail-preserving contrast reduction for displayed images*. PhD thesis, College of Computing Georgia Inst. of Technology.

WARD, G. J. 1994. The radiance lighting simulation and rendering system. In *Proceedings of SIGGRAPH 94*, ACM SIGGRAPH / ACM Press, Orlando, Florida, Computer Graphics Proceedings, Annual Conference Series, 459–472.

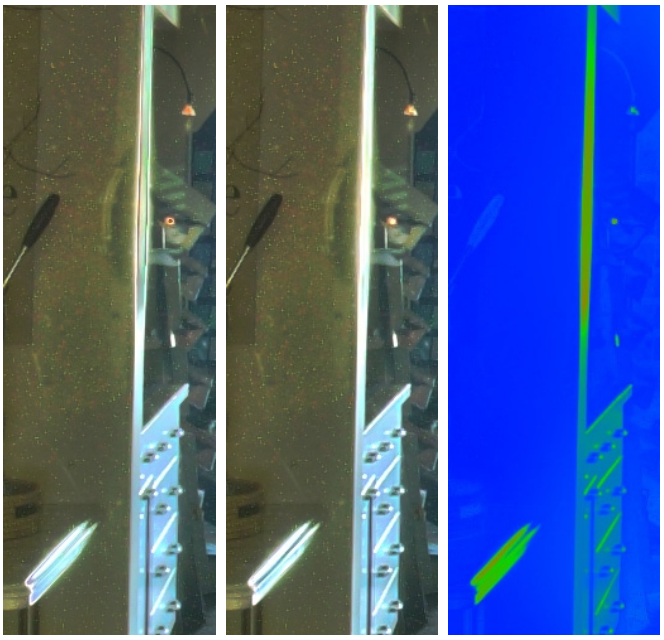
YANG, D., GAMAL, A. E., FOWLER, B., AND TIAN, H. 1999. A 640x512 cmos image sensor with ultrawide dynamic range floating-point pixel-level adc. *IEEE Journal of Solid State Circuits 34*, 12 (Dec.), 1821–1834.

Figure 15: Foggy scene. Radiance map courtesy of Jack Tumblin, Northwestern University [Tumblin and Turk 1999].

Figure 16: Grove scene. Radiance map courtesy of Paul Debevec, USC [Debevec and Malik 1997].

Figure 17: Interior scene.

Figure 18: Hotel room. The rightmost image shows the uncertainty. Designed and rendered by Simon Crone using RADIANCE [Ward 1994]. Source image: Proposed Burswood Hotel Suite Refurbishment (1995). Interior Design - The Marsh Partnership, Perth, Australia. Computer simulation - Lighting Images, Pert, Australia. Copyright (c) 1995 Simon Crone.



without with uncertainty fix uncertainty

Figure 19: Zoom of Fig. 17. The haloing artifacts in the vertical highlight and in the lamp are dramatically reduced. The noise is due to the sensor.

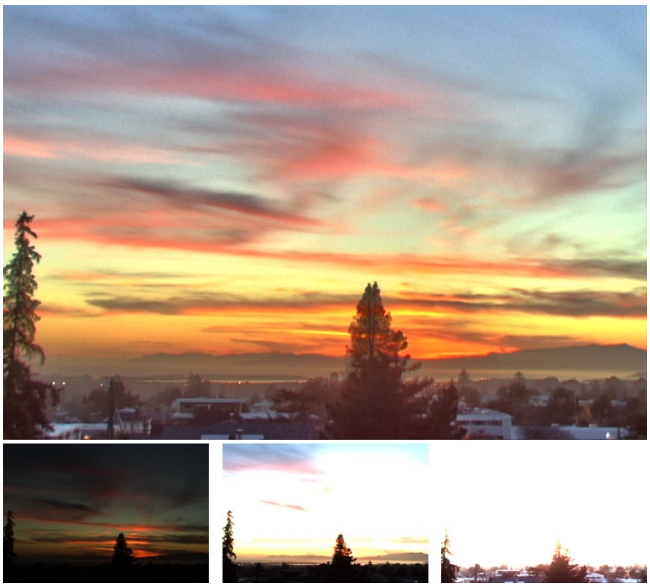
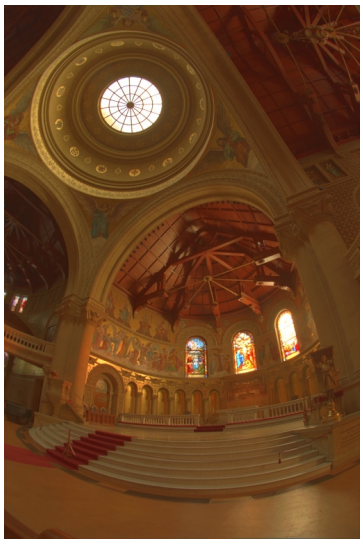
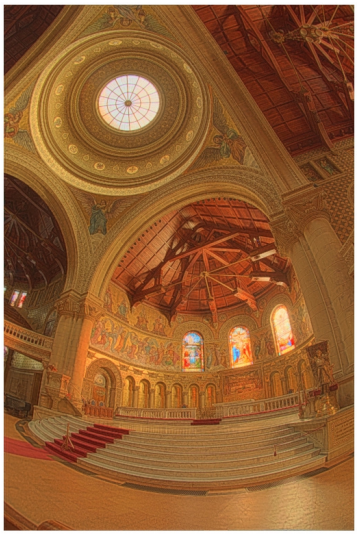


Figure 20: Vine scene. Radiance map courtesy of Paul Debevec, USC [Debevec and Malik 1997].



User-optimized gamma correction only on the intensity

Histogram adjustment [Larson et al. 1997]



LCIS. Image reprinted by permission, copyright ©1999 Jack Tumblin [Tumblin and Turk 1999]

Figure 21: Stanford Memorial Church, displayed with different methods.

Figure 22: Stanford Memorial Church displayed using bilateral filtering. The rightmost frame is the color-coded base layer. Radiance map courtesy of Paul Debevec, USC [Debevec and Malik 1997].

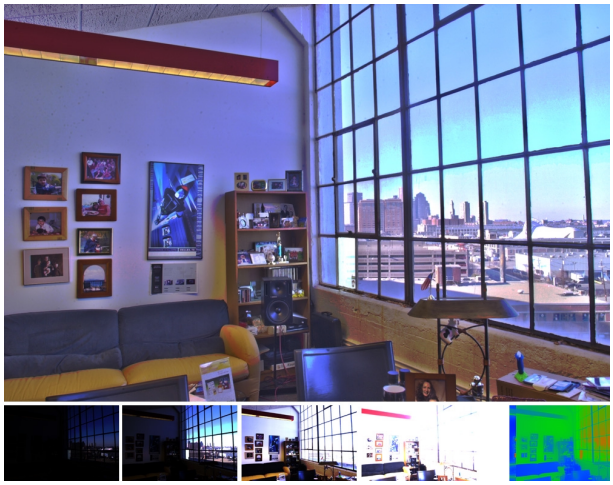


Figure 23: Window scene. The rightmost image shows the color-coded base layer.

Bilateral Mesh Denoising

Shachar Fleishman Iddo Drori Daniel Cohen-Or
School of Computer Science, Tel Aviv University*

Abstract

We present an anisotropic mesh denoising algorithm that is effective, simple and fast. This is accomplished by filtering vertices of the mesh in the normal direction using local neighborhoods. Motivated by the impressive results of bilateral filtering for image denoising, we adopt it to denoise 3D meshes; addressing the specific issues required in the transition from two-dimensions to manifolds in three dimensions. We show that the proposed method successfully removes noise from meshes while preserving features. Furthermore, the presented algorithm excels in its simplicity both in concept and implementation.

CR Categories: I.3.0 [COMPUTER GRAPHICS]: General
Keywords: Mesh Denoising, Bilateral Filtering

1 Introduction

With the proliferation of 3D scanning tools, interest in removing noise from meshes has increased. The acquired raw data of the sampled model contains additive noise from various sources. It remains a challenge to remove the noise while preserving the underlying sampled surface, in particular its fine features. Related techniques like mesh smoothing or mesh fairing alter the given surface to increase its degree of smoothness. The goal of these techniques is to create semi-uniform triangulation, often with subdivision connectivity. This paper focuses on mesh denoising, which is an important preprocess for many digital geometry applications that rely on local differential properties of the surface.

Denoising the sampled data can be applied either before or after generating the mesh. The advantage of denoising a mesh rather than a point-cloud, is that the connectivity information implicitly defines the surface topology and serves as a means for fast access to neighboring samples. The information in a mesh can be separated into two orthogonal components: a *tangential* and a *normal* component. The normal component encodes the geometric information of the shape, and the tangential component holds parametric information [Guskov et al. 1999]. In this formulation, moving vertices along their normal directions, modifies only the geometry of the mesh. Related to this notion are evolution curves [Osher and Sethian 1988], where points are shifted in the direction of the normal at a distance that is proportional to their curvature, to get smoother curves over time. Our denoising method is based on this idea, shifting mesh vertices along their normal direction.

The extensive research on image denoising serves as a foundation for surface denoising and smoothing algorithms. However, adapting these algorithms from the two dimensional plane to a surface in three dimensions is not straightforward for three main reasons: (i) **Irregularity**; unlike images, meshes are irregular both in connectivity and sampling, (ii) **Shrinkage**; image denoising algorithms are typically not energy preserving. While this is less noticeable in images, in meshes, this is manifested as shrinkage of the mesh, (iii) **Drifting**; naive adaptation of an image denoising technique may cause artifacts known as *vertex drifts*, in which the regularity of the mesh decreases.

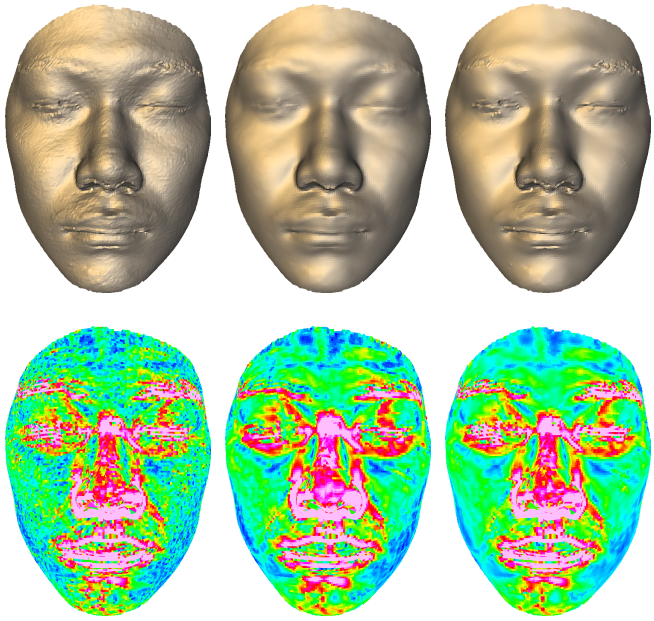


Figure 1: Denoising a scanned model: On the left is the input model, in the middle is the result of implicit fairing [Desbrun et al. 1999], and on the right is the result of our algorithm. The top row visualizes the details of the models, and on the bottom row is a mean curvature visualization. Data courtesy of Alexander Belyaev.

The bilateral filter, introduced by Tomasi and Manduchi [1998], is a nonlinear filter derived from Gaussian blur, with a feature preservation term that decreases the weight of pixels as a function of intensity difference. It was shown that bilateral filtering is linked to anisotropic diffusion [Barash 2002], robust statistics [Durand and Dorsey 2002], and Bayesian approaches [Elad 2001]. Despite its simplicity, it successfully competes with image denoising algorithms in the above categories. The bilateral filtering of images and its adaptation to meshes has an intuitive formulation, which leads to a simple method for selecting the parameters of the algorithm.

The contribution of this paper is a mesh denoising algorithm that operates on the geometric component of the mesh. The origin of the denoising algorithm is the bilateral filter that has a simple and intuitive formulation, is fast and easy to implement, and adapting it to meshes, yields results that are as successful as its predecessor.

1.1 Previous work

Image denoising is part of on-going research in image processing and computer vision. The state-of-the-art approaches to image de-

*e-mail: {shacharf | idrori | dcor}@tau.ac.il

noising include: wavelet denoising [Donoho 1995], nonlinear PDE based methods including total-variation [Rudin et al. 1992], and bilateral filtering [Tomasi and Manduchi 1998]. These approaches can be viewed in the framework of basis pursuit [Chen et al. 1999].

Typically, mesh denoising methods are based on image denoising approaches. Taubin [1995] introduced signal processing on surfaces that is based on the definition of the Laplacian operator on meshes. Peng et al. [2001] apply locally adaptive Wiener filtering to meshes. Geometric diffusion algorithms for meshes was introduced by Desbrun et al. [1999], they observed that fairing surfaces can be performed in the normal direction. Anisotropic diffusion for height fields was introduced by Desbrun et al. [2000], and Clarenz et al. [2000] formulated and discretized anisotropic diffusion for meshes. Recently, Bajaj and Xu [2003] achieved impressive results by combining the limit function of Loop subdivision scheme with anisotropic diffusion. Tasdizen et al. [2002] apply anisotropic diffusion to normals of the level-set representation of the surface, and in the final step, the level-set is converted to a mesh representation. Guskov et al. [1999] introduced a general signal processing framework that is based on subdivision, for which denoising is one application.

2 Bilateral mesh denoising

We open with a description of our method for filtering a mesh using local neighborhoods. The main idea is to define a local parameter space for every vertex using the tangent plane to the mesh at a vertex. The heights of vertices over the tangent plane are synonymous with the gray-level values of an image, and the closeness components of the filter are the tangential components. The term *offset* is used for the heights. Let S denote the noise-free surface, and let M be the input mesh with vertices that sample S with some additive noise. Let $\mathbf{v} \in M$ be a vertex of the input mesh, d_0 its signed-distance to S, and $\mathbf{n_0}$ the normal to S at the closest point to $\mathbf{v}$. The noise-free surface S is unknown and so is d_0, therefore we estimate the normal to the surface as the normal $\mathbf{n}$ to the mesh, and d estimates d_0 as the application of the filter, updating $\mathbf{v}$ as follows:

$$\hat{\mathbf{v}} = \mathbf{v} + d \cdot \mathbf{n}. \tag{1}$$

Note that we do not have to define a coordinate system for the tangential component; as explained below, we apply a one-dimensional filter with a spatial distance as a parameter. The filter is applied to one vertex at a time, computing a displacement for the vertex and updating its position.

Bilateral filtering of images. Following the formulation of Tomasi and Manduchi [1998], the bilateral filtering for image $I(\mathbf{u})$, at coordinate $\mathbf{u} = (x, y)$, is defined as:

$$\hat{I}(\mathbf{u}) = \frac{\sum_{\mathbf{p} \in N(\mathbf{u})} W_c(\|\mathbf{p}-\mathbf{u}\|) W_s(|I(\mathbf{u})-I(\mathbf{p})|) I(\mathbf{p})}{\sum_{\mathbf{p} \in N(\mathbf{u})} W_c(\|\mathbf{p}-\mathbf{u}\|) W_s(|I(\mathbf{u})-I(\mathbf{p})|)}, \tag{2}$$

where $N(\mathbf{u})$ is the neighborhood of $\mathbf{u}$. The closeness smoothing filter is the standard Gaussian filter with parameter σ_c: $W_c(x) = e^{-x^2/2\sigma_c^2}$, and a feature-preserving weight function, which we refer to as a *similarity weight function*, with parameter σ_s that penalizes large variation in intensity, is: $W_s(x) = e^{-x^2/2\sigma_s^2}$. In practice, $N(\mathbf{u})$ is defined by the set of points $\{\mathbf{q_i}\}$, where $\|\mathbf{u}-\mathbf{q_i}\| < \rho = \lceil 2\sigma_c \rceil$.

Algorithm. We begin introducing the algorithm by describing how to compute the normal and tangential components that are assigned to Eq. 2, yielding a new offset d. Since S is unknown, and we wish to use the edge preservation property of the bilateral filter, we define $S_v \subset S$ as the smooth connected component of S that is closest to $\mathbf{v}$. For the normal component, we would like to compute the offsets of the vertices in the neighborhood of $\mathbf{v}$, denoted by $\{\mathbf{q_i}\}$, over the noise-free smooth component S_v. We use the tangent plane to $\mathbf{v}$ defined by the pair $(\mathbf{v}, \mathbf{n})$ as a first-order approximation to S_v. The offset of a neighbor q_i is the distance between q_i and the plane. The following is the pseudo-code for applying a bilateral filter to a single vertex:

```
DenoisePoint( Vertex v, Normal n)
  {q_i} = neighborhood(v)
  K = |{q_i}|
  sum = 0
  normalizer = 0
  for i := 1 to K
    t = ||v - q_i||
    h = <n, v - q_i>
    w_c = exp(-t^2/(2σ_c^2))
    w_s = exp(-h^2/(2σ_s^2))
    sum += (w_c · w_s) · h
    normalizer += w_c · w_s
  end
  return Vertex v̂ = v + n · (sum/normalizer)
```

The plane that approximates the noise-free surface should on one hand, be a good approximation of the local surface, and on the other hand, preserve sharp features. The first requirement leads to smoothing the surface, while the latter maintains the noisy surface. Therefore, we compute the normal at a vertex as the weighted average (by the area of the triangles) of the normals to the triangles in the 1-ring neighborhood of the vertex. The limited neighborhood average smoothes the local surface without over-smoothing. In some cases, for example, of a synthetic surface, the normal of an edge vertex will erroneously point to the average direction and lead to a rounded edge.

For the tangential component, the correct measure of distance between vertices on the smooth surface is the geodesic distance between points. Since we use local neighborhoods, we approximate the geodesic distance using the Euclidean distance. This approximation seems to introduce artifacts in the neighborhood of sharp features, since vertices that happen to be geodesically far from the smoothed vertex may be geometrically close. Furthermore, the assumption from differential geometry that a neighborhood of a point on a surface can be evaluated by a function over the tangent plane to that point may not be satisfied. Both apparent problems do not hinder our algorithm because any of the above offending vertices is penalized by the similarity weight function.

Mesh shrinkage and vertex-drift. Image denoising and smoothing algorithms that are based on (possibly weighted) averaging of neighborhood, result is shrinkage of the object. Taubin [1995] solves this problem for the Laplacian operator by alternating shrink and expand operations. Another common approach is to preserve the volume of the object as suggested by Desbrun et al. [1999].

Our algorithm, also shrinks the object. This can be observed when smoothing a vertex that is a part of a curved patch; the offset of the vertex approaches the average of the offsets in its neighborhood. Therefore, we follow the volume preservation technique.

Vertex-drift is caused by algorithms that change the position of the vertices along the tangent plane as well as the normal direction. The result is an increase in the irregularity of the mesh. Our algorithm moves vertices along the normal direction, and so, no vertex-drift occurs.

Handling boundaries. Often meshes, in particular scanned data sets, are not closed. There are two aspects to note here: first, the shape of the boundary curve, which is the related problem of "mesh fairing". Second, is that a filter is defined for a neighborhood of a point. However for boundary points, part of the neighborhood is

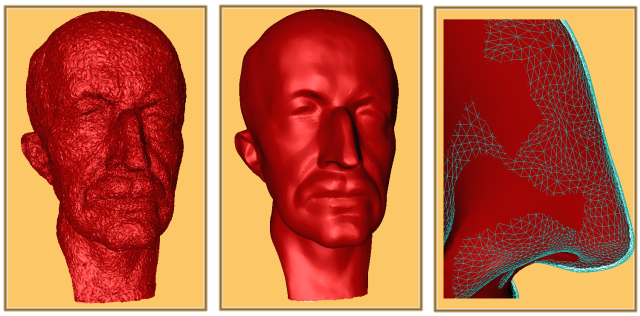


Figure 2: The shrinkage problem. On the left the model of Max Planck with heavily added random noise. In the middle the denoised model after four iterations of our algorithm without volume preservation. The Max-Planck model is courtesy of Christian Rössl from Max Planck Insitut für Informatik.

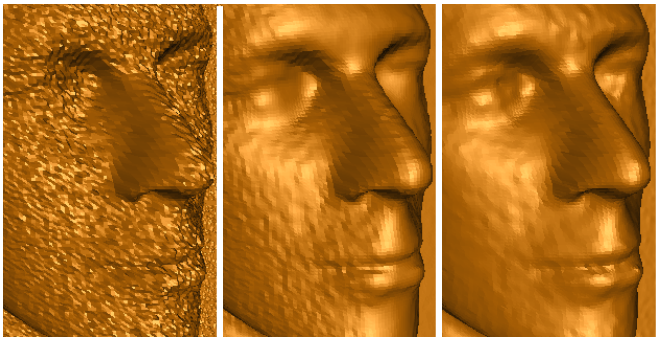


Figure 3: Comparison with AFP. On the left is the input model, in the middle is the result denoised by AFP, and on the right is the result of bilateral mesh denoising. Observe the difference in details in the area of the lips and eyes. The noisy model and the AFP denoised models are courtesy of Mathieu Desbrun.

not defined. One common solution to this problem is to define a virtual neighborhood by reflecting vertices over edges. Our filter inherently handles boundaries by treating them as sharp edges with virtual vertices at infinity. The similarity weight sets the weight of virtual vertices to zero, and thus, the normalization of the entire filter causes boundaries to be handled correctly.

Parameters. The parameters of the algorithm are: σ_c, σ_s, the kernel size ρ, and the number of iterations. We propose an intuitive user-assisted method for setting these parameters. Two parameters, σ_c and σ_s are interactively assigned: the user selects a point of the mesh where the surface is expected to be smooth, and then a radius of the neighborhood of the point is defined. The radius of the neighborhood is assigned to σ_c, and we set $\rho = 2\sigma_c$. Then σ_s is set to the standard deviation of the offsets in the selected neighborhood.

One may choose a large σ_c and perform a few iterations, or choose a narrow filter and increase the number of iterations. Multiple iterations with a narrow filter has the effect of applying a wider filter, and results in efficient computation. Using a small value for σ_c is sensible for two reasons: (i) large values may cross features as shown in, and (ii) smaller values result in a smaller neighborhood which leads to faster computation of every iteration.

In all the results shown in this paper, we used up to five iterations, we found a small number of iterations sufficient for our purposes, and advantageous both to the speed of computation and for the numerical stability of the filter.

Noisy data may lead to unstable computation of the normals if the 1-ring neighborhood of a vertex is used to compute the normals. For extremely noisy data, the normal to a vertex is computed using the k-ring neighborhood of the vertex, where k is defined by the user. For every scanned models that we denoised, the normals were

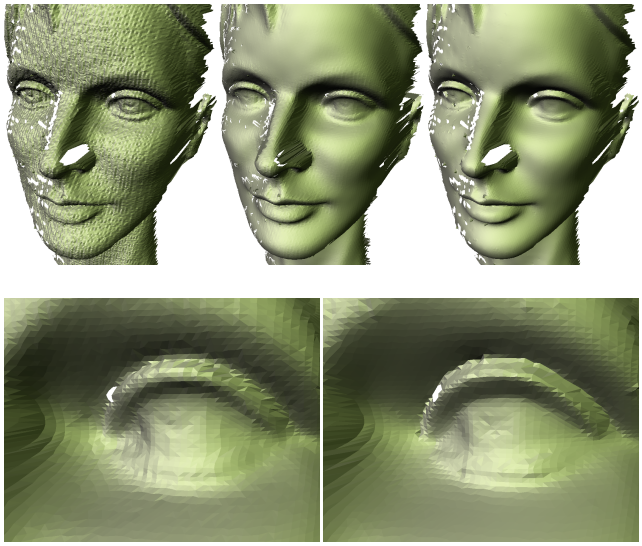


Figure 4: Results of denoising the face model. On the top row from left to right are the input noisy mode, the results of [Jones et al. 2003], and our result. On the bottom we zoom on the right eye of the model, where the bottom left image shows the results of Jones et al. , and on the bottom right is the result of our method. The face model is courtesy of Jean-Yves Bouguet.

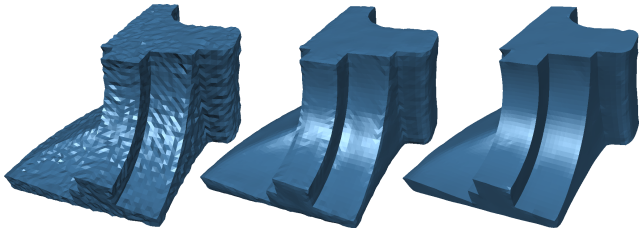


Figure 5: Results of denoising the Fandisk model. On the left is the input noisy model, in the middle is the results of [Jones et al. 2003], and on the right is our result.

computed using the 1-ring neighborhoods. Note that only for the Max Palanck (Figure 2) model, we were required to use the 2-ring neighborhood to compute normals.

3 Results

We have implemented the bilateral mesh denoising algorithm as described in the previous section and compared our results to the results of the anisotropic denoising of height fields algorithm (*AFP*) [Desbrun et al. 2000], Jones et al. [2003], and the *implicit fairing (IF)* algorithm [Desbrun et al. 1999] [1]. The times are reported on a 1GHz Pentium™ III. In Figure 3, we compare the AFP algorithm with our results. The mesh with 175K vertices is smoothed by three iterations in 10 seconds. Observe the preserved details near the mouth of the model denoised by our algorithm. Figure 1 shows a comparison with implicit fairing. The smoothness of the object can be appreciated from the visualization of the mean curvature in the second row. The model has 17K vertices, and it was denoised in three iterations, taking 1.8 seconds. In Figure 6 we show the denoising of a CAD object. Observe that the sharp features are preserved, but vertices with larger error are treated as outliers and thus are not smoothed out. For the Max Planck model (Figure2) (100k vertices), the timing for a single iteration is 5.75 seconds, and the total number of iterations was four.

[1] Implementation courtesy of Y. Ohtake

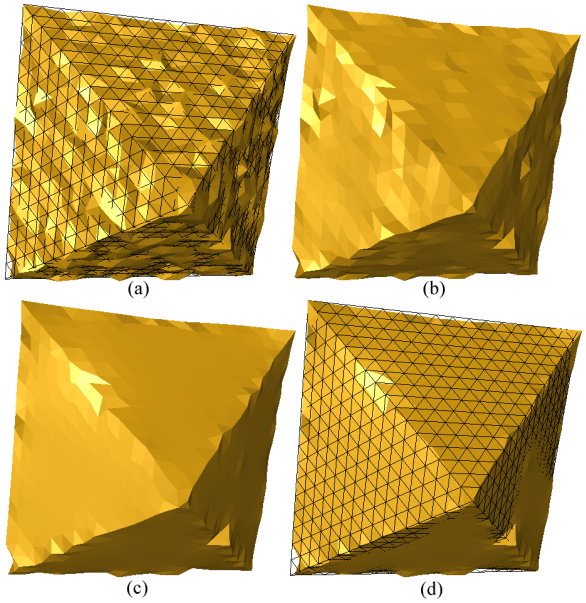


Figure 6: Denoising of CAD-like model. (a) is the input noisy model, (b) is the result of two iterations of our algorithm, (c) and (d) are the result of five iterations of our algorithm, where in (d) the clean model was superimposed on the denoised model.

Independently, Jones et al. [2003] present a similar algorithm that uses bilateral filtering for smoothing surfaces. The main difference between the two methods is in the surface predictor. More specifically, Jones et al. take the distance between the point and its projection onto the plane of a neighboring triangle, whereas our approach takes the distance between a neighboring point and the tangent plane. While we perform several filtering iterations, Jones et al. smooth a surface in a single pass. Figure 4 and 5 compare the results for two different types of models.

Volume preservation is a global operation, whereas denoising is a local operation. In Figure 2, we visualize the local change of volume by adding severe synthetic noise (of zero mean and variance of 0.05% of the diagonal of the bounding box of the model) to the clean model of Max Planck, and then denoised the model. On the right, we zoom on a feature of the model, superimposing the original mesh on top of the smoothed model, showing that the change in shape and volume is minimal.

4 Discussion and future work

Choosing the tangent plane as an approximation of S_v is an important component of our algorithm. Positioning the plane at the average of the offsets of neighboring points would improve our approximation of the smooth surface. However, this would nullify the effect of the similarity term of the bilateral filter. We expect that computing the offsets over a higher order function such as a polynomial of low degree will reduce the shrinkage problem. Finding a high-order, feature-preserving function for the noise-free surface is a difficult problem. In the future, we would like to investigate this possibility, by combining the bilateral filter with a rational function.

The key points of our algorithm are the choice of tangent plane and moving vertices along the normal direction. However this change in vertex position may lead to self-intersection of the denoised mesh. During the application of our algorithm to vertices on two sides of the edge, each vertex moves inwards, for sharp edges this will cause self-intersection after a number of iterations that is proportional to the angle of the edge.

The algorithm that we present assumes that the mesh is sampled regularly. This assumption is made when we fix the values of σ_c. Highly irregular meshes are uncommon in scanned data-sets. To handle irregular data-sets, the parameters must be adjusted locally.

We presented a mesh-denoising algorithm that modifies vertices in the normal direction. The bilateral filtering algorithm that we use is practical, clear and simple. The proposed method deals with irregular meshes and does not perform any reparameterization. In addition, the only property of the mesh that we use is the topological information, and therefore, the algorithm can be adapted to point-based representations.

Acknowledgements

We wish to thank Alexande Belyaev for his numerous remarks and suggestions. This work was supported in part by the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities, and by the Israeli Ministry of Science, and by a grant from the German Israel Foundation (GIF).

References

BAJAJ, C. L., AND XU, G. 2003. Anisotropic diffusion of subdivision surfaces and functions on surfaces. *ACM Transactions on Graphics (TOG) 22*, 1, 4–32.

BARASH, D. 2002. A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 6.

CHEN, S. S., DONOHO, D. L., AND SAUNDERS, M. A. 1999. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing 20*, 1, 33–61.

CLARENZ, U., DIEWALD, U., AND RUMPF, M. 2000. Anisotropic geometric diffusion in surface processing. In *IEEE Visualization 2000*.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of SIGGRAPH 99*, 317–324.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 2000. Anisotropic feature-preserving denoising of height fields and bivariate data. In *Graphics Interface*, 145–152.

DONOHO, D. L. 1995. De-noising by soft-thresholding. *IEEE Transactions on Information Theory 41*, 3, 613–627.

DURAND, F., AND DORSEY, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics (TOG) 21*, 3, 257–266.

ELAD, M. 2001. On the bilateral filter and ways to improve it. *IEEE Transactions On Image Processing*.

GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. 1999. Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH 99*, 325–334.

JONES, T., DURAND, F., AND DESBRUN, M. 2003. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*.

OSHER, S., AND SETHIAN, J. 1988. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics 79*, 12–49.

PENG, J., STRELA, V., AND ZORIN, D. 2001. A simple algorithm for surface denoising. In *IEEE Visualization 2001*, 107–112.

RUDIN, L., OSHER, S., AND FATEMI, E. 1992. Nonlinear total variation based noise removal algorithms. *Physica D 60*, (1–4), 259–268.

TASDIZEN, T., WHITAKER, R., BURCHARD, P., AND OSHER, S. 2002. Anisotropic geometric diffusion in surface processing. In *IEEE Visualization 2002*.

TAUBIN, G. 1995. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH 95*, 351–358.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *ICCV*, 839–846.

Non-Iterative, Feature-Preserving Mesh Smoothing

Thouis R. Jones
MIT

Frédo Durand
MIT

Mathieu Desbrun
USC

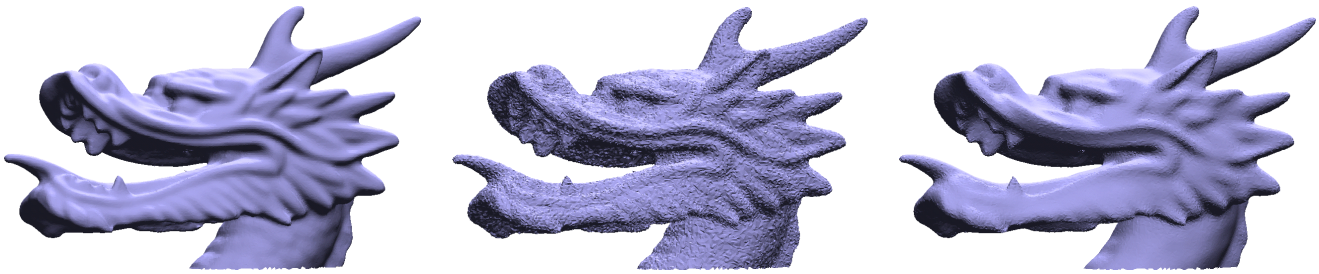


Figure 1: The dragon model (left) is artificially corrupted by Gaussian noise ($\sigma = 1/5$ of the mean edge length) (middle), then smoothed in a single pass by our method (right). Note that features such as sharp corners are preserved.

Abstract

With the increasing use of geometry scanners to create 3D models, there is a rising need for fast and robust mesh smoothing to remove inevitable noise in the measurements. While most previous work has favored diffusion-based iterative techniques for feature-preserving smoothing, we propose a radically different approach, based on robust statistics and local first-order predictors of the surface. The robustness of our local estimates allows us to derive a *non-iterative* feature-preserving filtering technique applicable to arbitrary "triangle soups". We demonstrate its simplicity of implementation and its efficiency, which make it an excellent solution for smoothing large, noisy, and non-manifold meshes.

Keywords: mesh processing, mesh fairing, robust estimation, mesh smoothing, anisotropic diffusion, bilateral filtering.

1 Introduction

With geometry scanners becoming more widespread and a corresponding growth in the number and complexity of scanned models, *robust* and *efficient* geometry processing becomes increasingly desirable. Even with high-fidelity scanners, the acquired 3D models are invariably noisy [Rusinkiewicz et al. 2002; Levoy et al. 2000], and therefore require smoothing. Similarly, shapes extracted from volume data (obtained by MRI or CT devices, for instance) often contain significant amounts of noise, be it topological [Guskov and Wood 2001; Wood et al. 2002] or geometric [Taubin 1995; Desbrun et al. 1999], that must be removed before further processing. Removing noise while preserving the shape is, however, no trivial matter. Sharp features are often blurred if no special care is taken. To make matters worse, scanned meshes often have cracks and non-manifold regions.

1.1 Previous Work

A wide variety of mesh smoothing algorithms have been proposed in recent years. Taubin [1995] pioneered fast mesh smoothing by proposing a simple, linear and isotropic technique to enhance the smoothness of triangulated surfaces without resorting to expensive functional minimizations. Desbrun et al. [1999] extended this approach to irregular meshes using a geometric flow analogy, and introduced the use of a conjugate gradient solver that safely removes the stability condition, allowing for significant smoothing in reasonable time even on large meshes. Other improvements followed, such as a method combining geometry smoothing and parameterization regularization [Ohtake et al. 2000]. However, these efficient techniques are all isotropic, and therefore indiscriminately smooth noise *and* salient features: a noisy cube as input will become extremely rounded before becoming smooth. This lack of selectivity is limiting in terms of applications.

Feature-preserving surface fairing has also been proposed more recently [Desbrun et al. 2000; Clarenz et al. 2000; Meyer et al. 2002; Zhang and Fiume 2002; Bajaj and Xu 2003], mostly inspired by image processing work on scale-space and anisotropic diffusion [Perona and Malik 1990]. The idea behind these approaches is to modify the diffusion equation to make it non-linear and/or anisotropic. The curvature tensor determines the local diffusion, thus preserving (or even enhancing) sharp features. Although the results are of much higher quality, these methods rely on shock formation to preserve details, which affects the numerical conditioning of the diffusion equations. This can cause significant computational times, even after mollification of the data.

Other researchers have proposed diffusion-type smoothing on the normal field itself [Taubin 2001; Belyaev and Ohtake 2001; Ohtake et al. 2002; Tasdizen et al. 2002]; fairing is achieved by first smoothing the normal field, and then evolving the surface to match the new normals. Here again, the results are superior to those from isotropic techniques, but with roughly similar computational cost as anisotropic diffusion on meshes.

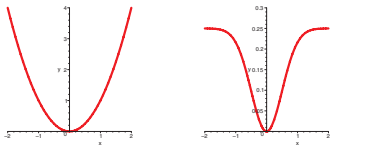
Locally adaptive Wiener filtering has also been used with success for 3D meshes by Peng et al. [2001], and for point-sampled surface by Pauly and Gross [2001]. However, these methods rely on semi-regular connectivity or local param-

eterization, respectively. A different approach is taken by Alexa [2002], similar to anisotropic diffusion, though with larger neighborhoods used for filtering. This also results in a fast method, and avoids some of the limitations discussed below, but still relies on a connected mesh and iterative application.

The diffusion-based feature-preserving techniques are, in essence, all *local and iterative*. From very local derivative approximations, geometry is iteratively updated until the noise has been diffused sufficiently. Numerical tools, such as preconditioned conjugate gradient solvers or algebraic multigrid solvers, can be used to improve efficiency by making the iterations more stable. Nevertheless, the diffusion-type setting that is the basis of these approaches requires manifoldness, not always present in raw scanned data. In order to address the need for robust and fast feature preserving smoothing, we propose to recast mesh filtering as a case of robust statistical estimation.

1.2 Robust Statistics

The field of *robust statistics* is concerned with the development of statistical estimators that are robust to the presence of outliers and to deviations from the theoretical distribution [Huber 1981; Hampel et al. 1986]. Naïve estimators such as least-squares give too much influence to outliers, because the error function or norm they minimize is large for data points far from the estimator (quadratic in the case of least squares). In contrast, robust estimators are based on minimizing an energy that gives low weight to outliers, as illustrated by the Gaussian robust norm in Fig. 2: after a certain distance from the estimator, controlled by a scale σ, an increasingly distant outlier has only limited effect.



x^2 $\quad$ $\rho_\sigma(x) = \sigma^2 - e^{-\frac{x^2}{2\sigma^2}}$

Figure 2: Least-square vs. Gaussian error norm (after [Black et al. 1998]).

Black et al. [1998] showed that anisotropic diffusion can be analyzed in the framework of robust statistics. The edge-stopping functions of anisotropic diffusion [Perona and Malik 1990] serve the same role as robust energy functions. Anisotropic diffusion minimizes such a function using an iterative method.

The bilateral filter is an alternative edge-preserving filter proposed by Smith and Brady [1997] (see also [Tomasi and Manduchi 1998]). The output $E(p)$ at a pixel p is a weighted average of the surrounding pixels in the input image I, where the weight of a pixel q depends not only on the spatial distance $||q-p||$, but also on the signal difference $||I(q) - I(p)||$:

$$E(p) = \frac{1}{k(p)} \sum_{q \in \Omega} I(q) \;\; f(q-p) \;\; g(I(q) - I(p)), \qquad (1)$$

where $k(p)$ is the normalization factor

$$k(p) = \sum_{q \in \Omega} f(q-p) \;\; g(I(q) - I(p)) \qquad (2)$$

In practice, a spatial Gaussian f and a Gaussian influence weight g are often used.

This dependence on the signal difference allows one to give less influence to outliers. Durand and Dorsey [2002] show that bilateral filtering is a robust estimator and that a Gaussian influence weight corresponds to minimizing a Gaussian error norm. They also show that bilateral filtering is essentially similar to anisotropic diffusion. However, the bilateral filter is a *non-iterative* robust estimator, or *w-estimator* [Huber 1981], which makes it more efficient than iterative schemes. In particular, this approach does not have to deal with shock formation at strong edges, and is therefore more stable than anisotropic diffusion. See also the work by Barash [2001] and Elad [2002].

1.3 Contributions

In this paper, we propose a novel feature-preserving fairing technique for arbitrary surface meshes based on non-iterative, robust statistical estimations[1]. Contrasting drastically with previous diffusion-based methods, our fast and stable approach relies on local robust estimations of shape. Moreover, our method does not require manifoldness of the input data, and can therefore be applied to "triangle soup".

One of our key insights is that feature preserving smoothing can be seen as estimating a surface in the presence of outliers. The extension from existing robust statistics techniques to surface filtering is, however, far from trivial because of the nature of the data: in a mesh, the *spatial location* and the *signal* are one and the same. This makes the definition of outliers and the control of their influence challenging. We propose to capture the smoothness of a surface by defining local first-order *predictors*. Using a robust estimator, we find the new position of each vertex as weighted sum of the predictions from the predictions in its spatial neighborhood. We will show that our method treats points on opposite sides of a sharp feature as *outliers* relative to one another. This limits smoothing across corners, which preserves features.

2 Non-Iterative, Feature-Preserving Mesh Smoothing

We cast feature-preserving mesh filtering as a robust estimation problem on vertex positions. The estimate for a vertex is computed using the prediction from nearby triangles. Moving each vertex to a robust estimate of its position removes noise and smoothes the mesh while preserving features.

2.1 Robust Estimation of Vertex Positions

To allow the proper definition of outliers, we must separate spatial location and signal. We capture surface smoothness using first-order predictors, i.e., tangent planes. In practice we use predictors based on triangles of the mesh as they represent natural tangent planes to the surface. The surface predictor Π_q defined by a triangle q is just the tangent plane of q (see Fig. 3a).

We use a method analogous to bilateral filtering for images [Smith and Brady 1997; Tomasi and Manduchi 1998], but we form the estimate for the new position of a vertex p based on the *predictions* $\Pi_q(p)$ from its spatially nearby triangles.

[1] In a contemporaneous work, Fleishman et al. [2003] present a similar technique (cf. Section 4).

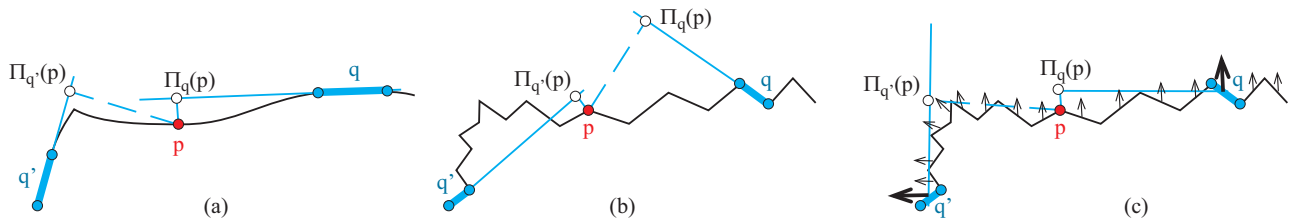


Figure 3: (a) The prediction $\Pi_q(p)$ for a point p based on the surface at q is the projection of p to the plane tangent to the surface at q. Points across a sharp feature result in predictions that are farther away, and therefore given less influence. (b) Noisy normals can lead to poor predictors. (c) Mollified normal alleviate this problem. Note that corners are preserved because points are not displaced by the mollification: only the normals are smoothed.

We employ a spatial weight f that depends on the distance $||p - c_q||$ between p and the centroid c_q of q. We also use an influence weight g that depends on the distance $||\Pi_q(p) - p||$ between the prediction and the original position of p. Finally we weight by the area a_q of the triangles to account for variations in the sampling rate of the surface. The estimate p' for a point on surface S is then:

$$p' = \frac{1}{k(p)} \sum_{q \in S} \Pi_q(p)\ a_q\ f(||c_q - p||)\ g(||\Pi_q(p) - p||), \quad (3)$$

where k is a normalizing factor (sum of the weights)

$$k(p) = \sum_{q \in S} a_q\ f(||c_q - p||)\ g(||\Pi_q(p) - p||), \quad (4)$$

Gaussians are used both for the spatial weight f and for the influence weight g in this paper. Other robust influence weights could also be used, but Gaussians have performed well in our experiments, as well as the work of others [Smith and Brady 1997; Tomasi and Manduchi 1998; Durand and Dorsey 2002]. The amount of smoothing is controlled by the widths σ_f of the spatial and σ_g of the influence weight Gaussians. As can be seen in Fig. 3(a), predictions from across a sharp feature are given less weight because the distance between the prediction $\Pi_q(p)$ and p is large, and is penalized by the influence weight g.

Filtering a mesh involves evaluating Equation (3) for every vertex and then moving them as a group to their estimated positions. Note that no connectivity is required beyond triangles: we simply use the Euclidean distance to the centroid of surrounding triangles to find the spatial neighborhood of a vertex. A wider spatial filter includes a larger number of neighbors in the estimate, and can therefore remove a greater amount of noise, or smooth larger features. The influence weight determines when the predictions of neighbors are considered outliers (by according them less weight), and thereby controls the size of features that are preserved in the filtering.

As shown by Black et al.[1998] and Durand and Dorsey [2002], the evaluation of Equation (3) corresponds to approximately minimizing

$$E(p) = \int_{q \in S} f(||c_q - p||)\ \rho(||\Pi_q(p) - p||)\ dq, \quad (5)$$

where $\rho(||\Pi_q(p) - p||)$ is the distance between a vertex and its predicted position under a robust error norm ρ [Hampel et al. 1986]. Robust error norms are bounded above by some maximum error, as discussed in Section 1.2. In our case, we seek to minimize a Gaussian error norm (see Fig. 2); The relation between ρ and g is $g(x) \equiv \rho'(x)/x$ [Black et al. 1998; Durand and Dorsey 2002; Hampel et al. 1986].

2.2 Mollification

Our predictors are based on the orientation of the tangent planes, as defined by the facet normals. Since the normals are first-order properties of the mesh, they are more sensitive to noise than vertex positions (Fig. 3(b)). Even so, the robust estimator performs well; we can however significantly improve the estimate with *mollification* [Huber 1981; Murio 1993]. We mollify our estimators by smoothing the normals.

We first perform a pass of non-robust smoothing using Equation (3) without the influence weight, and with the simplest predictor, $\Pi_q(p) = c_q$, corresponding to simple Gaussian smoothing. We use a different width for the spatial filter during mollification, and in practice have always set this to $\sigma_f/2$. The normals of the mollified mesh are then copied to the facets of the original mesh before the robust filtering is performed. Notice that we do *not* alter the positions of the vertices at all: we only need to mollify the first-order properties (the normals), not the 0-order location (see Fig. 3(c)). Some normals might be improperly smoothed by mollification near corners. This is why it is important not to move the vertices during mollification in order to preserve these features. Fig. 4 shows a comparison of filtering with and without mollification. Without mollification, the facet normals of the mesh are much noisier, resulting in less effective smoothing.

2.3 Feature Preservation

The filtering method we have proposed preserves features through two combined actions. First is the use of a robust influence weight function, as discussed, while the second is our use of a predictor for vertex positions based on the tangent planes of the mesh. This predictor does not move vertices located at sharp features separating smooth areas of the mesh, since feature vertices are "supported" by the prediction from both sides. Neither of these actions is sufficient alone (see the discussion of Fig. 7 below for examples of how the influence weight affects the filter), but together they provide excellent feature-preserving behavior. Note the connection to bilateral filtering for images, which uses a prior of piecewise constant images. This is a special case of our formulation, corresponding to the predictor $\Pi_q(p) = c_q$. As well, the use of the existing mesh facets helps to simplify our formulation and its implementation, as they provide direct estimates for surface tangents.

In essence, our technique also relates to ENO/WENO methods [Osher and Fedkiw 2002], a class of finite-difference-based, shock capturing numerical techniques for hyperbolic PDE integration. In a nutshell, they seek to avoid dissipation of shocks –the equivalent of sharp features in our geometric setting. They base their local evaluation of differential quantities only on the local neighbors of similar field

(a) Initial mesh (b) isotropic smoothing [Desbrun et al. 1999] (c) our approach without mollification (d) our approach with no influence weight g (e) our complete approach

Figure 4: Isotropic filtering vs. our method. Notice that details such as the upper and lower lids and the eye are better preserved, while flat regions are equivalently smoothed. (Original mesh courtesy of Jean-Yves Bouguet.)

value. For example, the evaluation of a second derivative at a shock is not centered as this would use information on both sides; in contrast the evaluation is one-sided to prevent numerical dissipation. Robust statistics offers a principled framework to extend similar key concepts to geometry.

3 Results

We demonstrate our results in Figs. 1 and 4-8. In each case, we use Gaussians for the spatial (f), influence weight (g), and mollification filters with standard deviations of $\sigma_f, \sigma_g, \frac{\sigma_f}{2}$, respectively. This choice for g corresponds to a Gaussian error norm. All meshes are rendered with flat shading to show faceting. Table 1 summarizes our results and the parameters used to generate them, given in terms of the mean edge length ($||e||$) of the particular mesh. The cost of each evaluation of Equation (3) depends on the number of facets that lie within the support of f, so the time to filter a mesh grows approximately as the size of the mesh times σ_f^2.

Fig. 4 shows a portion of a mesh from a 3D scan of a head. We show the original mesh, the result of isotropic smoothing by Desbrun et al. [1999], and our technique. We present this comparison to demonstrate the effectiveness of our approach for smoothing, even on noisy meshes with topological errors.

A comparison to the Wiener filtering approach of Peng et al [2001] is shown in Fig. 6. The parameters for our method were chosen to visually match the smoothness in flat areas. Our method preserves features better for larger amounts of smoothing (compare 6(d) and 6(e)). Also, as noted previously, Wiener filtering requires resampling the input to a semi-regular mesh, and only operates on surfaces with manifold topology, while our method can be applied more generally, to non-regular and disconnected meshes. We estimate that their implementation would take about 15 seconds to filter this mesh on our machine, in comparison to 60 (or more, depending on the smoothing parameters) for our technique.

In Fig. 10 we compare our method to anisotropic diffusion smoothing by Clarenz et al. [2000]. The original, noisy mesh is smoothed and smaller features removed by four iterations of diffusion. We have chosen the parameters of our method to match the result as closely as possible. One benefit of anisotropic diffusion is the ability to iteratively enhance edges and corners. Our method is not able to perform such enhancement in a single pass, resulting in a slightly different overall appearance, particularly in the hair, and slightly more noise around edges in the model.

We also show a false-color plot of the confidence measure k in Fig. 10. As can be seen, smoother areas (such as

Model	Fig.	Verts.	Time	$\sigma_f/\|\|e\|\|$/		$\sigma_g/\|\|e\|\|$	
Dragon head	1	100k	80 s	4	(14)	1	(4)
Face	4(d)	41k	16 s	1.5	(9.2)	0.4	(2.4)
	(c)		10 s	1.5	(0.9)	0.5	(0.3)
Dog	6(c)	195k	82 s	2.7	(6.6))	0.4	(0.9)
	(e)		132 s	4	(9.9)	1.3	(3.3)
Bunny	7(b)	35k	11 s	2	(12)	0.2	(1.2)
	(c)		12 s	2	(1.2)	4	(24)
	(d)		23 s	4	(24)	4	(24)
Venus	10	134k	54 s	2.5	(8.1)	1	(3.3)
Dragon	8	100k	79 s	4	(14)	2	(7)

Table 1: Results on a 1.4Ghz Athlon with 2GB of RAM. Times do not include the time to load meshes. The σs are expressed as ratios of the mean edges length $||e||$, and the numbers in parentheses are in thousandths of the bounding box diagonal for the particular meshes.

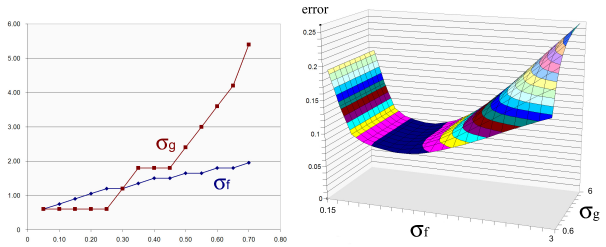
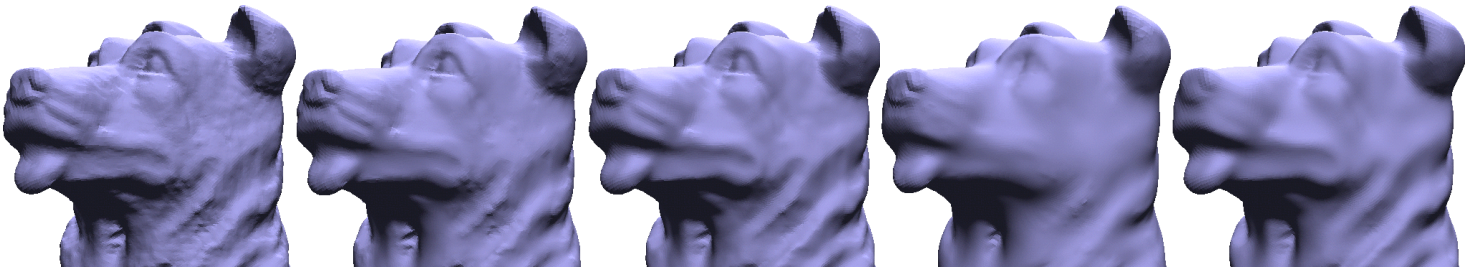


Figure 5: (a) Values of σ_f and σ_g that yield the most accurate denoising for a mesh corrupted with Gaussian noise, as a function of the variance of the noise. (b) Evolution of the error for a given noise level as a function of σ_f and σ_g. All values are in terms of the mean edge length.



Original | Wiener (low noise) | our method | Wiener (high noise) | our method

Figure 6: Comparison of our method and Wiener filtering. Parameters for Wiener filtering from [Peng et al. 2001]. Parameters for our method chosen to approximately match surface smoothness in flat areas. (Original and Wiener filtered meshes courtesy of Jianbo Peng.)

original | noise removal narrow spatial, narrow influence ($\sigma_f = 2$, $\sigma_g = 0.2$) | smooth small features narrow spatial, wide influence ($\sigma_f = 2$, $\sigma_g = 4$) | smooth large features wide spatial, wide influence ($\sigma_f = 4$, $\sigma_g = 4$)

Figure 7: The effect of varying spatial and influence weight functions. Filter widths σ_f, σ_g given in terms of mean edge length in the mesh. (Mesh from the Stanford University Computer Graphics Laboratory 3D scanning repository.)

the cheek) and features bordered by smooth areas (such as the edges of the nose) have higher confidence, while curved areas or those near more complicated features have lower confidence. See also Fig. 8.

We show the effects of varying spatial and influence weight function widths in Fig. 7. For a wide spatial filter but narrow influence weight, the mesh is smoothed only in mostly flat areas. In the converse, a narrow spatial filter and wide influence weight, small features are smoothed away but larger variations kept. Finally, for a wide spatial filter and wide influence weight, only the largest and strongest features are preserved. See Fig. 8 for a similar example of our method used to remove all but the most salient features of a mesh.

In order to facilitate denoising with our approach, we have performed experiments to find good values for σ_f and σ_g to smooth a model corrupted with a given amount of noise, such as might be produced by a scanner. If the amount of noise can be quantified, by examining an area on the model known to be flat, for example, then the plot in Fig. 5(a) shows the optimal values for σ_f, σ_g from our experiments. These values have been found effective on several models. The surface plot in Fig. 5(b) shows how, for a particular (representative) noise level, the post-filtering error changes. As can be seen, the error is most sensitive to σ_f. We compute the error as the L^2 distance between the original mesh before corruption and the filtered mesh [Khodakovsky et al. 2000].

In other applications, our general approach has been to increase σ_f and σ_g together until the filtered mesh is sufficiently smooth for our goals. We then decrease σ_g until features or noise that we are trying to remove begin to reappear.

All of our results demonstrate the effectiveness of our technique at feature preservation, due to a combination of a robust influence weight function and a first-order predictor, as discussed in Section 2.3. In particular, the tips of the ears of the bunny are preserved, as are the head and extremities of the dragon. See also Fig. 9, part of a scan of an origami piece.

We apply our filtering method to a mesh corrupted with synthetic noise in Fig. 1. In the noisy mesh, each vertex is displaced by zero-mean Gaussian noise with $\sigma_{\text{noise}} = \frac{1}{5}$ of the mean edge length, along the normal. We filter the dragon mesh to recover an estimate of the original shape. For comparison, in the scanned mesh of Fig. 4 we estimate $\sigma_{\text{noise}} \approx \frac{1}{7}$. These results shows the ability of our method to smooth even in the presence of extreme amounts of noise. Fig. 4 also indicates an area where our algorithm could be improved. Where a feature and noise coincide (e.g. in the nose), it is difficult to correctly separate the two. In Fig. 1, we have aimed for a smoother reconstruction, but lose some details in the process.

We have applied two basic optimizations to our implementation. We truncate the spatial filter at $2\sigma_f$ to limit the number of estimates that must be considered per vertex. This does not noticeably affect the results. We also group vertices and facets spatially for processing, to improve local-

ity of reference.

As presented, our method is not necessarily volume preserving. We have not encountered a mesh where this is an issue. Adjusting the mesh after filtering to preserve its volume is a straightforward extension [Desbrun et al. 1999].

4 Conclusion and Future Work

We have developed a novel, fast feature-preserving surface smoothing and denoising technique, applicable to triangle soups. We use a first-order predictor to capture the local shape of smooth objects. This decouples the signal and the spatial location in a surface, and allows us to use robust statistics as a solid base for feature preservation. A robust error norm is applied to the predictors and the vertex positions, from which we derive an efficient and stable one-step smoothing operator. Mollification is used to better capture shape and to obtain more reliable detection of outliers and features. We have demonstrated our algorithm on several models, for both noise removal and mesh smoothing.

Contemporaneous with this work, Fleishman et al. [2003] have proposed an extension of bilateral filtering to meshes with similar goals as this work, but with a different approach. The main contrasts are that vertex normals are computed from the mesh to perform local projections, after which a vertex's updated position is computed as the bilateral filter of its neighborhood treated as a height field. In rough terms, their method is faster but requires a connected mesh. The speed increase is due to two factors: they do not mollify normals, and the density of triangles is roughly half that of vertices in a mesh. They require connectivity to estimate normals. They also apply their filter iteratively, while we have concentrated on a single-pass technique. There is also a fundamental difference in how the two methods form predictions for a vertex's filtered position. Our method projects the central vertex to the planes of nearby triangles, while that of Fleishman et al. projects nearby vertices to the plane of the central vertex. The relative costs and benefits of these two approaches merits further study.

There are several avenues for improvement of our method. The normalization factor k in Equation (3) is the sum of weights applied to the individual estimates from a point's neighborhood (see Fig. 10). It therefore provides a measure of the confidence that should be attached to the estimate of the point's new position, as noted by Durand and Dorsey [2002]. We have not made use of the confidence measure k in this work, but feel that it could be a valuable tool in future approaches. In particular, we believe that it could be used to detect areas where a good estimate could not be formed, as on the sharp features in Fig. 1. Such areas could be processed further, perhaps by iterative filtering.

In our experience, the $O(\sigma_f^2)$ growth rate of our algorithm has not been a limiting factor. If it were to become so, a promising approach is to subsample the mesh by simplifying it with some fast method, and then filter the original mesh vertices based on the simplified version. Our method should also extend easily to out-of-core evaluation, since it does not require connectivity information and since the computations are spatially local. This would allow our method to be applied to extremely large models.

Finally, the extension of robust statistics to meshes suggests other possibilities for their application. The influence weight could include other data on the mesh, such as color. It should also be straightforward to extend our filter to other shape representations, such as volume data or point-sample models [Zwicker et al. 2002]. In the latter case, where each sample includes a normal, the methods transfer directly, as should the results. We also plan to explore how robust statistics could be added to existing techniques for surface approximation, such as Moving Least Squares [Levin 2001], to improve their robustness and sensitivity to noise.

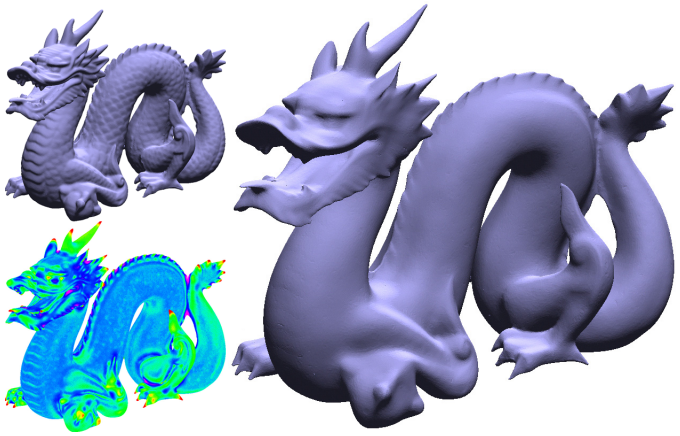


Figure 8: Original and smoothed dragon, and the confidence k for the smoothed dragon. Note that sharp features are preserved while other details are removed. (Mesh from the Stanford University Computer Graphics Laboratory 3D scanning repository.)

Acknowledgments Special thanks to Udo Diewald, Martin Rumpf, Jianbo Peng, Denis Zorin, and Jean-Yves Bouguet for providing us with their meshes and results, to Erik Demaine for the origami, to the Stanford 3D Scanning Repository, and to the SigDraft and MIT pre-reviewers for their feedback. This work was funded in part by the NSF (CCR-0133983, DMS-0221666, DMS-0221669, EEC-9529152).

References

Alexa, M. 2002. Wiener Filtering of Meshes. In *Proceedings of Shape Modeling International*, 51–57.

Bajaj, C., and Xu, G. 2003. Anisotropic Diffusion on Surfaces and Functions on Surfaces. *ACM Trans. Gr. 22*, 1, 4–32.

Barash, D. 2001. A Fundamental Relationship between Bilateral Filtering, Adaptive Smoothing and the Nonlinear Diffusion Equation. *IEEE PAMI 24*, 6, 844.

Belyaev, A., and Ohtake, Y. 2001. Nonlinear Diffusion of Normals for Crease Enhancement. In *Vision Geometry X, SPIE Annual Meeting*, 42–47.

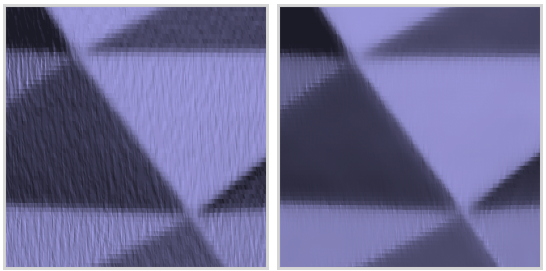
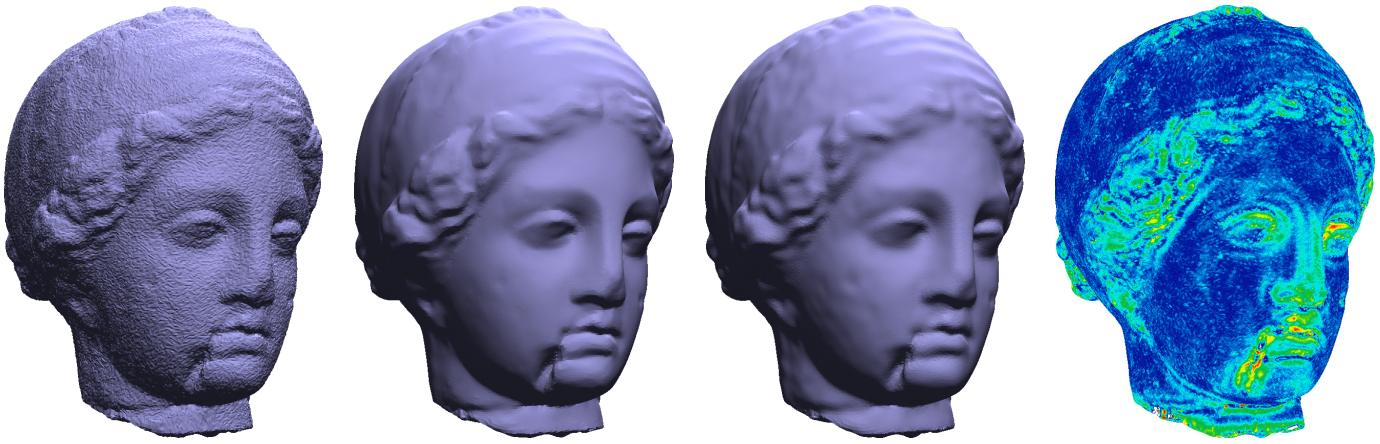


Figure 9: Corners and straight edges are preserved by our method, as shown in a portion of a 3D scan of an origami sculpture.



input mesh | anisotropic diffusion | our method | confidence k

Figure 10: Comparison of our method with anisotropic diffusion (four iterations) [Clarenz et al. 2000]. Parameters for our method chosen to match as best as possible. In the confidence plot, dark blue is high confidence, green and yellow lower confidence, and red least confidence. (Original and anisotropically smoothed mesh courtesy of Martin Rumpf.)

Black, M., Sapiro, G., Marimont, D., and Heeger, D. 1998. Robust anisotropic diffusion. *IEEE Trans. Image Processing 7*, 3, 421–432.

Clarenz, U., Diewald, U., and Rumpf, M. 2000. Anisotropic geometric diffusion in surface processing. In *IEEE Visualization 2000*, 397–405.

Desbrun, M., Meyer, M., Schröder, P., and Barr, A. H. 1999. Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. In *Proceedings of SIGGRAPH 99*, 317–324.

Desbrun, M., Meyer, M., Schröder, P., and Barr, A. H. 2000. Anisotropic Feature-Preserving Denoising of Height Fields and Bivariate Data. In *Graphics Interface*, 145–152.

Durand, F., and Dorsey, J. 2002. Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. *ACM Trans. Gr. 21*, 3, 257–266.

Elad, M. 2002. On the Bilateral Filter and Ways to Improve It. *IEEE Trans. on Image Processing 11*, 10, 1141–1151.

Fleishman, S., Drori, I., and Cohen-Or, D. 2003. Bilateral Mesh Denoising. *ACM Trans. Gr. (Proceedings of ACM SIGGRAPH)*.

Guskov, I., and Wood, Z. 2001. Topological Noise Removal. In *Graphics Interface 2001*, 19–26.

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. 1986. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley and Sons. ISBN 0471-63238-4.

Huber, P. J. 1981. *Robust Statistics*. John Wiley and Sons.

Khodakovsky, A., Schröder, P., and Sweldens, W. 2000. Progressive Geometry Compression. In *Proceedings of ACM SIGGRAPH 2000*, 271–278.

Levin, D. 2001. Mesh-independent surface interpolation. In *Advances in Computational Mathematics*, in press.

Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., and Fulk, D. 2000. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *Proceedings of SIGGRAPH 2000*, 131–144.

Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H. 2002. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Proceedings of Visualization and Mathematics*.

Murio, D. A. 1993. *The mollification method and the numerical solution of ill-posed problems*. Wiley.

Ohtake, Y., Belyaev, A., and Bogaeski, I. 2000. Polyhedral Surface Smoothing with Simultaneous Mesh Regularization. In *Geometric Modeling and Processing*, 229–237.

Ohtake, Y., Belyaev, A., and Seidel, H.-P. 2002. Mesh Smoothing by Adaptive and Anisotropic Gaussian Filter. In *Vision, Modeling and Visualization*, 203–210.

Osher, S., and Fedkiw, R. P. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, NY.

Pauly, M., and Gross, M. 2001. Spectral Processing of Point-Sampled Geometry. In *Proceedings of ACM SIGGRAPH 2001*, 379–386.

Peng, J., Strela, V., and Zorin, D. 2001. A Simple Algorithm for Surface Denoising. In *Proceedings of IEEE Visualization 2001*, 107–112.

Perona, P., and Malik, J. 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE PAMI 12*, 7, 629–639.

Rusinkiewicz, S., Hall-Holt, O., and Levoy, M. 2002. Real-Time 3D Model Acquisition. *ACM Trans. Gr. 21*, 3, 438–446.

Smith, S., and Brady, J. 1997. SUSAN - a new approach to low level image processing. *IJCV 23*, 45–78.

Tasdizen, T., Whitaker, R., Burchard, P., and Osher, S. 2002. Geometric Surface Smoothing via Anisotropic Diffusion of Normals. In *Proceedings, IEEE Visualization 2002*, 125–132.

Taubin, G. 1995. A Signal Processing Approach to Fair Surface Design. In *Proceedings of SIGGRAPH 95*, 351–358.

Taubin, G. 2001. Linear Anisotropic Mesh Filtering. Tech. Rep. IBM Research Report RC2213.

Tomasi, C., and Manduchi, R. 1998. Bilateral Filtering for Gray and Color Images. In *Proc. IEEE Int. Conf. on Computer Vision*, 836–846.

Wood, Z., Hoppe, H., Desbrun, M., and Schröder, P. 2002. Isosurface Topology Simplification. http://www.multires.caltech.edu/pubs/.

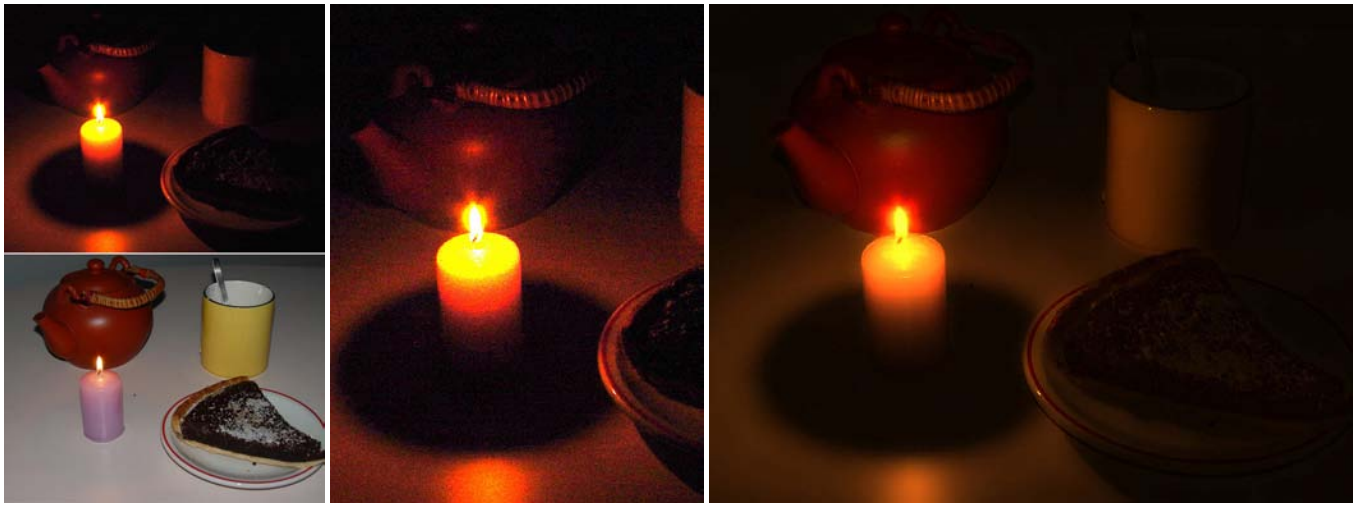
Zhang, H., and Fiume, E. L. 2002. Mesh Smoothing with Shape or Feature Preservation. In *Advances in Modeling, Animation, and Rendering, J. Vince and R. Earnshaw, editors*, 167–182.

Zwicker, M., Pauly, M., Knoll, O., and Gross, M. 2002. Pointshop 3D: An Interactive System for Point-Based Surface Editing. *ACM Trans. Gr. 21*, 3, 322–329.

Flash Photography Enhancement via Intrinsic Relighting

Elmar Eisemann*
MIT / ARTIS‡GRAVIR/IMAG-INRIA

Frédo Durand
MIT



(a) (b) (c)

Figure 1: (a) Top: Photograph taken in a dark environment, the image is noisy and/or blurry. Bottom: Flash photography provides a sharp but flat image with distracting shadows at the silhouette of objects. (b) Inset showing the noise of the available-light image. (c) Our technique merges the two images to transfer the ambiance of the available lighting. Note the shadow of the candle on the table.

Abstract

We enhance photographs shot in dark environments by combining a picture taken with the available light and one taken with the flash. We preserve the ambiance of the original lighting and insert the sharpness from the flash image. We use the bilateral filter to decompose the images into detail and large scale. We reconstruct the image using the large scale of the available lighting and the detail of the flash. We detect and correct flash shadows. This combines the advantages of available illumination and flash photography.

Keywords: Computational photography, flash photography, relighting, tone mapping, bilateral filtering, image fusion

1 Introduction

Under dark illumination, a photographer is usually faced with a frustrating dilemma: to use the flash or not. A picture relying on the available light usually has a warm atmosphere, but suffers from noise and blur (Fig. 1(a) top and (b)). On the other hand, flash photography causes three unacceptable artifacts: red eyes, flat and harsh lighting, and distracting sharp shadows at silhouettes (Fig. 1(a) bottom). While much work has addressed red-eye removal [Zhang and Lenders 2000; Gaubatz and Ulichney 2002], the harsh lighting and shadows remain a major impediment.

We propose to combine the best of the two lightings by taking two successive photographs: one with the available lighting only, and one with the flash. We then recombine the two pictures and take advantage of the main qualities of each one (Fig. 1(c)). Our central tool is a decomposition of an image into a large-scale layer that is assumed to contain the variation due to illumination, and a small-scale layer containing albedo variations.

Related work Most work on flash photography has focused on red-eye removal [Zhang and Lenders 2000; Gaubatz and Ulichney 2002]. Many cameras use a pre-flash to prevent red eyes. Professional photographers rely on off-centered flash and indirect lighting to prevent harsh lighting and silhouette shadows.

Our work is related to the continuous flash by Hoppe and Toyama [2003]. They use a flash and a no-flash picture and combine them linearly. The image-stack interface by Cohen et al. [2003] provides additional control and the user can spatially vary the blending. Raskar et al. [2004] and Akers et al. [2003] fuse images taken with different illuminations to enhance context and legibility. DiCarlo et al. [2001] use a flash and a no-flash photograph for white balance.

Multiple-exposure photography allows for high-dynamic-range images [Mann and Picard 1995; Debevec and Malik 1997]. New techniques also compensate for motion between frames [Kang et al. 2003; Ward 2004]. Note that multiple-exposure techniques are different from our flash-photography approach. They operate on the same lighting in all pictures and invert a non-linear and clamped

*e-mail: eisemann@graphics.csail.mit.edu, fredo@mit.edu

†ARTIS is a research project in the GRAVIR/IMAG laboratory, a joint unit of CNRS, INPG, INRIA and UJF.

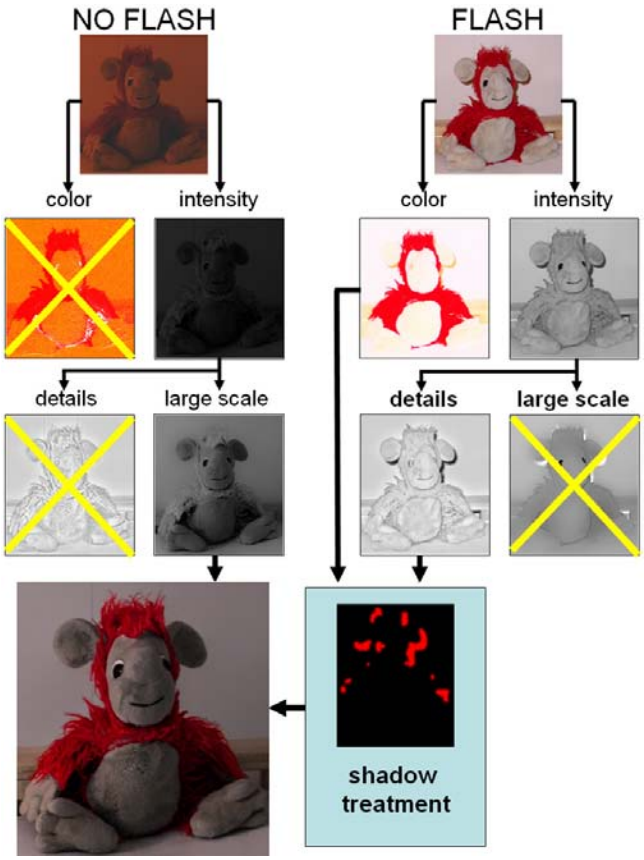


Figure 2: We take two images with the available light and the flash respectively. We decouple their color, large-scale and detail intensity. We correct flash shadows. We re-combine the appropriate layers to preserve the available lighting but gain the sharpness and detail from the flash image.

response. In contrast, we have a quite-different lighting in the two images and try to extract the lighting ambiance from the no-flash picture and combine it with the fine detail of the flash picture.

We build on local tone-mapping techniques that decompose an image into two or more layers that correspond to small- and large-scale variations, e.g. [Chiu et al. 1993; Jobson et al. 1997; Tumblin and Turk 1999; DiCarlo and Wandell 2000; Durand and Dorsey 2002; Reinhard et al. 2002; Ashikhmin 2002; Choudhury and Tumblin 2003]. Only the contrast of the large scales is reduced, thereby preserving detail.

These methods can be interpreted in terms of *intrinsic images* [Tumblin et al. 1999; Barrow and Tenenbaum 1978]. The large scale can be seen as an estimate of illumination, while the detail corresponds to albedo [Oh et al. 2001]. Although this type of decoupling is hard [Barrow and Tenenbaum 1978; Weiss 2001; Tappen et al. 2003], tone mapping can get away with a coarse approximation because the layers are eventually recombined. We exploit the same approach to decompose our flash and no-flash images.

A wealth of efforts has been dedicated to relighting, e.g. [Marschner and Greenberg 1997; Sato et al. 1999; Yu et al. 1999]. Most methods use acquired geometry or a large set of input images. In contrast, we perform lighting transfer from only two images.

In this volume, Petschnigg et al. [2004] present a set of techniques based on flash/no-flash image pairs. Their decoupling approach shares many similarities with our work, in particular the use of the bilateral filter. The main difference between the two approaches lies in the treatment of flash shadows.

2 Image decoupling for flash relighting

Our approach is summarized in Fig. 2. We take two photos, with and without the flash. We align the two images to compensate for camera motion between the snapshots. We detect the shadows cast by the flash and correct color using local white balance. We finally perform a non-linear decomposition of the two images into large-scale and detail layers, and we recombine them appropriately.

We first present our basic technique before discussing shadow correction in Section 3. We then introduce more advanced reconstruction options in Section 4 and present our results in Section 5.

Taking the photographs The two photographs with and without the flash should be taken as rapidly as possible to avoid motion of either the photographer or subject. The response curve between the two exposures should ideally be known for better relative radiometric calibration, but this is not a strict requirement. Similarly, we obtain better results when the white balance can be set to manual. In the future, we foresee that taking the two images in a row will be implemented in the firmware of the camera. To perform our experiments, we have used a tripod and a remote control (Fig. 1 and 8) and hand-held shots (Fig. 2, 5, 7). The latter in particular requires good image alignment. In the rest of this paper, we assume that the images are normalized so that the flash image is in $[0,1]$.

The registration of the two images is not trivial because the lighting conditions are dramatically different. Following Kang et al. [2003], we compare the image gradients rather than the pixel values. We use a low-pass filter with a small variance (2 pixels) to smooth-out the noise. We keep only the 5% highest gradients and we reject gradients in regions that are too dark and where information is not reliable. We use a pyramidal refinement strategy similar to Ward [2004] to find the transformation that minimizes the gradients that were kept. More advanced approaches could be used to compensate for subject motion, e.g. [Kang et al. 2003].

Bilateral decoupling We first decouple the images into intensity and color (Fig. 2). Assume we use standard formulas, although we show in the appendix that they can be improved in our context. The color layer simply corresponds to the original pixel values divided by the intensity. In the rest of the paper, we use I^f and I^{nf} for the intensity of the flash and no-flash images.

We then want to decompose each image into layers corresponding to the illumination and the sharp detail respectively. We use the bilateral filter [Tomasi and Manduchi 1998; Smith and Brady 1997] that smoothes an image but respects sharp features, thereby avoiding halos around strong edges [Durand and Dorsey 2002].

The bilateral filter is defined as a weighted average where the weights depend on a Gaussian f on the spatial location, but also on a weight g on the pixel difference. Given an input image I, The output of the bilateral filter for a pixel s is:

$$J_s = \frac{1}{k(s)} \sum_{p \in \Omega} f(p-s)\ g(I_p - I_s)\ I_p, \qquad (1)$$

where $k(s)$ is a normalization: $k(s) = \sum_{p \in \Omega} f(p-s)\ g(I_p - I_s)$. In practice, g is a Gaussian that penalizes pixels across edges that have large intensity differences. This filter was used by Oh et al. [2001] for image editing and by Durand et al. for tone mapping [2002].

We use the fast bilateral filter where the non-linear filter is approximated by a set of convolutions [Durand and Dorsey 2002]. We perform computation in the $\log_{10}$ domain to respect intensity ratios. The output of the filter provides the log of the large-scale layer. The detail layer is deduced by a division of the intensity by the large-scale layer (subtraction in the log domain). We use a spatial variance σ_f of 1.5% of the images diagonal. For the intensity influence g, we use $\sigma_g = 0.4$, following Durand and Dorsey [2002].

Figure 3: Basic reconstruction and shadow correction. The flash shadow on the right of the face and below the ear need correction. In the naïve correction, note the yellowish halo on the right of the character and the red cast below its ear. See Fig. 4 for a close up.


Figure 4: Enlargement of Fig. 3. Correction of smooth shadows. From left to right: no flash, flash, naïve white balance, our color correction

Reconstruction Ignoring the issue of shadows for now, we can recombine the image (Fig. 2). We use the detail and color layer of the flash image because it is sharper and because white balance is more reliable. We use the large-scale layer of the no-flash picture in order to preserve the mood and tonal modeling of the original lighting situation. The layers are simply added in the log domain. Fig. 3 illustrates the results from our basic approach. The output combines the sharpness of the flash image with the tonal modeling of the no-flash image.

For dark scenes, the contrast of the large scale needs to be enhanced. This is the opposite of contrast reduction [Durand and Dorsey 2002]. We set a target contrast for the large-scale layer and scale the range of log values accordingly. The low quantization from the original image does not create artifacts because the bilateral filter results in a piecewise-smooth large-scale layer.

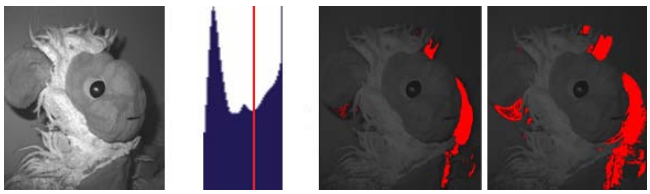
In addition, we compute the white balance between the two images by computing the weighted average of the three channels with stronger weights for bright pixels with a white color in the flash image. We then take the ratios w_r, w_g, w_b as white-balance coefficients. This white balance can be used to preserve the warm tones of the available light. In practice, the color cast of the no-flash image is usually too strong and we only apply it partially using w^t where t is usually 0.2.

We must still improve the output in the flash shadow. While their intensity is increased to match the large scale of the no-flash image, there is a distinct color cast and noise. This is because, by definition, these areas did not receive light from the flash and inherit from the artifacts of the no-flash image. A ring flash might reduce these artifacts, but for most cameras, we must perform additional processing to alleviate them.

3 Shadow treatment

In order to correct the aforementioned artifacts, we must detect the pixels that lie in shadow. Pixels in the umbra and penumbra have different characteristics and require different treatments. After detection, we correct color and noise in the shadows. The correction applied in shadow is robust to false positives; Potential detection errors at shadow boundaries do not create visible artifacts.

Umbra detection We expect the difference image ΔI between flash and no-flash to tell how much additional light was received from the flash. When the images are radiometrically calibrated,


ΔI histogram of ΔI detected umbra with penumbra
Figure 5: Shadow detection

ΔI is exactly the light received from the flash. However, shadows do not always correspond to $\Delta I = 0$ because of indirect lighting. While shadow pixels always correspond to the lowest values of ΔI, the exact cutoff is scene-dependent.

We use histogram analysis to compute a threshold $t_{\Delta I}$ that determines umbra pixels. Shadows correspond to a well-marked mode in the histogram of ΔI. While the additional light received by parts of the scene lit by the flash varies with albedo, distance and normal, the parts in shadow are only indirectly illuminated and receive a more uniform and very low amount of light.

We compute the histogram of pixels ΔI. We use 128 bins and smooth it with a Gaussian blur of variance two bins. We start with a coarse threshold of 0.2 and discard all pixels where ΔI is above this value. We then use the first local minimum of the histogram before 0.2 as our threshold for shadows detection (Fig. 5). This successfully detects pixels in the umbra. However, pixels in the penumbra correspond to a smoother gradation and cannot be detected with our histogram technique. This is why we use a complementary detection based on the gradient at shadow boundaries.

Penumbra detection Shadow boundaries create strong gradients in the flash image that do not correspond to gradients in the no-flash image. We detect these pixels using two criteria: the gradients difference, and connectedness to umbra pixels.

We compute the magnitude of the gradient ∇I^f and ∇I^{nf} and smooth it with a Gaussian of variance 2 pixels to remove noise. We identify candidate penumbra pixels as pixels where the gradient is stronger in the flash image. We then keep only pixels that are "close" to umbra pixels, that is, such that at least one of their neighbors is in umbra. In practice, we use a square neighborhood of size 1% of the photo's diagonal. This computation can be performed efficiently by convolving the binary umbra map with a box filter.

We also must account for shadows cast by tiny objects such as

pieces of fur, since these might have a pure penumbra without umbra. We use a similar strategy and consider as shadow pixels that have a large number of neighbors with higher gradient in the flash image. We use a threshold of 80% on a square neighborhood of size 0.7% of the photo's diagonal.

We have observed that the parameters concerning the penumbra are robust with respect to the scene. The image-space size of the penumbra does not vary much in the case of flash photography because the distance to the light is the same as the distance to the image plane. The variation of penumbra size (ratio of blocker-receiver distances) and perspective projection mostly cancel each other.

Flash detail computation Now that we have detected shadows, we can refine the decoupling of the flash image. We exploit the shadow mask to exclude shadow pixels from the bilateral filtering. This results in a higher-quality detail layer for the flash image because it is not affected by shadow variation.

Color and noise correction Color in the shadow cannot simply be corrected using white balance [DiCarlo et al. 2001] for two reasons. First, shadow areas receive different amounts of indirect light from the flash, which results in hybrid color cast affected by the ambient lighting and color bleeding from objects. Second, the no-flash image often lacks information in the blue channel due to the yellowish lighting and poor sensitivity of sensors in the small wavelengths. Fig. 3 illustrates the artifacts caused by a global white balance of the shadow pixels.

In order to address these issues, we use a *local* color correction that copies colors from illuminated regions in the flash image. For example, in Fig. 3, a shadow falls on the wall, sofa frame and jacket. For all these objects, we have pixels with the same intrinsic color in the shadow and in the illuminated region.

Inspired by the bilateral filter, we compute the color of a shadow pixel as a weighted average of its neighbors in the flash image $\mathbf{I}^f$ (with full color information). The weight depends on three terms: a spatial Gaussian, a Gaussian on the color similarity in $\mathbf{I}^{nf}$, and a binary term that excludes pixels in shadow (Fig. 6). We perform computation only on the color layer (see Fig. 2) in Luv. We use σ_f of 2.5% of the photo's diagonal for the spatial Gaussian and $\sigma_g = 0.01$ for the color similarity. As described by Durand and Dorsey [2002] we use the sum of the weights k as a measure of pixel uncertainty. We discard color correction if k is below a threshold. In practice, we use a smooth feathering between 0.02 and 0.002 to avoid discontinuities.

Recall that the large-scale layer of intensity is obtained from the no-flash image and is not affected by shadows. In the shadow, we do not use the detail layer of the flash image because it could be affected by high-frequencies due to shadow boundary. Instead, we copy the detail layer of the no-flash image, but we correct its noise level. For this we scale the no-flash detail to match the variance of the flash detail outside shadow regions.

In order to ensure continuity of the shadow correction, we use feathering at the boundary of the detected shadow: We follow a linear ramp and update pixels as a linear combination of the original and shadow-corrected value. Fig. 3 and 4 show the results of our shadow correction. It is robust to false shadow positives because it simply copies colors from the image. If a pixel is wrongly classified in shadow, its color and noise are preserved as long as there are other pixels with similar color that were not classified in shadow.

4 Advanced decoupling

The wealth of information provided by the pair of images can be further exploited to enhance results for very dark situations and more advanced lighting transfer.

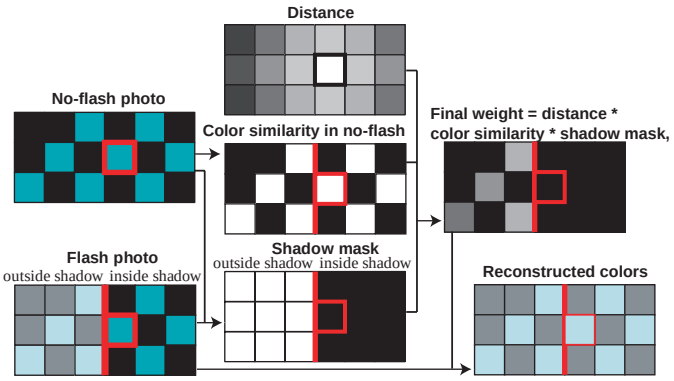


Figure 6: For a pixel in the flash shadow, the color layer is computed as a weighted average of non-shadow colors. The weights depend on three terms: distance, similarity in the no-flash image and a binary shadow mask.

When the no-flash picture is too dark, the edge-preserving property of the bilateral filter is not reliable, because noise level is in the range of the signal level. Similar to the technique we use for color correction, we can use the flash image as a similarity measure between pixels. We propose a *cross-bilateral filter*[1] where we modify Eq. 1 for the no-flash image and compute the edge-preserving term g as a function of the flash-image values:

$$J_s^{nf} = \frac{1}{k(s)} \sum_{p \in \Omega} f(p-s)\ g(I_p^f - I_s^f)\ I_p^{nf}, \qquad (2)$$

This preserves edges although they are not really present in the no-flash image. Shadow correction can however not be performed because the shadow edges of the flash picture are transferred by the g term. Fig. 1 exploits cross-bilateral decomposition.

The large-scale layer of the flash image can also be exploited to drive the reconstruction. The distance falloff makes objects closer to the camera brighter. We use this pseudo-distance to emphasize the main object. We use a shadow-corrected version of ΔI as our pseudo-distance. Pixels in shadow are assigned a pseudo-distance using a bilateral-weighted average of their neighbors where similarity is defined in the no-flash image. The principle is to multiply the large scale of the no-flash image by the pseudo-distance. This can be performed using a user-provided parameter. Pseudo-distance was used in Fig. 8.

5 Results and discussion

Our technique takes about 50 seconds on a 866 MHz Pentium 3 for a 1280x960 image. The majority of the time is spent in the color correction, because this bilateral filter cannot be efficiently piecewise-linearized [Durand and Dorsey 2002] since it operates on the three channels. Images such as Fig. 8 that do not include shadow correction take about 10 seconds.

Fig 1, 3, 7 and 8 illustrate our results. The ambiance of the available light is preserved and the color, sharpness and detail of the flash picture is gained. In our experience, the main cause of failure of our technique is poor quality (not quantity) of available lighting. For example, if the light is behind the subject, the relighting results in an under-exposed subject. We found, however, that it is not hard to outperform the poor lighting of the flash. It is well known that lighting along the optical axis does not result in good tonal modeling. In contrast, Fig. 2 and 8 present a nice 3/4 side lighting. We

[1] Petschnigg et al. [2004] propose a similar approach that they call *joint* bilateral filter.

no-flash flash result

Figure 7: The flash lighting results in a flat image. In our result, light seems to be coming from the window to the right.

received conflicting feedback on Fig. 7, which shows that image quality is a subjective question. In this image, the light is coming from the 3/4 back, which is an unusual lighting for a photograph. Some viewers appreciate the strong sense of light it provides, while others object to the lack of tonal modeling.

Another cause of failure is overexposure of the flash, leading to a flat detail layer. In this situation, the detail information is neither in the no-flash (due to noise) nor in the flash photo (due to saturation).

Shadow detection works best when the depth range is limited. Distant objects do not receive light from the flash and are detected in shadow. While this is technically correct, this kind of shadow due to falloff does not necessitate the same treatment as cast shadow. Fortunately, our color correction is robust to false positives and degrades to identity in these cases (although transition areas could potentially create problems). Similarly, black objects can be detected as shadows, but this does not affect quality since they are black in the two images and remain black in the output. Light flares can cause artifacts by brightening shadow pixels. The method by Ward [2004] could alleviate this problem.

We have used our algorithms with images from a variety of cameras including a Sony Mavica MVC-CD400 (Fig. 1), a Nikon Coolpix 4500 (all other images), a Nikon D1 and a Kodak DC4800 (not shown in the paper). The choice of the camera was usually dictated by availability at the time of the shot. The specifications that affected our approach are the noise level, the flexibility of control, the accuracy of flash white balance, and compression quality. For example, the Kodak DC4800 exhibited strong JPEG artifacts for dark images, which required the use of the cross-bilateral filter.

The need for the cross-bilateral filter was primarily driven by the SNR in the no-flash picture. The Kodak DC4800 has higher noise levels because it is old. Despite its age, the size of its photosites allows the Nikon D1 to take images in dark conditions. In addition, the use of the RAW format with 12 bits/channel allows for higher precision in the flash image (the lower bits of the no-flash image are dominated by noise). However, with the sensitivity at 1600 equivalent ISO, structured noise makes cross-bilateral filtering necessary.

Figure 8: The tonal modeling on the cloth and face are accurately transferred from the available lighting. The main subject is more visible in the result than he was in the original image.

6 Conclusions and future work

We have presented a method that improves the lighting and ambiance of flash photography by combining a picture taken with the flash and one using the available lighting. Using a feature-preserving filter, we estimate what can be seen as intrinsic layers of the image and use them to transfer the available illumination to the flash picture. We detect shadows cast by the flash and correct their color balance and noise level. Even when the no-flash picture is extremely noisy, our method successfully transfers lighting due to the use of the flash image to perform edge-preserving filtering.

The method could be tailored to particular cameras by fine-tuning parameters such as σ_g based on a sensor-noise model. Traditional red-eye removal could benefit from the additional information provided by the pair of images. Texture synthesis and inpainting could be used to further improve shadow correction. Ideally, we want to alleviate the disturbance of the flash and we are considering the use of infrared illumination. This is however challenging because it requires different sensors and these wavelengths provide limited resolution and color information.

The difference of the flash and no-flash images contains much information about the 3D scene. Although a fundamental ambiguity remains between albedo, distance and normal direction, this additional information could greatly expand the range and power of picture enhancement such as tone mapping, super-resolution, photo editing, and image based-modeling.

Acknowledgments We acknowledge support from an NSF CISE Research Infrastructure Award (EIA-9802220) and a Desh-

pande Center grant. Elmar Eisemann's stay at MIT was supported by MIT-France and ENS Paris. Many thanks to the reviewers, Joëlle Thollot, Marc Lapierre, Ray Jones, Eric Chan, Martin Eisemann, Almuth Biard, Shelly Levy-Tzedek, Andrea Pater and Adel Hanna.

Appendix: Intensity-Color decoupling Traditional approaches rely on linear weighted combinations of R, G, and B for intensity estimation. While these formulae are valid from a color-theory point of view, they can be improved for illumination-albedo decoupling. Under the same illumination, a linear intensity computation results in lower values for primary-color albedo (in particular blue) than for white objects. As a result, the intensity transfer might overcompensate as shown in Fig. 9(left) where the red fur becomes too bright. To alleviate this, we use the channels themselves as weights in the linear combination:
$I = \frac{R}{R+G+B}R + \frac{G}{R+G+B}G + \frac{B}{R+G+B}B$.

In practice, we use the channels of the flash image as weight for both pictures to ensure consistency between the two decoupling operations. The formula can also be used with tone mapping operators for higher color fidelity.

Figure 9: The computation of intensity from RGB can greatly affect the final image. Left: with linear weights, the red pixels of the fur become too bright. Right: using our non-linear formula.

References

AKERS, LOSASSO, KLINGNER, AGRAWALA, RICK, AND HANRAHAN. 2003. Conveying shape and features with image-based relighting. In *Visualization*.

ASHIKHMIN. 2002. A tone mapping algorithm for high contrast images. In *Eurographics Workshop on Rendering*, 145–156.

BARROW, AND TENENBAUM. 1978. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*. Academic Press.

CHIU, HERF, SHIRLEY, SWAMY, WANG, AND ZIMMERMAN. 1993. Spatially nonuniform scaling functions for high contrast images. In *Graphics Interface*.

CHOUDHURY, AND TUMBLIN. 2003. The trilateral filter for high contrast images and meshes. In *Eurographics Symposium on Rendering*.

COHEN, COLBURN, AND DRUCKER. 2003. Image stacks. Tech. Rep. 40, MSR.

DEBEVEC, AND MALIK. 1997. Recovering high dynamic range radiance maps from photographs. In *Proc. SIGGRAPH*.

DICARLO, J., AND WANDELL, B. 2000. Rendering high dynamic range images. *Proc. SPIE: Image Sensors 3965*, 392–401.

DICARLO, J. M., XIAO, F., AND WANDELL, B. A. 2001. Illuminating illumination. In *9th Color Imaging Conference*, 27–34.

DURAND, AND DORSEY. 2002. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. on Graphics 21*, 3.

GAUBATZ, AND ULICHNEY. 2002. Automatic red-eye detection and correction. In *IEEE Int. Conf. on Image Processing*.

HOPPE, AND TOYAMA. 2003. Continuous flash. Tech. Rep. 63, MSR.

JOBSON, RAHMAN, AND WOODELL. 1997. A multi-scale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Trans. on Image Processing 6*, 965–976.

KANG, UYTTENDAELE, WINDER, AND SZELISKI. 2003. High dynamic range video. *ACM Trans. on Graphics 22*, 3.

MANN, AND PICARD. 1995. Being 'undigital' with digital cameras: Extending dynamic range by combining differently exposed pictures. In *Proc. IS&T 46th ann. conference*.

MARSCHNER, AND GREENBERG. 1997. Inverse lighting for photography. In *Proc. IS&T/SID 5th Color Imaging Conference*.

OH, CHEN, DORSEY, AND DURAND. 2001. Image-based modeling and photo editing. In *Proc. SIGGRAPH*.

PETSCHNIGG, AGRAWALA, HOPPE, SZELISKI, COHEN, AND TOYAMA. 2004. Digital photography with flash and no-flash image pairs. *ACM Trans. on Graphics in this volume*.

RASKAR, ILIE, AND YU. 2004. Image fusion for context enhancement. In *Proc. NPAR*.

REINHARD, STARK, SHIRLEY, AND FERWERDA. 2002. Photographic tone reproduction for digital images. *ACM Trans. on Graphics 21*, 3.

SATO, SATO, AND IKEUCHI. 1999. Illumination distribution from brightness in shadows: Adaptive estimation of illumination distribution with unknown reflectance properties in shadow regions. In *ICCV*.

SMITH, S. M., AND BRADY, J. M. 1997. SUSAN - a new approach to low level image processing. *IJCV 23*, 45–78.

TAPPEN, M. F., FREEMAN, W. T., AND ADELSON, E. H. 2003. Recovering intrinsic images from a single image. In *NIPS*.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *ICCV*, 836–846.

TUMBLIN, AND TURK. 1999. Lcis: A boundary hierarchy for detail-preserving contrast reduction. In *Proc. SIGGRAPH*.

TUMBLIN, HODGINS, AND GUENTER. 1999. Two methods for display of high contrast images. *ACM Trans. on Graphics 18*, 1.

WARD. 2004. Fast, robust image registration for compositing high dynamic range photographs from handheld exposures. *J. of Graphics Tools 8*, 2.

WEISS. 2001. Deriving intrinsic images from image sequences. In *ICCV*.

YU, DEBEVEC, MALIK, AND HAWKINS. 1999. Inverse global illumination: Recovering reflectance models of real scenes from photographs from. In *Proc. SIGGRAPH*.

ZHANG, AND LENDERS. 2000. Knowledge-based eye detection for human face recognition,. In *Conf. on Knowledge-based Intelligent Systems and Allied Technologies,*.

Digital Photography with Flash and No-Flash Image Pairs

Georg Petschnigg Maneesh Agrawala Hugues Hoppe
Richard Szeliski Michael Cohen Kentaro Toyama
Microsoft Corporation

Orig. (top) Detail Transfer (bottom) Flash No-Flash Detail Transfer with Denoising

Figure 1: This candlelit setting from the wine cave of a castle is difficult to photograph due to its low light nature. A flash image captures the high-frequency texture and detail, but changes the overall scene appearance to cold and gray. The no-flash image captures the overall appearance of the warm candlelight, but is very noisy. We use the detail information from the flash image to both reduce noise in the no-flash image and sharpen its detail. Note the smooth appearance of the brown leather sofa and crisp detail of the bottles. For full-sized images, please see the supplemental DVD or the project website http://research.microsoft.com/projects/FlashNoFlash.

Abstract

Digital photography has made it possible to quickly and easily take a pair of images of low-light environments: one with flash to capture detail and one without flash to capture ambient illumination. We present a variety of applications that analyze and combine the strengths of such flash/no-flash image pairs. Our applications include denoising and detail transfer (to merge the ambient qualities of the no-flash image with the high-frequency flash detail), white-balancing (to change the color tone of the ambient image), continuous flash (to interactively adjust flash intensity), and red-eye removal (to repair artifacts in the flash image). We demonstrate how these applications can synthesize new images that are of higher quality than either of the originals.

Keywords: Noise removal, detail transfer, sharpening, image fusion, image processing, bilateral filtering, white balancing, red-eye removal, flash photography.

1 Introduction

An important goal of photography is to capture and reproduce the visual richness of a real environment. Lighting is an integral aspect of this visual richness and often sets the mood or atmosphere in the photograph. The subtlest nuances are often found in low-light conditions. For example, the dim, orange hue of a candlelit restaurant can evoke an intimate mood, while the pale blue cast of moonlight can evoke a cool atmosphere of mystery.

When capturing the natural ambient illumination in such low-light environments, photographers face a dilemma. One option is to set a long exposure time so that the camera can collect enough light to produce a visible image. However, camera shake or scene motion during such long exposures will result in motion blur. Another option is to open the aperture to let in more light. However, this approach reduces depth of field and is limited by the size of the lens. The third option is to increase the camera's gain, which is controlled by the ISO setting. However, when exposure times are short, the camera cannot capture enough light to accurately estimate the color at each pixel, and thus visible image noise increases significantly.

Flash photography was invented to circumvent these problems. By adding artificial light to nearby objects in the scene, cameras with flash can use shorter exposure times, smaller apertures, and less sensor gain and still capture enough light to produce relatively sharp, noise-free images. Brighter images have a greater signal-to-noise ratio and can therefore resolve detail that would be hidden in the noise in an image acquired under ambient illumination. Moreover, the flash can enhance surface detail by illuminating surfaces with a crisp point light source. Finally, if one desires a white-balanced image, the known flash color greatly simplifies this task.

As photographers know, however, the use of flash can also have a negative impact on the lighting characteristics of the environment. Objects near the camera are disproportionately brightened, and the mood evoked by ambient illumination may be destroyed. In addition, the flash may introduce unwanted artifacts such as red eye, harsh shadows, and specularities, none of which are part of the natural scene. Despite these drawbacks, many amateur photographers use flash in low-light environments, and consequently, these snapshots rarely depict the true ambient illumination of such scenes.

Today, digital photography makes it fast, easy, and economical to take a pair of images of low-light environments: one with flash to capture detail and one without flash to capture ambient illumination. (We sometimes refer to the no-flash image as the ambient image.) In this paper, we present a variety of techniques that

analyze and combine features from the images in such a flash/no-flash pair:

Ambient image denoising: We use the relatively noise-free flash image to reduce noise in the no-flash image. By maintaining the natural lighting of the ambient image, our approach creates an image that is closer to the look of the real scene.

Flash-to-ambient detail transfer: We transfer high-frequency detail from the flash image to the denoised ambient image, since this detail may not exist in the original ambient image.

White balancing: The user may wish to simulate a whiter illuminant while preserving the "feel" of the ambient image. We exploit the known flash color to white-balance the ambient image, rather than relying on traditional single-image heuristics.

Continuous flash intensity adjustment: We provide continuous interpolation control between the image pair so that the user can interactively adjust the flash intensity. The user can even extrapolate beyond the original ambient and flash images.

Red-eye correction: We perform red-eye detection by considering how the color of the pupil changes between the ambient and flash images.

While many of these problems are not new, the primary contribution of our work is to show how to exploit information in the flash/no-flash pair to improve upon previous techniques[1]. One feature of our approach is that the manual acquisition of the flash/no-flash pair is relatively straightforward with current consumer digital cameras. We envision that the ability to capture such pairs will eventually move into the camera firmware, thereby making the acquisition process even easier and faster.

One recurring theme of recent computer graphics research is the idea of taking multiple photographs of a scene and combining them to synthesize a new image. Examples of this approach include creating high dynamic range images by combining photographs taken at different exposures [Debevec and Malik 1997; Kang et al. 2003], creating mosaics and panoramas by combining photographs taken from different viewpoints [e.g. Szeliski and Shum 1997], and synthetically relighting images by combining images taken under different illumination conditions [Haeberli 1992; Debevec et al. 2000; Masselus et al. 2002; Akers et al. 2003; Agarwala et al. 2004]. Although our techniques involve only two input images, they share the similar goal of synthesizing a new image that is of better quality than any of the input images.

2 Background on Camera Noise

The intuition behind several of our algorithms is that while the illumination from a flash may change the appearance of the scene, it also increases the signal-to-noise ratio (SNR) in the flash image and provides a better estimate of the high-frequency detail.

As shown in Figure 2(a), a brighter image signal contains more noise than a darker signal. However, the slope of the curve is less than one, which implies that the signal increases faster than the noise and so the SNR of the brighter image is better. While the flash does not illuminate the scene uniformly, it does significantly increase scene brightness (especially for objects near the camera) and therefore the flash image exhibits a better SNR than the ambient image.

As illustrated in Figure 2(b), the improvement in SNR in a flash image is especially pronounced at higher frequencies. Properly exposed image pairs have similar intensities after passing through

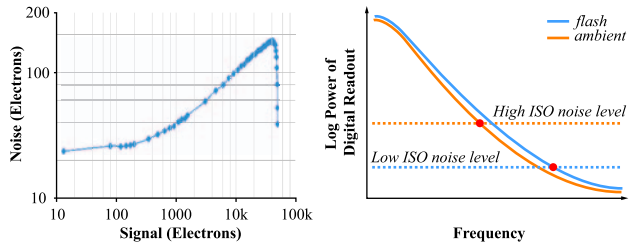


Figure 2: (a-left) Noise vs. signal for a full-frame Kodak CCD [2001]. Since the slope is less than one, SNR increases at higher signal values. (b-right) The digital sensor produces similar log power spectra for the flash and ambient images. However, the noise dominates the signal at a lower frequency in the high-ISO ambient image than in the low-ISO flash image.

the imaging system (which may include aperture, shutter/flash duration, and camera gain). Therefore their log power spectra are roughly the same. However, the noise in the high-ISO ambient image is greater than in the low-ISO flash image because the gain amplifies the noise. Since the power spectrum of most natural images falls off at high frequencies, whereas that of the camera noise remains uniform (i.e. assuming white noise), noise dominates the signal at a much lower frequency in the ambient image than in the flash image.

3 Acquisition

Procedure. We have designed our algorithms to work with images acquired using consumer-grade digital cameras. The main goal of our acquisition procedure is to ensure that the flash/no-flash image pair capture exactly the same points in the scene. We fix the focal length and aperture between the two images so that the camera's focus and depth-of-field remain constant. Our acquisition procedure is as follows:

1. Focus on the subject, then lock the focal length and aperture.
2. Set exposure time Δt and ISO for a good exposure.
3. Take the ambient image A.
4. Turn on the flash.
5. Adjust the exposure time Δt and ISO to the smallest settings that still expose the image well.
6. Take the flash image F.

A rule of thumb for handheld camera operation is that exposure times for a single image should be under $\frac{1}{30}$ s for a 30mm lens to prevent motion blur. In practice, we set the exposure times for both images to $\frac{1}{60}$ s or less so that under ideal circumstances, both images could be shot one after another within the $\frac{1}{30}$ s limit on handheld camera operation. Although rapidly switching between flash and non-flash mode is not currently possible on consumer-grade cameras, we envision that this capability will eventually be included in camera firmware. Most of the images in this paper were taken with a Canon EOS Digital Rebel.

We acquire all images in a RAW format and then convert them into 16-bit TIFF images. By default, the Canon conversion software performs white balancing, gamma correction and other non-linear tone-mapping operations to produce perceptually pleasing images with good overall contrast. We apply most of our algorithms on these non-linear images in order to preserve their high-quality tone-mapping characteristics in our final images.

Registration. Image registration is not the focus of our work and we therefore acquired most of our pairs using a tripod setup. Nevertheless we recognize that registration is important for images taken with handheld cameras since changing the camera settings (i.e. turning on the flash, changing the ISO, etc.) often results in camera motion. For the examples shown in Figure 11 we took the photographs without a tripod and then applied the registration technique of Szeliski and Shum [1997] to align them.

[1] In concurrent work, Eisemann and Durand [2004] have developed techniques similar to ours for transferring color and detail between the flash/no-flash images.

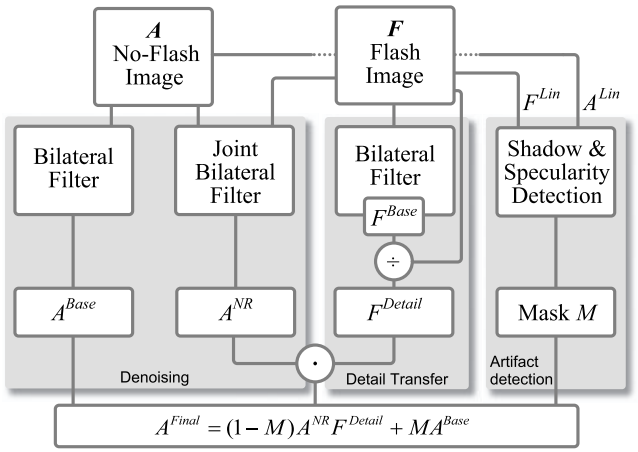


Figure 3: Overview of our algorithms for denoising, detail transfer, and flash artifact detection.

While we found this technique works well, we note that flash/no-flash images do have significant differences due to the change in illumination, and therefore robust techniques for registration of such image pairs deserve further study.

Linearization. Some of our algorithms analyze the image difference $F - A$ to infer the contribution of the flash to the scene lighting. To make this computation meaningful, the images must be in the same linear space. Therefore we sometimes set our conversion software to generate linear TIFF images from the RAW data. Also, we must compensate for the exposure differences between the two images due to ISO settings and exposure times Δt. If A'^{Lin} and F^{Lin} are the linear images output by the converter utility, we put them in the same space by computing:

$$A^{Lin} = A'^{Lin} \frac{ISO_F \Delta t_F}{ISO_A \Delta t_A}. \quad (1)$$

Note that unless we include the superscript *Lin, F* and *A* refer to the non-linear versions of the images.

4 Denoising and Detail Transfer

Our denoising and detail transfer algorithms are designed to enhance the ambient image using information from the flash image. We present these two algorithms in Sections 4.1 and 4.2. Both algorithms assume that the flash image is a good local estimator of the high frequency content in the ambient image. However, this assumption does not hold in shadow and specular regions caused by the flash, and can lead to artifacts. In Section 4.3, we describe how to account for these artifacts. The relationships between the three algorithms are depicted in Figure 3.

4.1 Denoising

Reducing noise in photographic images has been a long-standing problem in image processing and computer vision. One common solution is to apply an edge-preserving smoothing filter to the image such as anisotropic diffusion [Perona and Malik 1990] or bilateral filtering [Tomasi and Manduchi 1998]. The bilateral filter is a fast, non-iterative technique, and has been applied to a variety of problems beyond image denoising, including tone-mapping [Durand and Dorsey 2002; Choudhury and Tumblin 2003], separating illumination from texture [Oh et al. 2001] and mesh smoothing [Fleishman et al. 2003; Jones et al. 2003].

Our ambient image denoising technique also builds on the bilateral filter. We begin with a summary of Tomasi and Manduchi's basic bilateral filter and then show how to extend their approach to also consider a flash image when denoising an ambient image.

Bilateral filter. The bilateral filter is designed to average together pixels that are spatially near one another and have similar intensity values. It combines a classic low-pass filter with an edge-stopping function that attenuates the filter kernel weights when the intensity difference between pixels is large. In the notation of Durand and Dorsey [2002], the bilateral filter computes the value of pixel *p* for ambient image *A* as:

$$A_p^{Base} = \frac{1}{k(p)} \sum_{p' \in \Omega} g_d(p' - p)\, g_r(A_p - A_{p'})\, A_{p'}, \quad (2)$$

where $k(p)$ is a normalization term:

$$k(p) = \sum_{p' \in \Omega} g_d(p' - p)\, g_r(A_p - A_{p'}). \quad (3)$$

The function g_d sets the weight in the spatial domain based on the distance between the pixels, while the edge-stopping function g_r sets the weight on the range based on intensity differences. Typically, both functions are Gaussians with widths controlled by the standard deviation parameters σ_d and σ_r respectively.

We apply the bilateral filter to each RGB color channel separately with the same standard deviation parameters for all three channels. The challenge is to set σ_d and σ_r so that the noise is averaged away but detail is preserved. In practice, for 6 megapixel images, we set σ_d to cover a pixel neighborhood of between 24 and 48 pixels, and then experimentally adjust σ_r so that it is just above the threshold necessary to smooth the noise. For images with pixel values normalized to [0.0, 1.0] we usually set σ_r to lie between 0.05 and 0.1, or 5 to 10% of the total range. However, as shown in Figure 4(b), even after carefully adjusting the parameters, the basic bilateral filter tends to either over-blur (lose detail) or under-blur (fail to denoise) the image in some regions.

Joint bilateral filter. We observed in Section 2 that the flash image contains a much better estimate of the true high-frequency information than the ambient image. Based on this observation, we modify the basic bilateral filter to compute the edge-stopping function g_r using the flash image F instead of A. We call this technique the *joint bilateral filter*[2]:

$$A_p^{NR} = \frac{1}{k(p)} \sum_{p' \in \Omega} g_d(p' - p)\, g_r(F_p - F_{p'})\, A_{p'}, \quad (4)$$

where $k(p)$ is modified similarly. Here A^{NR} is the noise-reduced version of A. We set σ_d just as we did for the basic bilateral filter. Under the assumption that F has little noise, we can set σ_r to be very small and still ensure that the edge-stopping function $g_r(F_p - F_{p'})$ will choose the proper weights for nearby pixels and therefore will not over-blur or under-blur the ambient image. In practice, we have found that σ_r can be set to 0.1% of the total range of color values. Unlike basic bilateral filtering, we fix σ_r for all images.

The joint bilateral filter relies on the flash image as an estimator of the ambient image. Therefore it can fail in flash shadows and specularities because they only appear in the flash image. At the edges of such regions, the joint bilateral filter may under-blur the ambient image since it will down-weight pixels where the filter straddles these edges. Similarly, inside these regions, it may over-blur the ambient image.

We solve this problem by first detecting flash shadows and specular regions as described in Section 4.3 and then falling back to basic bilateral filtering within these regions. Given the mask *M*

[2] Eisemann and Durand [2004] call this the cross bilateral filter.

produced by our detection algorithm, our improved denoising algorithm becomes:

$$A^{NR'} = (1 - M)\, A^{NR} + M A^{Base}. \quad (5)$$

Results & Discussion. The results of denoising with the joint bilateral filter are shown in Figure 4(c). The difference image with the basic bilateral filter, Figure 4(d), reveals that the joint bilateral filter is better able to preserve detail while reducing noise.

One limitation of both bilateral and joint bilateral filtering is that they are nonlinear and therefore a straightforward implementation requires performing the convolution in the spatial domain. This can be very slow for large σ_d.

Recently Durand and Dorsey [2002] used Fourier techniques to greatly accelerate the bilateral filter. We believe their technique is also applicable to the joint bilateral filter and should significantly speed up our denoising algorithm.

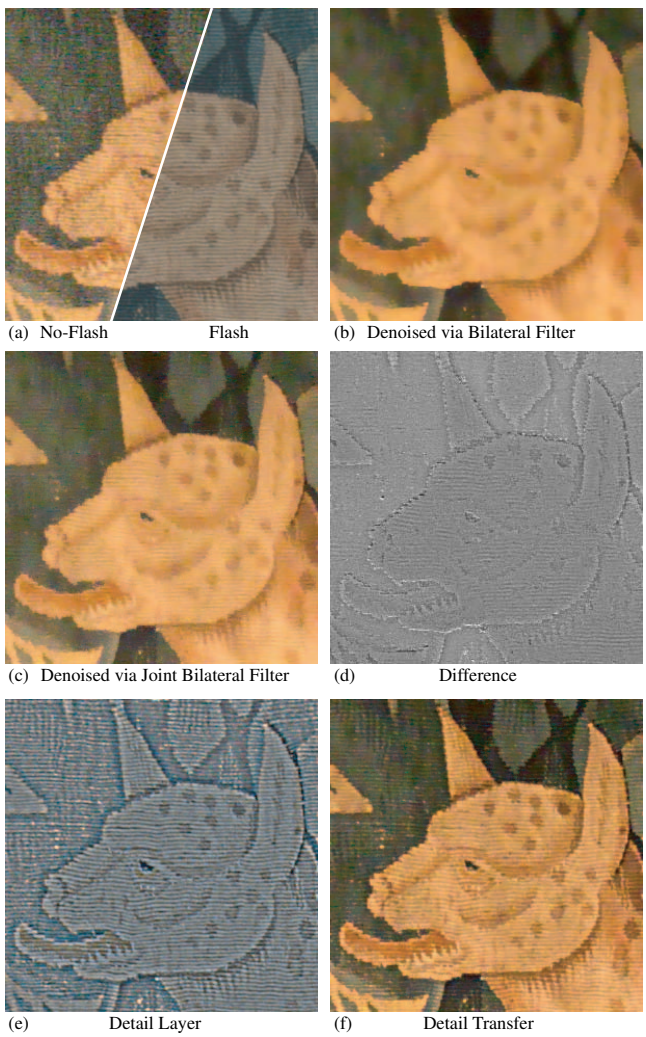


Figure 4: (a) A close-up of a flash/no-flash image pair of a Belgian tapestry. The no-flash image is especially noisy in the darker regions and does not show the threads as well as the flash image. (b) Basic bilateral filtering preserves strong edges, but blurs away most of the threads. (c) Joint bilateral filtering smoothes the noise while also retaining more thread detail than the basic bilateral filter. (d) The difference image between the basic and joint bilateral filtered images. (e) We further enhance the ambient image by transferring detail from the flash image. We first compute a detail layer from the flash image, and then (f) combine the detail layer with the image denoised via the joint bilateral filter to produce the detail-transferred image.

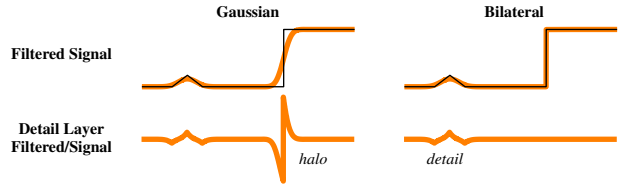


Figure 5: (left) A Gaussian low-pass filter blurs across all edges and will therefore create strong peaks and valleys in the detail image that cause halos. (right) The bilateral filter does not smooth across strong edges and thereby reduces halos, while still capturing detail.

4.2 Flash-To-Ambient Detail Transfer

While the joint bilateral filter can reduce noise, it cannot add detail that may be present in the flash image. Yet, as described in Section 2, the higher SNR of the flash image allows it to retain nuances that are overwhelmed by noise in the ambient image. Moreover, the flash typically provides strong directional lighting that can reveal additional surface detail that is not visible in more uniform ambient lighting. The flash may also illuminate detail in regions that are in shadow in the ambient image.

To transfer this detail we begin by computing a detail layer from the flash image as the following ratio:

$$F^{Detail} = \frac{F + \varepsilon}{F^{Base} + \varepsilon}, \quad (6)$$

where F^{Base} is computed using the basic bilateral filter on F. The ratio is computed on each RGB channel separately and is independent of the signal magnitude and surface reflectance. The ratio captures the local detail variation in F and is commonly called a quotient image [Shashua and Riklin-Raviv 2001] or ratio image [Liu et al. 2001] in computer vision. Figure 5 shows that the advantage of using the bilateral filter to compute F^{Base} rather than a classic low-pass Gaussian filter is that we reduce haloing.

At low signal values, the flash image contains noise that can generate spurious detail. We add ε to both the numerator and denominator of the ratio to reject these low signal values and thereby reduce such artifacts (and also avoid division by zero). In practice we use $\varepsilon = 0.02$ across all our results. To transfer the detail, we simply multiply the noise-reduced ambient image A^{NR} by the ratio F^{Detail}. Figure 4(e-f) shows examples of a detail layer and detail transfer.

Just as in joint bilateral filtering, our transfer algorithm produces a poor detail estimate in shadows and specular regions caused by the flash. Therefore, we again rely on the detection algorithm described in Section 4.3 to estimate a mask M identifying these regions and compute the final image as:

$$A^{Final} = (1 - M) A^{NR} F^{Detail} + M A^{Base}. \quad (7)$$

With this detail transfer approach, we can control the amount of detail transferred by choosing appropriate settings for the bilateral filter parameters σ_d and σ_r used to create F^{Base}. As we increase these filter widths, we generate increasingly smoother versions of F^{Base} and as a result capture more detail in F^{Detail}. However, with excessive smoothing, the bilateral filter essentially reduces to a Gaussian filter and leads to haloing artifacts in the final image.

Results & Discussion. Figures 1, 4(f), and 6–8 show several examples of applying detail transfer with denoising. Both the lamp (Figure 6) and pots (Figure 8) examples show how our detail transfer algorithm can add true detail from the flash image to the ambient image. The candlelit cave (Figure 1 and 7) is an extreme case for our algorithms because the ISO was originally set to 1600 and digitally boosted up to 6400 in a post-processing step. In this

Orig. (top) Detail Transfer (bottom) | Flash | No-Flash | Detail Transfer with Denoising

Figure 6: An old European lamp made of hay. The flash image captures detail, but is gray and flat. The no-flash image captures the warm illumination of the lamp, but is noisy and lacks the fine detail of the hay. With detail transfer and denoising we maintain the warm appearance, as well as the sharp detail.

No-Flash | Detail Transfer with Denoising | Long Exposure Reference

Figure 7: We captured a long exposure image of the wine cave scene (3.2 seconds at ISO 100) for comparison with our detail transfer with denoising result. We also computed average mean-square error across the 16 bit R, G, B color channels between the no-flash image and the reference (1485.5 MSE) and between our result and the reference (1109.8 MSE). However, it is well known that MSE is not a good measure of perceptual image differences. Visual comparison shows that although our result does not achieve the fidelity of the reference image, it is substantially less noisy than the original no-flash image.

case, the extreme levels of noise forced us to use relatively wide Gaussians for both the domain and range kernels in the joint bilateral filter. Thus, when transferring back the true detail from the flash image, we also used relatively wide Gaussians in computing the detail layer. As a result, it is possible to see small halos around the edges of the bottles. Nevertheless, our approach is able to smooth away the noise while preserving detail like the gentle wrinkles on the sofa and the glazing on the bottles. Figure 7 shows a comparison between a long exposure reference image of the wine cave and our detail transfer with denoising result.

In most cases, our detail transfer algorithm improves the appearance of the ambient image. However, it is important to note that the flash image may contain detail that looks unnatural when transferred to the ambient image. For example, if the light from the flash strikes a surfaces at a shallow angle, the flash image may pick up surface texture (i.e. wood grain, stucco, etc.) as detail. If this texture is not visible in the original ambient image, it may look odd. Similarly if the flash image washes out detail, the ambient image may be over-blurred. Our approach allows the user to control how much detail is transferred over the entire image. Automatically adjusting the amount of local detail transferred is an area for future work.

4.3 Detecting Flash Shadows and Specularities

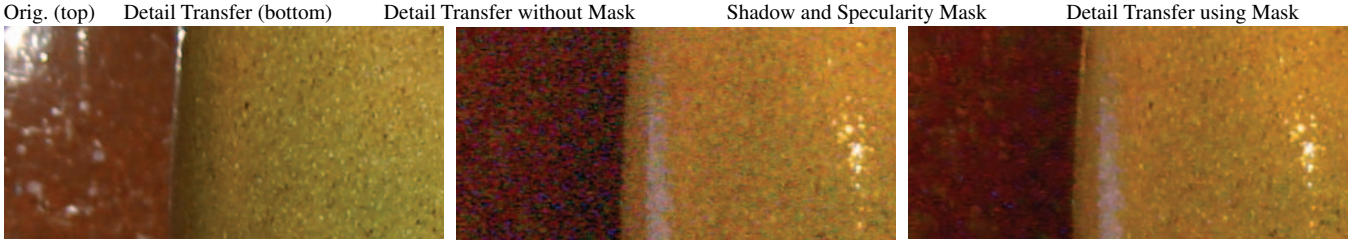
Light from the flash can introduce shadows and specularities into the flash image. Within flash shadows, the image may be as dim as the ambient image and therefore suffer from noise. Similarly, within specular reflections, the flash image may be saturated and lose detail. Moreover, the boundaries of both these regions may form high-frequency edges that do not exist in the ambient image. To avoid using information from the flash image in these regions, we first detect the flash shadows and specularities.

Flash Shadows. Since a point in a flash shadow is not illuminated by the flash, it should appear exactly as it appears in the ambient image. Ideally, we could linearize A and F as described in Section 3 and then detect pixels where the luminance of the difference image $F^{Lin} - A^{Lin}$ is zero. In practice, this approach is confounded by four issues: 1) surfaces that do not reflect any light (i.e. with zero albedo) are detected as shadows; 2) distant surfaces not reached by the flash are detected as shadows; 3) noise causes non-zero values within shadows; and 4) inter-reflection of light from the flash causes non-zero values within the shadow.

The first two issues do not cause a problem since the results are the same in both the ambient and flash images and thus whichever image is chosen will give the same result. To deal with noise and inter-reflection, we add a threshold when computing the shadow mask by looking for pixels in which the difference between the linearized flash and ambient images is small:

$$M^{Shad} = \begin{cases} 1 \text{ when } F^{Lin} - A^{Lin} \le \tau_{Shad} \\ 0 \text{ else} \end{cases}. \quad (8)$$

Orig. (top) Detail Transfer (bottom) | Detail Transfer without Mask | Shadow and Specularity Mask | Detail Transfer using Mask



Flash | No-Flash | Detail Transfer with Denoising

Figure 8: (top row) The flash image does not contain true detail information in shadows and specular regions. When we naively apply our denoising and detail transfer algorithms, these regions generate artifacts as indicated by the white arrows. To prevent these artifacts, we revert to basic bilateral filtering within these regions. (bottom row). The dark brown pot on the left is extremely noisy in the no-flash image. The green pot on the right is also noisy, but as shown in the flash image it exhibits true texture detail. Our detail transfer technique smoothes the noise while maintaining the texture. Also note that the flash shadow/specularity detection algorithm properly masks out the large specular highlight on the brown pot and does not transfer that detail to the final image.

We have developed a program that lets users interactively adjust the threshold value τ_{Shad} and visually verify that all the flash shadow regions are properly captured.

Noise can contaminate the shadow mask with small speckles, holes and ragged edges. We clean up the shadow mask using image morphological operations to erode the speckles and fill the holes. To produce a conservative estimate that fully covers the shadow region, we then dilate the mask.

Flash Specularities. We detect specular regions caused by the flash using a simple physically motivated heuristic. Specular regions should be bright in F^{Lin} and should therefore saturate the image sensor. Hence, we look for luminance values in the flash image that are greater than 95% of the range of sensor output values. We clean, fill holes, and dilate the specular mask just as we did for the shadow mask.

Final Merge. We form our final mask M by taking the union of the shadow and specular masks. We then blur the mask to feather its edges and prevent visible seams when the mask is used to combine regions from different images.

Results & Discussion. The results in Figures 1 and 6–8 were generated using this flash artifact detection approach. Figure 8 (top row) illustrates how the mask corrects flash shadow artifacts in the detail transfer algorithm. In Figure 1 we show a failure case of our algorithm. It does not capture the striped specular highlight on the center bottle and therefore this highlight is transferred as detail from the flash image to our final result.

Although both our shadow and specular detection techniques are based on simple heuristics, we have found that they produce good masks for a variety of examples. More sophisticated techniques developed for shadow and specular detection in single images or stereo pairs [Lee and Bajcsy 1992; Funka-Lea and Bajcsy 1995; Swaminathan et al. 2002] may provide better results and could be adapted for the case of flash/no-flash pairs.

5 White Balancing

Although preserving the original ambient illumination is often desirable, sometimes we may want to see how the scene would appear under a more "white" illuminant. This process is called white-balancing, and has been the subject of much study [Adams et al. 1998].

When only a single ambient image is acquired, the ambient illumination must be estimated based on heuristics or user input. Digital cameras usually provide several white-balance modes for different environments such as sunny outdoors and fluorescent lighting. Most often, pictures are taken with an "auto" mode, wherein the camera analyzes the image and computes an image-wide average to infer ambient color. This is, of course, only a heuristic, and some researchers have considered semantic analysis to determine color cast [Schroeder and Moser 2001].

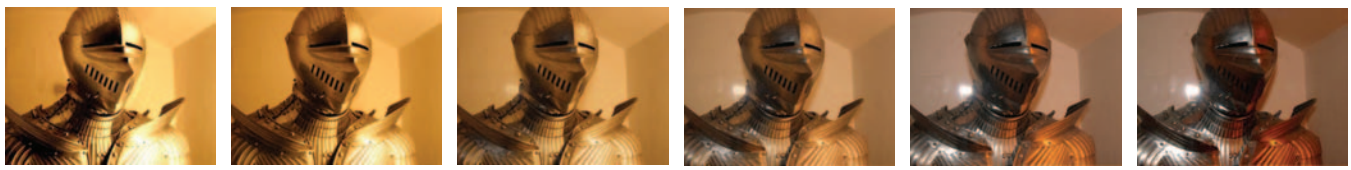
A flash/no-flash image pair enables a better approach to white balancing. Our work is heavily inspired by that of DiCarlo et al. [2001], who were the first to consider using flash/no-flash pairs for illumination estimation. They infer ambient illumination by performing a discrete search over a set of 103 illuminants to find the one that most closely matches the observed image pair. We simplify this approach by formulating it as a continuous optimization problem that is not limited by this discrete set of illuminants. Thus, our approach requires less setup than theirs.

We can think of the flash as adding a point light source of known color to the scene. By setting the camera white-balance mode to "flash" (and assuming a calibrated camera), this flash color should appear as reference white in the acquired images.

The difference image $\Delta = F^{Lin} - A^{Lin}$ corresponds to the illumination due to the flash only, which is proportional to the surface albedo at each pixel p. Note that the albedo estimate Δ has unknown scale, because both the distance and orientation of the surface are unknown. Here we are assuming either that the surface is diffuse or that its specular color matches its diffuse color. As a

Original No-Flash | Estimated ambient illumination | White-Balanced

Figure 9: (left) The ambient image (after denoising and detail transfer) has an orange cast to it. The insets show the estimated ambient illumination colors $\mathcal{C}$ and the estimated overall scene ambience. (right) Our white-balancing algorithm shifts the colors and removes the orange coloring .



-0.5 | 0.0 (No-Flash) | 0.33 | 0.66 | 1.0 (Flash) | 1.5

Figure 10: An example of continuous flash adjustment. We can extrapolate beyond the original flash/no-flash pair.

counter-example, this is not true of plastics. Similarly, semi-transparent surfaces would give erroneous estimates of albedo.

Since the surface at pixel p has color A_p in the ambient image and the scaled albedo Δ_p, we can estimate the ambient illumination at the surface with the ratio:

$$\mathcal{C}_p = \frac{\Delta_p}{A_p}, \tag{9}$$

which is computed per color channel. Again, this estimated color $\mathcal{C}_p$ has an unknown scale, so we normalize it at each pixel p (see inset Figure 9). Our goal is to analyze $\mathcal{C}_p$ at all image pixels to infer the ambient illumination color c. To make this inference more robust, we discard pixels for which the estimate has low confidence. We can afford to do this since we only need to derive a single color from millions of pixels. Specifically, we ignore pixels for which either $\|A_p\| < \tau_1$ or the luminance of $\Delta_p < \tau_2$ in any channel, since these small values make the ratio less reliable. We set both τ_1 and τ_2 to about 2% of the range of color values.

Finally, we compute the ambient color estimate c for the scene as the mean of $\mathcal{C}_p$ for the non-discarded pixels. (An alternative is to select c as the principal component of $\mathcal{C}$, obtained as the eigenvector of $\mathcal{C}^T\mathcal{C}$ with the largest eigenvalue, and this gives a similar answer.)

Having inferred the scene ambient color c, we white-balance the image by scaling the color channels as:

$$A_p^{WB} = \frac{1}{c} A_p . \tag{10}$$

Again, the computation is performed per color channel.

Results & Discussion. Figure 9 shows an example of white balancing an ambient image. The white balancing significantly changes the overall hue of the image, setting the color of the wood table to a yellowish gray, as it would appear in white light.

In inferring ambient color c, one could also prune outliers and look for spatial relationships in the image $\mathcal{C}$. In addition, the scene may have multiple regions with different ambient colors, and these could be segmented and processed independently.

White-balancing is a challenging problem because the perception of "white" depends in part on the adaptation state of the viewer. Moreover, it is unclear when white-balance is desirable. However we believe that our estimation approach using the known information from the flash can be more accurate than techniques based on single-image heuristics.

6 Continuous Flash Adjustment

When taking a flash image, the intensity of the flash can sometimes be too bright, saturating a nearby object, or it can be too dim, leaving mid-distance objects under-exposed. With a flash and non-flash image pair, we can let the user adjust the flash intensity after the picture has been taken.

We have explored several ways of interpolating the ambient and flash images. The most effective scheme is to convert the original flash/no-flash pair into YCbCr space and then linearly interpolate them using:

$$F^{Adjusted} = (1-\alpha)A + (\alpha)F . \tag{11}$$

To provide more user control, we allow extrapolation by letting the parameter α go outside the normal [0,1] range. However, we only extrapolate the *Y* channel, and restrict the *Cb* and *Cr* channel interpolations to their extrema in the two original images, to prevent excessive distortion of the hue. An example is shown in Figure 10.

7 Red-Eye Correction

Red-eye is a common problem in flash photography and is due to light reflected by a well vascularized retina. Fully automated red-eye removal techniques usually assume a single image as input and rely on a variety of heuristic and machine-learning techniques to localize the red eyes [Gaubatz and Ulichney 2002; Patti et al. 1998]. Once the pupil mask has been detected, these techniques darken the pixels within the mask to make the images appear more natural.

We have developed a red-eye removal algorithm that considers the change in pupil color between the ambient image (where it is usually very dark) and the flash image (where it may be red). We convert the image pair into YCbCr space to decorrelate luminance

from chrominance and compute a relative redness measure as follows:

$$R = F_{Cr} - A_{Cr} \,. \tag{12}$$

We then initially segment the image into regions where:

$$R > \tau_{Eye} \,. \tag{13}$$

We typically set τ_{Eye} to 0.05 so that the resulting segmentation defines regions where the flash image is redder than the ambient image and therefore may form potential red eyes. The segmented regions also tend to include a few locations that are highly saturated in the Cr channel of the flash image but are relatively dark in the Y channel of the ambient image. Thus, if μ_R and σ_R denote the mean and standard deviation of the redness R, we look for seed pixels where:

$$R > \max[0.6, \mu_R + 3\sigma_R] \text{ and } A_Y < \tau_{Dark} \,. \tag{14}$$

We usually set $\tau_{Dark} = 0.6$. If no such seed pixels exist, we assume the image does not contain red-eye. Otherwise, we use these seed pixels to look up the corresponding regions in the segmentation and then apply geometric constraints to ensure that the regions are roughly the same size and elliptical. In particular, we compute the area of each region and discard large outliers. We then check that the eccentricity of the region is greater than 0.75. These regions then form a red-eye pupil mask. Finally to correct these red-eye regions we use the technique of Patti et al.[1998]. We first remove the highlights or "glints" in the pupil mask using our flash specularity detection algorithm. We then set the color of each pixel in the mask to the gray value equivalent to 80% of its luminance value. This approach properly darkens the pupil while maintaining the specular highlight which is important for maintaining realism in the corrected output.

Results & Discussion. Figure 11 illustrates our red-eye correction algorithm with two examples. The second example shows that our algorithm performs well even when the red-eye is subtle. In both examples our algorithm is able to distinguish the pupils from the reddish skin. Moreover, the specular highlight is preserved and the eye shows no unnatural discoloration. Both of these examples were automatically aligned using the approach of Szeliski and Shum [1997]. Since color noise could invalidate our chromaticity comparison, we assume a relatively noise free ambient image, like the ones generated by our denoising algorithm.

8 Future Work and Conclusions

While we have developed a variety of applications for flash/no-flash image pairs, we believe there remain many directions for future work.

In some cases, the look of the flash image may be preferable to the ambient image. However, the flash shadows and specularities may be detracting. While we have developed algorithms for detecting these regions, we would like to investigate techniques for removing them from the flash image.

Flash shadows often appear at depth discontinuities between surfaces in the scene. Using multiple flash photographs it may be possible to segment foreground from background. Raskar et al. [2003] have recently explored this type of approach to generate non-photorealistic renderings.

Motion blur is a common problem for long-exposure images. It may be possible to extend our detail transfer technique to de-blur such images. Recent work by Jia et al. [2004] is beginning to explore this idea.

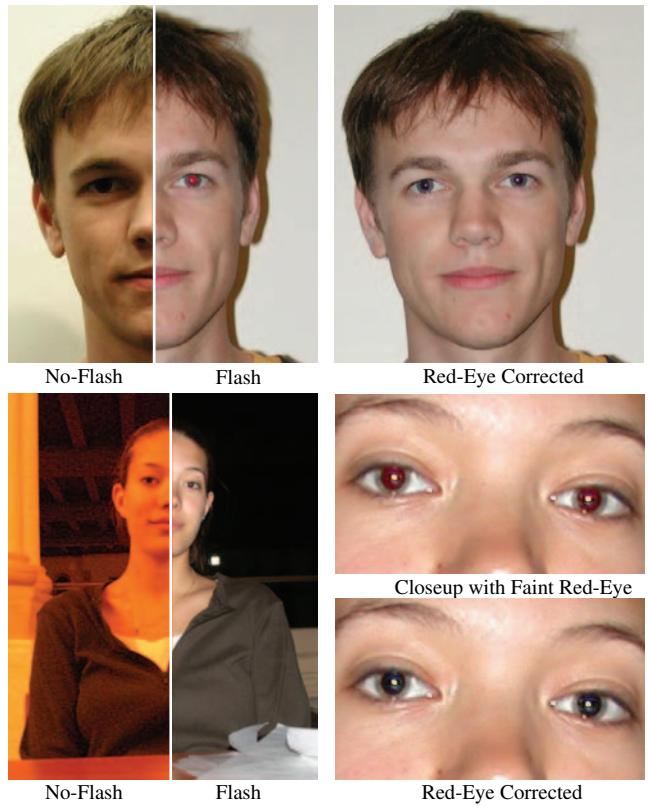


Figure 11: Examples of red-eye correction using our approach. Although the red eye is subtle in the second example, our algorithm is still able to correct the problem. We used a Nikon CoolPix 995 to acquire these images.

While our approach is designed for consumer-grade cameras, we have not yet considered the joint optimization of our algorithms and the camera hardware design. For example, different illuminants or illuminant locations may allow the photographer to gather more information about the scene.

An exciting possibility is to use an infrared flash. While infrared illumination yields incomplete color information, it does provide high-frequency detail, and does so in a less intrusive way than a visible flash.

We have demonstrated a set of applications that combine the strengths of flash and no-flash photographs to synthesize new images that are of better quality than either of the originals. The acquisition procedure is straightforward. We therefore believe that flash/no-flash image pairs can contribute to the palette of image enhancement options available to digital photographers. We hope that these techniques will be even more useful as cameras start to capture multiple images every time a photographer takes a picture.

Acknowledgements

We wish to thank Marc Levoy for mentoring the early stages of this work and consistently setting an example for scientific rigor. Steve Marschner gave us many pointers on the fine details of digital photography. We thank Chris Bregler and Stan Birchfield for their encouragement and Rico Malvar for his advice. Mike Braun collaborated on an early version of these ideas. Georg would like to thank his parents for maintaining Castle Pyrmont, which provided a beautiful setting for many of the images in the paper. Finally, we thank the anonymous reviewers for their exceptionally detailed feedback and valuable suggestions.

References

ADAMS, J., PARULSKI, K. AND SPAULDING, K., 1998. Color processing in digital cameras. *IEEE Micro,* 18(6), pp. 20-30.

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D. H., AND COHEN, M., 2003. Interactive Digital Photomontage. *ACM Transaction on Graphics,* 23(3), in this volume.

AKERS, D., LOSASSO, F., KLINGNER, J., AGRAWALA, M., RICK, J., AND HANRAHAN, P., 2003. Conveying shape and features with image-based relighting. *IEEE Visualization 2003,* pp. 349-354.

CHOUDHURY, P., AND TUMBLIN, J., 2003. The trilateral filter for high contrast images and meshes. In *Eurographics Rendering Symposium,* pp. 186-196.

DEBEVEC, P. E., AND MALIK, J., 1997. Recovering high dynamic range radiance maps from photographs. *ACM SIGGRAPH 97*, pp. 369-378.

DEBEVEC, P., HAWKINS, T., TCHOU, C., DUIKER, H., SAROKIN, W. AND SAGAR, M., 2000. Acquiring the reflectance field of the human face. *ACM SIGGRAPH 2000,* pp. 145-156.

DICARLO, J. M., XIAO, F., AND WANDELL, B. A., 2001. Illuminating illumination. *Ninth Color Imaging Conference,* pp. 27-34.

DURAND, F., AND DORSEY, J., 2002. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics*, 21(3), pp. 257-266.

EISEMANN, E., AND DURAND, F., 2004. Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics*, 23(3), in this volume.

FLEISHMAN, S., DRORI, I. AND COHEN-OR, D., 2003. Bilateral mesh denoising. *ACM Transaction on Graphics,* 22(3), pp. 950-953.

FUNKA-LEA, G., AND BAJCSY, R., 1995. Active color and geometry for the active, visual recognition of shadows. *International Conference on Computer Vision,* pp. 203-209.

GAUBATZ, M., AND ULICHNEY, R., 2002. Automatic red-eye detection and correction. *IEEE International Conference on Image Processing,* pp. 804-807.

HAEBERLI, P., 1992. Synthetic lighting for photography. *Grafica Obscura*, http://www.sgi.com/grafica/synth.

JIA, J., SUN, J., TANG, C.-K., AND SHUM, H., 2004. Bayesian correction of image intensity with spatial consideration. *ECCV 2004*, LNCS 3023, pp. 342-354.

JONES, T.R., DURAND, F. AND DESBRUN, M., 2003. Non-iterative feature preserving mesh smoothing. *ACM Transactions on Graphics,* 22(3), pp. 943-949.

KANG, S. B., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R., 2003. High dynamic range video. *ACM Transactions on Graphics*, 22(3), pp. 319-325.

KODAK, 2001. CCD Image Sensor Noise Sources. *Application Note MPT/PS-0233*.

LEE, S. W., AND BAJCSY, R., 1992. Detection of specularity using color and multiple views. *European Conference on Computer Vision,* pp. 99-114.

LIU, Z., SHAN., Y., AND ZHANG, Z., 2001. Expressive expression mapping with ratio images. *ACM SIGGRAPH 2001,* pp. 271-276.

MASSELUS, V., DUTRE, P., ANRYS, F., 2002. The free-form light stage. In *Eurographics Rendering Symposium,* pp. 247-256.

OH, B.M., CHEN, M., DORSEY, J. AND DURAND, F., 2001. Image-based modeling and photo editing. *ACM SIGGRAPH 2001,* pp. 433-442.

PATTI, A., KONSTANTINIDES, K., TRETTER, D. AND LIN, Q., 1998. Automatic digital redeye reduction. *IEEE International Conference on Image Processing,* pp. 55-59.

PERONA, P., AND MALIK, J., 1990. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7), pp. 629-639.

RASKAR, R., YU, J. AND ILIE, A., 2003. A non-photorealistic camera: Detecting silhouettes with multi-flash. *ACM SIGGRAPH 2003 Technical Sketch*.

SCHROEDER, M., AND MOSER, S., 2001. Automatic color correction based on generic content-based image analysis. *Ninth Color Imaging Conference,* pp. 41-45.

SHASHUA, A., AND RIKLIN-RAVIV, T., 2001. The quotient image: class based re-rendering and recognition with varying illuminations. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 23(2), pp. 129-139.

SWAMINATHAN, R., KANG, S. B., SZELISKI, R., CRIMINISI, A. AND NAYAR, S. K., 2002. On the motion and appearance of specularities in image sequences. *European Conference on Computer Vision,* pp. I:508-523

SZELISKI, R., AND SHUM, H., 1997. Creating full view panoramic image mosaics and environment maps. *ACM SIGGRAPH 97,* pp. 251-258.

TOMASI, C., AND MANDUCHI, R., 1998. Bilateral filtering for gray and color images. *IEEE International Conference on Computer Vision,* pp. 839-846.

Video Enhancement Using Per-Pixel Virtual Exposures

Eric P. Bennett Leonard McMillan
The University of North Carolina at Chapel Hill

Abstract

We enhance underexposed, low dynamic range videos by adaptively and independently varying the exposure at each photoreceptor in a post-process. This virtual exposure is a dynamic function of both the spatial neighborhood and temporal history at each pixel. Temporal integration enables us to expand the image's dynamic range while simultaneously reducing noise. Our non-linear exposure variation and denoising filters smoothly transition from temporal to spatial for moving scene elements. Our virtual exposure framework also supports temporally coherent per frame tone mapping. Our system outputs restored video sequences with significantly reduced noise, increased exposure time of dark pixels, intact motion, and improved details.

CR Categories: I.4.3: Image Processing and Computer Vision – enhancement, filtering. I.3.3: Computer Graphics – picture/image generation, display algorithms.

Keywords: Digital video, Noise reduction, Low dynamic range, Exposure, Bilateral filter, Restoration, Tone mapping.

1 Introduction

High dynamic range (HDR) imaging, processing, and display have recently received considerable attention. An implicit assumption of most HDR systems is a sizeable signal-to-noise ratio achieved via long exposures in low-light areas. Generally, multiple low dynamic range (LDR) images with different exposure settings are combined to generate a single HDR image, which implies a static scene. However, people have long been accidentally capturing poorly exposed video with camcorders and motion-picture cameras (countless home videos of school plays and dance recitals lay testament to this phenomenon). We address the problem of enhancing such videos. Aside from the noise characteristics of dark videos, there is a surprising commonality between HDR and LDR imaging. In this paper, we develop methods for enhancing LDR video to simulate the characteristics of individually tone-mapped HDR video frames, for applications in filmmaking, surveillance, forensics, and high-speed imaging.

Humans simultaneously perceive regions with high luminous intensities alongside intensities several orders of magnitude lower by spatially adapting the visual field's local sensitivity. Modern digital still cameras, however, rely on a single exposure time across the entire frame and photosites with uniform sensitivities necessitating that multiple images be taken at varying exposures to capture the full nuance of HDR scenes. This is problematic for dynamic scenes, where it is seldom possible to capture multiple exposures. Furthermore, HDR construction assumes an abundance of light and/or exposure intervals long enough to cancel the random noise fluctuations characteristic of image sensors. Once acquired, the problem becomes the accurate depiction of HDR results on LDR displays through tone mapping.

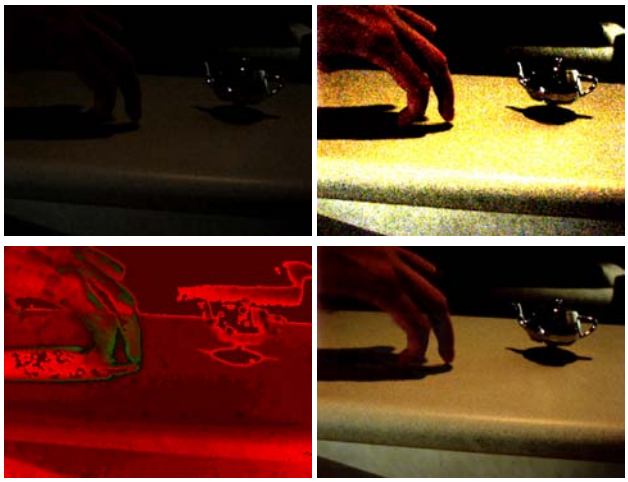


Figure 1: A frame from a video processed using virtual exposures. Upper Left: original frame; Upper Right: histogram stretched version; Bottom Left: red = number of temporal pixels integrated, green = number of spatial pixels integrated; Bottom Right: our result after filtering and tone mapping.

Alternatively, HDR images can be assembled by additively combining multiple uniformly exposed digital images [Liu et al 03] [Jostschulte et al 98]. This approach is particularly compatible with digital video, but it has been largely overlooked since it requires processing O(N) source images compared to the O(log(N)) of variable exposure methods. Nevertheless, the combination of multiple uniformly exposed images affords certain advantages, including noise reduction. Moreover, if one varies the number of uniformly exposed images combined on a pixel-by-pixel basis, it becomes possible to adjust each pixel's exposure independently, allowing for direct tone mapping without explicit construction of an intermediate HDR image. Local exposure control also provides a tool for handling dynamic scenes. This approach constitutes our Virtual Exposure Camera (VEC) conceptual model.

Our model of processing brings out hidden details that are barely noticeable in video frames due to underexposure and noise. It also synthesizes perceptually plausible and temporally consistent renditions of each video frame. Our process begins by estimating each pixel's exposure setting based on a spatially uniform tone mapping of each frame. It then attempts to recreate a corresponding gain ratio at each pixel by combining temporal samples of static scene elements and spatial samples of dynamic elements. This effectively denoises and tone maps the video.

Our virtual exposure method significantly enhances low dynamic range and noisy videos, making previously unwatchable material acceptable. The quantity of noise present directly affects the quality of the result, but so long as the noise is zero-mean, our method brings out details that are barely visible in the original.

The primary contributions of this work are:

- A virtual exposure camera model for enhancing LDR videos
- The Adaptive Spatio-Temporal Accumulation (ASTA) filter for reducing noise in LDR videos
- A tone mapping approach to enhance LDR videos

2 Related Work

Enhancing low-dynamic range images has much in common with HDR acquisition, processing, and display. HDR representations have long been recognized as essential for accurately modeling light transport [Ward 91]. More recently, Debevec and Malik [97] developed accurate methods for assembling HDR images from a series of still photographs with increasingly long exposure times.

The problem of mapping an HDR image for display on devices with limited dynamic range was formalized by Tumblin and Rushmier [93], and has led to a variety of spatially uniform [Drago et al 03] and spatially varying [Tumblin and Turk 99] [Durand and Dorsey 02] [Fattal et al 02] tone mapping approaches. A variety of methods have been proposed to tone map HDR images so that the maximum amount of information is visible on a monitor. Retinex theory, such as in the multiscale Retinex [Jobson et al 97], suggests that a Gaussian-like kernel can be convolved at each point in the image and subtracted from the original image in log space, providing for a more "viewable" version of a still image. The advantage of the Retinex approach is that it is non-iterative, but it can generate unwanted edge blurring artifacts. Durand and Dorsey [02] built a similar system, but used an edge preserving bilateral filter to maintain sharp edges. Pattanaik et al [00] presented an approach that mimics the time dependent local adaptation of the human visual system. They also discussed temporal coherence to avoid introducing frame-by-frame tone mapping "flicker". In gradient domain HDR compression [Fattal et al 02], the gradient of an image is attenuated and then reintegrated. They also described a modification for improving images that already use a display's full dynamic range. Raskar et al [04] also used gradient domain methods, but to fuse day and night images together— adding daytime context to nighttime footage.

The idea of using multiple temporally adjacent frames to enhance knowledge about a pixel's true or desired value was considered in Cohen et al [03]. Multiple images were registered and then each pixel of the output image was computed as a function of its temporal neighbors. HDR compression using this algorithm was also described. Sand and Teller [04] discussed a video matching method for aligning slightly different video sources. Specifically, it contains a robust system for frame-to-frame alignment. We handle moving cameras by warping spatio-temporal volumes as described by Bennett and McMillan [03].

There is a long history of noise filtering methods throughout the signal processing literature. We are most interested in edge-preserving filters from the anisotropic diffusion and bilateral filter families. Anisotropic diffusion of images [Perona and Malik 90] provides an iterative filtering method that adapts to the image's gradient. Bilateral filtering [Tomasi and Manduchi 98] provides a single-step noise removal process that shares many visual and mathematical qualities with anisotropic diffusion [Barash 02]. However, both of these methods are designed for single images and not for videos. The Trilateral filter [Choudhury and Tumblin 03] builds on the bilateral filter model by biasing its kernel away from edges and dynamically choosing the kernel's size in an attempt to model signals as piecewise linear rather than piecewise constant functions. Other modifications have been proposed to improve the standard bilateral filter's ability to handle noise [Boomgaard and Weijer 02] [Francis and Jager 03]. We combine the attributes of median filters with the bilateral filter. A "bilateral median" filter was described by Francis and Jager [03], but it used a weighted median for summation purposes, unlike ours that uses it to establish a dissimilarity value. Spatio-Temporal Anisotropic Diffusion [Lee et al 98] used a three dimensional kernel to remove video noise, treating temporal and spatial dimensions similarly. Instead, we adapt from temporal to spatial filtering.

Other video filtering approaches have appeared that use temporal filtering. Dubois and Sabri [84] performed nonlinear temporal noise filtering assisted by displacement estimation. Each pixel is combined temporally using a recursive low-pass temporal filter weighted by the reliability of the displacement estimate. This method requires well-exposed, easy-to-track video to correctly filter. Our method adapts from temporal to spatial filtering to be robust to tracking inaccuracies. Jostschulte et al [98] presented a spatio-temporal shot noise filter that first spatially and then temporally filters video while preserving edges that match a template set. A motion-sensing algorithm is used to vary the amount of temporal filtering. We prefer to only use temporal filtering when possible and adapt the mix of temporal and spatial filtering based on a tone-mapping objective and local motion characteristics.

Recently, Eisemann and Durand [04] and Petschnigg et al [04] have developed methods to remove noise and improve the dynamic range of underexposed images by incorporating features derived from properly exposed "flash images". The extent of noise removal depends on how well exposed a given region is in the flash image. Furthermore, the underlying luminance model used in the processing is not HDR, either explicitly (as in previous tone-mapping systems) or implicitly (as in our case). It is also unclear how to extend these methods to video sequences. The goal of our virtual exposure approach is similar to these methods, but we incorporate temporal information instead of flash image features to improve the exposure. Thus, the illuminations of our enhancements are consistent with the original source.

Researchers have also constructed actual high dynamic range video capture systems. Kang et al [03] built a system based on a camera that could sequence through different exposure settings. Once the images were registered using optical flow, it was possible to combine exposures to improve the dynamic range. The small number of frames combined suggests that a high signal-to-noise ratio (SNR) was assumed, and therefore, it would only be useful for well-lit scenes. Nayar and Branzoi [03] presented a system whereby a computer controlled LCD panel was placed in front of the CCD. The per-pixel transparency was varied to modulate the exposure of image regions based on the previous frame's luminance. They also discussed a local and global tone mapping approach that addresses temporal coherence issues. Using LCDs implies attenuation of the incoming light, thus further complicating low-light imaging. Nayar and Branzoi [04] later suggested a second variant using a DLP micromirror array to modulate the exposure, via time-division multiplexing (like a camera shutter), throughout the image. In theory, such systems could provide continuous exposure control at each pixel compared to our discretized exposure settings. However, they require additional hardware and are strictly causal; whereas our virtual exposure approach allows the incorporation of future information into virtual exposure decisions, assuming a constant latency.

Acosta-Serafini et al [04] described an HDR camera that selectively resets a pixel based on a prediction of when it will saturate. The reset interval and the digitized pixel level combine to form a floating-point value. They primarily focused on high-speed, HDR sensing and do not specifically address low-light situations. Liu et al [03] combined high-speed samples to reduce noise and improve dynamic range. Their approach is similar, but much lower-level than ours. It depends on specific imaging device features such as high-speed non-destructive reads. It also relies mostly on linear filters, and uses only single pixel areas to detect motion. In contrast, our method uses bilateral filtering, considers a larger context for motion detection, and targets a tone-mapped objective. Bidermann et al [03] described an HDR high-speed CMOS imager platform with per-pixel ADCs and storage, which could use the Liu et al [03] algorithm and targets well-lit scenes.

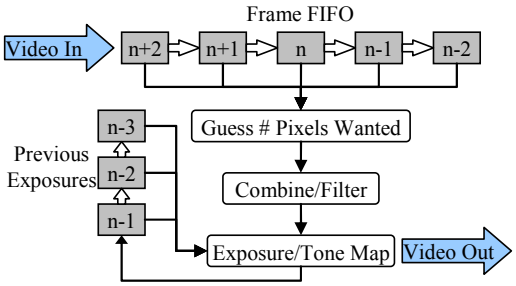


Figure 2: The VEC model for processing LDR video. Since no single frame contains sufficient information for noise reduction and tone mapping, processing is done with knowledge of recent frames and how tone mapping was applied. Rudimentary tone mapping is performed before filtering to guide the adaptive filter's settings.

3 The Virtual Exposure Camera Model

The Virtual Exposure Camera (VEC) is our conceptual model for analyzing and enhancing low dynamic range (LDR) video. Many common applications result in LDR videos. For instance, filming theatrical lighting is difficult because background scenery is seldom well exposed in comparison to the spotlights placed on the actors. LDR video also results from high speed imaging, where fast shutter speeds are desirable. Small aperture video, to increase depth-of-field, can also lead to LDR video. Poor lighting scenarios, such as is common in surveillance applications, also lead to underexposed videos.

3.1 LDR Video Noise Characteristics

The LDR videos we are interested in processing have a small signal-to-noise ratio and low precision. Our system also enhances videos with "peaky" histograms. Such scenes are composed of elements that span a significant dynamic range, but the combination of exposure settings and quantization leads to low precision renditions of all elements.

There are a variety of noise sources in CCD and CMOS sensors that confound imaging in low-light situations, such as readout, photon shot, dark current, and fixed pattern noise in addition to photon response non-uniformities [Reibel et al 03]. We assume that dark current noise and fixed pattern noise can be removed via subtraction of a reference dark image at the same temperature and exposure setting. Photon shot and readout noise are our primary problems, but we assume they are zero-mean, so if we can get multiple samples of the same pixel from temporally adjacent frames, we can average out the error. A significant problem for dealing with dark areas captured with CCDs is that the amplitude of sensor read noise is independent of exposure whereas photon shot noise varies linearly with exposure time. Read noise is more significant than shot noise at very dark pixels. Thus, for the darkest pixels, the SNR is comparatively smaller.

Computing the mean of n samples will improve the precision of the luminance readings by a $\sqrt{n}$ factor. These assumptions are not true for compressed video footage, where quantization is non-uniform across frequencies. We assume a linear camera response, which is true for raw CCD samples, but not for the hidden post-processing found in many camcorders.

3.2 Synthesizing Virtual Exposures

Our VEC model identifies poorly exposed regions of video and increases precision by simulating longer exposure times. This simulation involves temporal integration of the contributions of as many pixel values as would have been sampled over the interval of the longer exposure.

We process a spatio-temporal volume implemented as a FIFO queue (Fig. 2), where filtering occurs in the current frame but with knowledge of the frames that come before or after it (in a real-time, low latency system, the future might not be known). Therefore, the processing of a pixel can benefit from information in adjacent frames while also ensuring that tone mapping is temporally consistent. Pixels are indexed using *(x,y,t)* notation, with *t* being the frame number.

When integrating the contributions of multiple pixels together to simulate a longer exposure time, pixels that come before and after temporally can often be used. However, since some frames capture individual pixels with varying noise contributions, it is advantageous to exclude the noisiest pixels from the integration. Similarly, pixels that change due to object motion should not be included, to avoid blurring and "ghosting" artifacts.

Given LDR video, we apply a tone mapping algorithm targeted at improving poorly exposed areas and handling noise. Such a tone mapper is discussed in Section 5.

Prior to filtering, we estimate a gain factor for each pixel that scales its original luminance to achieve the pixel's final filtered output level. Because we cannot know this before filtering, we choose to estimate the filtered and tone mapped luminance by applying a spatially uniform tone mapping function $m(x,\not{y})$ to a Gaussian blurred version of the image. This gain factor is used by our non-linear filter, described in section 4, to determine how many pixels are additively combined thus establishing a per-pixel exposure time. We call this gain value λ and we use it to establish our adaptive filter's support.

4 The ASTA Filter

Our virtual exposure filter seeks out similar pixels to integrate. Two major factors affect how ASTA filters: how many pixels it wants to combine and if these pixels are in an area of the image with motion. ASTA adapts by transitioning between temporal-only and spatial-only bilateral-inspired filtering while choosing parameters based on local illumination.

4.1 The Spatial Bilateral Filter

ASTA is based on the edge-preserving bilateral filter [Tomasi and Manduchi 98]. The bilateral filter maintains edges by performing a Gaussian convolution but attenuates the contributions of pixels by how different their intensities are from the intensity at the center of the kernel. Although simple subtractive difference is often used to measure this difference of intensities, we generalize this notion to include non-photometric differences which we treat as dissimilarity values. A dissimilarity value is any relationship that satisfies the following properties: $D(x,x) = 0$ and $D(x,y) = D(y,x)$. A dissimilarity is metric if the triangle inequality holds: $D(x,y) + D(y,z) \geq D(x,z)$.
The spatial bilateral filter (for a pixel s), with a subtractive dissimilarity value $D(p,s)$, is shown in Equations 1 and 2:

$$B(s,\sigma_h,\sigma_i) = \frac{\sum_{p\in N_s} g\left(\|p-s\|,\sigma_h\right) g\left(D(p,s),\sigma_i\right) I_p}{\sum_{p\in N_s} g\left(\|p-s\|,\sigma_h\right) g\left(D(p,s),\sigma_i\right)} \quad \textbf{(1)}$$

$$g(x,\sigma) = e^{\frac{-x^2}{2\sigma^2}} / \left(\sigma\sqrt{2\pi}\right)$$

$$N_s = Kernel = \begin{bmatrix} p_x = [s_x - k, s_x + k] \\ p_y = [s_y - k, s_y + k] \end{bmatrix}$$

$$\text{where } D(p,s) \equiv I_p - I_s \quad \textbf{(2)}$$

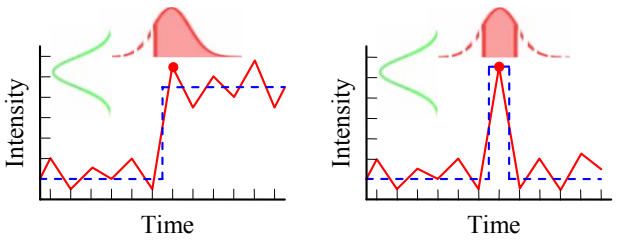


Figure 3: Left: The bilateral filter recovers the signal (blue) from the noisy input (red). Right: The bilateral filter is unable to attenuate the shot noise because no other pixels fall within the intensity dissimilarity Gaussian.

Three variables control the bilateral filter's operation. First, σ_h controls how quickly the spatial Gaussian falls off. The second, σ_i, controls the Gaussian dissimilarity weighting. It attenuates the contributions of neighboring pixels that are that are too different and is typically chosen based on an estimate of the signal's SNR. Finally, *k* determines kernel size.

The bilateral filter does a good job of smoothing out small imperfections while preserving edges, but it is incapable of removing shot noise from a signal (Fig. 3). When the filter kernel is centered on an outlier pixel, the intensity Gaussian will exclude all other values, leaving it unchanged, which accentuates it compared to the otherwise cleaned signal.

4.2 Bilateral Filtering in Time

In the case of a fixed camera, the best estimate of a pixel's true value is predicted from those pixels at the same location in different frames. In the absence of motion, a simple average of all pixels at each *(x,y)* coordinate through time gives an optimal answer, assuming zero-mean noise. However, averaging in the presence of motion creates "ghosting" artifacts. Our solution is to consider changes in a pixel's value due to motion as "temporal edges". A bilateral filter maintains edges while providing noise reduction in areas with small amplitude noise. Thus, we employ a temporal 1D-bilateral filter as a primary component in our noise reduction process.

A difficulty of applying a temporal bilateral filter is choosing an appropriate value for σ_i (the dissimilarity falloff) that simultaneously removes noise while preserving motion based entirely on differences of pixel luminance. If σ_i is too large, "ghosting" still results, and if σ_i is too small, noise will remain. Such a simple σ_i does often not exist for noisy video.

An alternative is to filter video with a volumetric bilateral kernel that operates in spatio-temporal volumes, much like how anisotropic diffusion was extended to 3D by Lee et al [98]. However, this symmetric approach does not take into account the difference in sampling density between space and time in a spatio-temporal volume.

4.3 Alternate Dissimilarity Values

As a solution to the typical bilateral filter's inability to remove shot noise, we introduce an alternate dissimilarity value *D(p,s)* in the bilateral filter. Instead of using the simple intensity difference, we substitute an arbitrary function that returns a value for each pair of pixels in a video or image that may or may not be solely intensity-based.

For example, the dissimilarity value could be the difference between *p* and some statistic of the local spatial neighborhood around *s*, making the filter more robust to shot noise. We use a median-centered bilateral filter that uses a small kernel median filter centered at *s* to improve quality in noisy image areas. The problem of choosing the intensity at the bilateral filter's center as the sole reference was discussed by Boomgaard and Weijer [02], but no suggestion of an alternative statistic was given. A wide variety of statistics could be applied to choose the *s* pixel's intensity, such as local minima, local maxima, or even other bilateral filters. Even measures not directly associated with luminance could be used.

4.4 Spatial Neighborhood Dissimilarity Value

We use a different dissimilarity value in our temporal bilateral filter. Specifically, the method is to compare the local spatial neighborhoods centered at the same pixel in different frames. Equation 3 shows our normalized Gaussian weighted dissimilarity for an $n \times n$ neighborhood and a temporal edge tolerance of σ_e.

$$D(p_{xyt}, s_{xyt}) = \frac{\sum_{x=sx-n}^{sx+n} \sum_{y=sy-n}^{sy+n} g\left(\left\|x - p_x, y - p_y\right\|, \sigma_e\right) \left|I_{x,y,pt} - I_{x,y,st}\right|}{\sum_{x=sx-n}^{sx+n} \sum_{y=sy-n}^{sy+n} g\left(\left\|x - p_x, y - p_y\right\|, \sigma_e\right)} \quad \textbf{(3)}$$

The difference between two pixels' intensities does not provide enough information to judge if they are significantly different. However, by comparing spatial neighborhoods, a judgment can be reached. Thus if only a small percentage of pixels change, we assume it to be noise and integrate into the filter. If many pixels change, we assume it to be a more significant event, and no blending occurs. For clarification, despite the fact we are comparing neighborhoods, it is only the pixels at the center of each neighborhood that will ultimately be blended together. The neighborhood size, often between 3 and 5, can be varied depending on noise characteristics, as can σ_e (usually between 2 and 6). Our dissimilarity value is inspired by correspondence measures frequently used in stereo imaging. We have used Sum of Absolute Differences (SAD) and Sum of Squared Differences (SSD). We implemented both versions and got similar results, although SSD occasionally created artificially sharp edges. Figure 4 illustrates our SAD version.

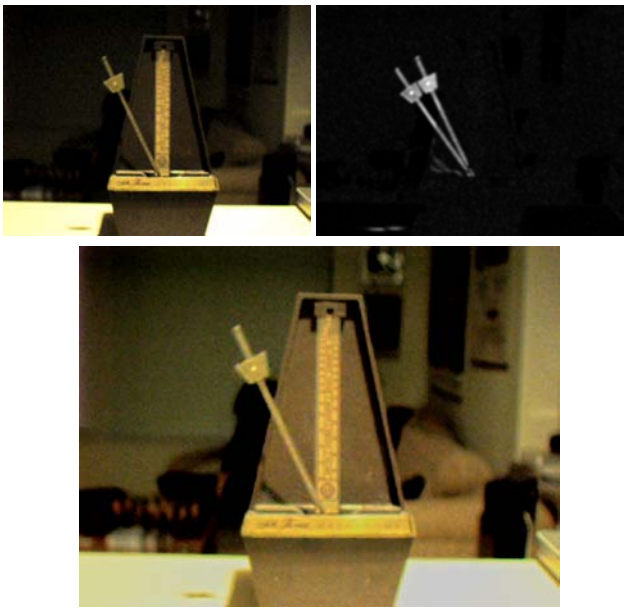


Figure 4: Illustration of our spatial neighborhood dissimilarity value used in temporal filtering. The original frame is shown in the upper left. Each *(x,y)* for a pair of nearby frames are shown in the upper right. Two metronome arms are seen because the dissimilarity value is based on absolute value. The bottom image is the same frame processed using ASTA and our tone mapper.

4.5 Implementing ASTA

The VEC model determines how many pixels should be combined to achieve our tone map brightness target. If only temporal bilateral filtering with the spatial neighborhood dissimilarity value is used, and it is in an area of high motion, only the center pixel of the kernel will make a sizable contribution to the result. In this case, it would not integrate enough pixels to achieve the desired gain factor. To overcome this problem we instead use an Adaptive Spatio-Temporal Accumulation filter (ASTA) that adapts to its surroundings to find enough pixels in the presence of motion. For a static pixel, it reduces to a temporal bilateral filter with the spatial neighborhood difference dissimilarity value. However, if it does not find enough similar pixels to achieve the desired exposure based on the size of the normalizing factor in the denominator of Equation 1, it transitions to a spatial-only median-centered bilateral filter, as shown in Figure 5. Like Yee et al [01], we also exploit the psychophysical phenomenon that in areas of motion, the human visual system's ability to perceive high frequencies is reduced. Thus, in areas with insufficient temporal information due to motion, we can transition to spatial filtering.

One way to conceptualize ASTA is as a voting scheme, where each vote is a measure of the support of the filter. Before ASTA is run on a pixel, we determine how many votes (pixels) are required (defined as λ, Section 3.2). The temporal bilateral filter gathers some votes, and if they are not sufficient, more votes are gathered from the spatial bilateral filter.

The number of votes desired is defined as $\lambda \times g(0,\sigma_h) \times g(0,\sigma_i)$. The factor $g(0,\sigma_h) \times g(0,\sigma_i)$ is our definition of a vote because it is the contribution to the denominator of the bilateral filter from a pixel that is an exact match in space and intensity ($D(x,y)=0$). The larger the dissimilarity value, the lower its contribution to the denominator is. Thus, by analyzing the denominator of a bilateral filter, we can determine if a sufficient number of votes were tallied. ASTA is thus formalized in Equation 4. The terms n and d represent the numerator and denominator of Equation 1, respectively.

$$\begin{aligned}
\frac{n_T}{d_T} &= temporalBilateral(x,y,t,\sigma_h,\sigma_i) \\
\frac{n_S}{d_S} &= spatialBilateral(x,y,t,\sigma_h',\sigma_i') \\
\omega &= \lambda \times g(0,\sigma_h) \times g(0,\sigma_i) \\
ASTA(x,y,t,\lambda,\sigma_i,\sigma_i') &= \begin{cases}
\dfrac{n_T}{d_T}, & d_T \geq \omega \\
\dfrac{n_T + n_S}{d_T + d_S}, & d_T < \omega \; AND \; d_T + d_S < \omega \\
\dfrac{n_t + n_s \dfrac{(w - d_t)}{d_s}}{w}, & d_T < \omega \; AND \; d_T + d_S \geq \omega
\end{cases}
\end{aligned} \tag{4}$$

ASTA changes its filtering settings based on the number of pixels it wants to combine. First, not every pixel could ever get a full vote, because even though it may have the same neighborhood it is attenuated by the distance Gaussian. Therefore, we choose the temporal filter kernel size and Gaussian σ_h dynamically such that if every comparison were a perfect match, $D_t \approx 2 \times \omega$. Similarly, if the vote count for the temporal bilateral comes up short, the spatial bilateral attempts to have the remaining number of votes fall within the area of one standard deviation of its distance Gaussian by dynamically choosing σ_h'. The remaining sigmas, σ_i for the temporal bilateral (and σ_e for its dissimilarity value) and σ_i' for the spatial bilateral, are held constant in each video's processing.

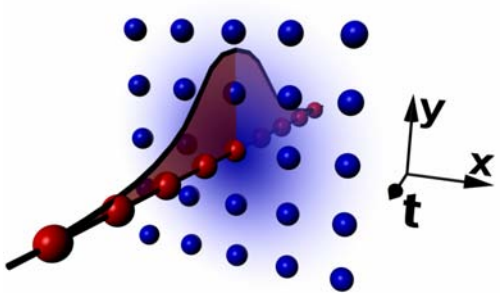


Figure 5: Illustration of the temporal-only and spatial-only nature of ASTA. The temporally filtered red pixels are preferred to be integrated into the filter, but if not enough are similar to the center of the kernel, the blue spatial pixels begin to be integrated.

Temporal bilateral filters are run on the image's luminance and mapped to each channel, but only spatial filtering is done on each color channel. Furthermore, spatial filtering is done in the log domain, whereas temporal filtering is not.

So far, we have assumed that the camera used to capture footage is stationary, assuring spatial correspondences for background pixels. For moving cameras, feature tracking is used. Sand and Teller [03] detail a system for finding accurate frame-to-frame correspondences which can identify temporal neighbors. In our system video registration and alignment takes place prior to noise reduction. We only consider "high-confidence" trackable points (as determined by OpenCV's *GoodFeaturesToTrack()*). We then select high-confidence optical-flow vectors (as determined using OpenCV's feature tracking) that correspond to the trackable points that occur on the dominant flow field (typically the background). Finally, we select the mean of this feature set as a translation for each frame. Our approach removes only the dominant motion, although more complex tracking methods could also be used. We used the spatio-temporal video editing system of [Bennett and McMillan 03] to do this. Once stabilized, video can be processed and then the stabilization can be removed. Any residual motions or misalignment are treated as moving objects by our ASTA filter.

5 LDR Tone Mapping

Our tone mapping approach considers that SNR varies with intensity. Thus, details in dark regions are less accurate than those in brighter regions. A tone mapper specialized for underexposed video should therefore associate a confidence level for details based on their luminous intensity. For instance, in the brightest areas of a video where the CCD received a reasonable exposure, the mix of details and large-scale features should be adjusted to achieve the tone mapping objectives. In darker areas the details should be attenuated to suppress noise.

Using the tone-mapping approach of Durand and Dorsey [02], it is possible to separate an image into details and large scale features. Subtracting the original log-image from a bilaterally filtered log-image provides an estimate of the image details. Durand and Dorsey then attenuate the large scale features by a uniform scale factor in the log domain to reduce the overall contrast of the HDR image, but leave the details untouched. This is not a problem for low-noise source images. In contrast, our LDR tone-mapping processes the details and large scale parts with different pipelines that attenuate details based on their estimated accuracy, as determined by local luminance, and it attenuates the large scale features to achieve the desired contrast. These two signals are then remixed to form the final output.

The same nonlinear mapping function, with independent parameters, is used to attenuate image details and to adjust the

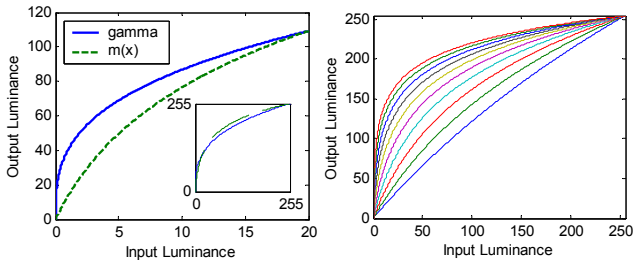


Figure 6: Plots showing our nonlinear mapping function. The left plot shows how our function does not have as severe a slope for luminances near 0 as does gamma correction as to not over accentuate dark regions (γ=2.0 for gamma correction, ψ=64 for $m(x, \psi)$). The inset shows that over the rest of 0-255, they are mostly similar. The right plot shows a family of $m(x, \psi)$ curves of ψ=2 (the most linear) through ψ=1024 (the most curved).

contrast of large scale features. It obeys the Weber-Fechner law of just-noticeable difference response in human perception but provides a parameter to adapt the logarithmic mapping in a way similar to the logmap function of Drago et al [03] and Stockham [72]. The mapping is given by:

$$m(x,\psi) = \frac{\log\left(\frac{x}{xMax}(\psi-1)+1\right)}{\log(\psi)} \quad \textbf{(5)}$$

The white level of the input luminance is set by $xMax$ and ψ controls the attenuation profile. As shown in Figure 6, the shape of our detail attenuation and contrast mapping function, $m(x, \psi)$, is similar to a traditional gamma function, but it exhibits better behavior near the origin. As noted by Drago [03] the high slope of standard gamma correction for low intensities can result in loss of detail in shadow regions. This is particularly troublesome for underexposed images like those we target.

Tone mapping (Fig. 7) begins by extracting the luminance of each frame and the chrominance ratio of each color component as discussed by Eisemann and Durand [04]. A bilateral filter is then applied to the log-image to extract the large scale image features. A temporal bilateral filter, with narrow support (small σ_i), is then applied to maintain temporal coherence. This result is then subtracted of from the log luminance of the original image to yields the detail features.

The linear intensities of the large scale features are next uniformly tone mapped using Equation 5, with a ψ_1 of approximately 40. The log-intensities of the details are attenuated based on the brightness of the linear large scale features. With a linear attenuation, a pixel with a brightness of .5×maximum would have half of its high frequency masked. Since the confidence of details degrades at dark values, we attenuate based on the curve in Equation 5 with a different ψ_2 (often around 700.0), resulting in a steep roll off for low intensities.

The log large scale features and log detail features are recombined to generate the final output luminance. Noise in the chrominance is attenuated via standard Gaussian blurring. Finally, the luminance and chrominance ratios are then recombined.

6 Results

It is difficult to evaluate the results of our LDR video processing based on still images which fail to capture temporally varying noise. Also, it is difficult to obtain a ground truth comparison for videos with dynamic elements. Our methods degenerate to temporal averaging followed by tone mapping for static scenes. The following examples are stills taken from video sequences with moving foreground objects under low light. However, processed videos most accurately demonstrate our results.

Figure 1 depicts the entire VEC process for a noisy piece of footage of "walking fingers" with a single, dim light source. The pseudo-color image demonstrates how ASTA adapts its integration strategy in different areas of each frame. All color video footage in this paper was captured using a Sony DFW-V500 4:2:2 uncompressed video camera. The high-speed grayscale footage in the supplementary video was captured using a Point Grey Research Dragonfly Express operating at 120 frames per second. Some of the videos in the supplemental video were captured via the Point Grey Research Color Flea at 30 frames per second. Figures 8 and 9 show similar examples of our method. Figure 8 illustrates the processing of a typical LDR frame, and Figure 9 shows an example of initial poor utilization of the full dynamic range. Figure 10 illustrates the histograms of raw and processed virtual exposures. ASTA does not noticeably change the histogram from the original, but our tone mapped result shows the enhanced dynamic range of our virtual exposure approach.

7 Future Work

Our current system enhances raw uncompressed video streams offline. This allows us to consider temporal extents of arbitrary lengths into both the past and the future. Ideally, we would like to apply our methods in real-time and assume more modest resources— perhaps only a second or two of temporal state sampled at 180 fps. This would allow our enhancement algorithms to be performed in-camera prior to compression. Our approach is a good fit for next generation video cameras incorporating capabilities like those described by Bidermann et al [03]. Noise filtering prior to compression might also lead to reduced bit rates, and better support compression schemes that incorporate foreground and background models.

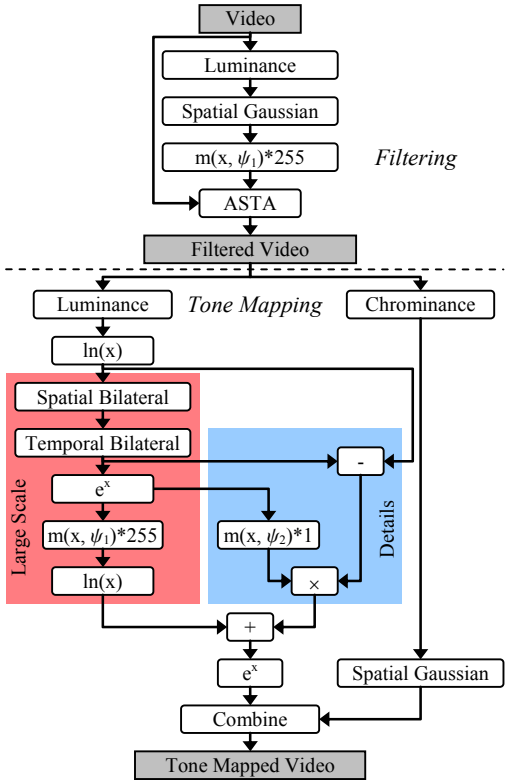


Figure 7: A flowchart of the entire process for creating virtual exposures, including detail of the LDR tone-mapping process. The highlighted areas show the different processing paths of large scale and detail features.

Our current implementation is slow since it relies on multiple non-linear filtering steps. Currently, the processing of 640x480 video takes approximately one minute per frame, and the processing times depend on the lighting level (since the filter's temporal extent varies with luminance) and various parameters that control the filter extents. Durand and Dorsey [02] discuss a "fast-bilateral" approximation which would significantly improve our system's performance.

We have demonstrated the effectiveness of our methods for moving cameras, but its success depends on the ability to reliably track features in an underexposed video sequence. This becomes more difficult when the image is composed of many independently moving regions. Our moving camera scenes currently stabilize only the single largest flow field, which was the background in our experiments. A more general solution would establish temporal correspondence for all image regions, perhaps by using optical flow methods. However, it is likely that typical optical flow techniques, which depend on robust gradient estimates, would fail on our noisy underexposed source images.

In severely underexposed video sequences it is difficult to disambiguate moving foreground elements from the background, due to insufficient spatial-neighborhood dissimilarity values. This can result in visible ghosting artifacts, as seen near where the fingers occlude the far edge of the table-top in the lower-right image of Figure 1. Our tone mapper reduces these artifacts by assigning low confidences to details in dark areas. Likewise, increasing the SAD window size helps to reduce these artifacts.

8 Conclusions

We have presented a conceptual model of a Virtual Exposure Camera as a framework for enhancing noisy, underexposed, and poorly exposed video sequences. Our system enhances each frame's dynamic range subject to a tone-mapping objective, thus computing a perceptually consistent image rather than a photometric measurement. We employ an adaptive filtering approach that simulates virtual exposure control and transitions from a temporal to a spatial filter depending on the motion in the scene. We have presented a tone-mapping algorithm targeted at the special needs of LDR imagery. With our virtual exposure framework, we can process underexposed video sequences such that they have a visual appearance similar to that of a well-exposed video, or, ideally, a compressed HDR video. Finally we are able to restore virtually unwatchable videos into acceptable footage with reduced noise, improved dynamic range, and preserved motion.

9 Acknowledgements

The authors wish to thank our actors: Aaron Block, Tim Malone, John Moriconi, and the pyramid of marshmallows. Thanks also to Jingyi Yu for his input. This work was sponsored through DARPA-funded, AFRL-managed Agreement FA8650-04-2-6543.

10 References

ACOSTA-SERAFINI, P. M., MASAKI, I., and SODINI, C.G. 2004, Predictive Multiple Sampling Algorithm with Overlapping Integration Intervals for Linear Wide Dynamic Range Integrating Image Sensors. *IEEE Transactions on Intelligent Transportation Systems*, 5, 1, 33-41.

BARASH, D. 2002. A Fundamental Relationship Between Bilateral Filtering, Adaptive Smoothing, and the Nonlinear Diffusion Equation. *Transactions on Pattern Matching and Machine Learning*, 24, 6, 844-847.

BENNETT E.P. and MCMILLAN, L. 2003. Proscenium: A Framework for Spatio-Temporal Video Editing. *In Proceedings of ACM Multimedia 2003, 177-183.*

BIDERMANN, W., EL GAMAL, A., EWEDEMI, S., REYNERI, J., TIAN, H., WILE, D., and YANG, D., 2003. A .18μm High Dynamic Range NTSC/PAL Imaging System-on-Chip with Embedded DRAM Frame Buffer, *In Proceedings of the IEEE International Solid-State Circuits Conference*, 212-213.

BOOMGAARD R. v. d., and WEIJER, J. v. d. 2002. On the Equivalence of Local-Mode Finding, Robust Estimation and Mean Shift Analysis As Used In Early Vision Tasks. *In Proceedings of the International Conference on Pattern Recognition,* 927-930.

CHOUDHURY, P. and TUMBLIN, J. 2003. The Trilateral Filter for High Contrast Images and Meshes. *In Proceedings of the Eurographics Symposium on Rendering 2003*. 1-11.

COHEN, M., COLBURN, A., and DRUCKER, S. 2003. Image Stacks. *Microsoft Research Technical Report,* MSR-TR-2003-40.

DEBEVEC, P. E. and MALIK, J. 1997. Recovering High Dynamic Range Radiance Maps from Photographs. *In Proceedings of ACM SIGGRAPH 1997*, ACM SIGGRAPH / Addison Wesley, Computer Graphics Proceedings, Annual Conference Series, 369-378.

DRAGO, F., MYSZKOWSKI, K., ANNEN, T., and CHIBA, N. 2003. Adaptive Logarithmic Mapping for Displaying High Contrast Scenes. *In Proceedings of EUROGRAPHICS 2003*, 22, 3, 419-426.

DUBOIS, E. and SABRI, S., 1984. Noise Reduction in Image Sequences Using Motion-Compensated Temporal Filtering, *IEEE Transactions on Communications*, 32, 7, 826-831.

DURAND, F. and DORSEY, J. 2002. Fast Bilateral Filtering for the Display of High-Dynamic Range Images. *ACM Transactions on Graphics*, 21, 3, 257-266.

EISEMANN, E. and DURAND, F. 2004. Flash Photography Enhancement via Intrinsic Relighting. *ACM Transactions on Graphics*, 23, 3, 670-675.

FATTAL, R., LISCHINSKI, D., and WERMAN, M. 2002. Gradient Domain High Dynamic Range Compression. *ACM Transactions on Graphics*, 21, 3, 249-256.

FRANCIS, J. J. and JAGER, G. D. 2003. The Bilateral Median Filter. *In Proceedings of the 14th Symposium of the Pattern Recognition Association of South Africa*.

JOBSON, D. J., RAHMAN, Z.-U., and WOODELL, G. A. 1997. A Multiscale Retinex for Bridging the Gap Between Color Images and the Human Observation of Scenes. *IEEE Transactions on Image Processing*, 6, 7, 965-976.

JOSTSCHULTE, K., AMER, A., SCHU, M., and SCRODER, H., 1998. Perception Adaptive Temporal TV-Noise Reduction Using Contour Preserving Prefilter Techniques, *IEEE Transactions on Consumer Electronics*, 44, 3 (August), 1091-1096.

KANG, S. B., UYTTENDAELE, M., WINDER, S., and SZELISKI, R. 2003. High Dynamic Range Video, *ACM Transactions on Graphics,* 22, 3, 319-325.

LEE, S. H. and KANG, M. G. 1998. Spatio-Temporal Video Filtering Algorithm based on 3-D Anisotropic Diffusion Equation. *In Proceedings of the International Conference on Image Processing, 98*, 2, 447-450.

LIU, X., and EL GAMAL, A., 2003. Synthesis of High Dynamic Range Motion Blur Free Image From Multiple Captures. *IEEE Transactions on Circuits and Systems, Fundamental Theory and Applications*, 50, 4, 530-539.

NAYAR, S. and BRANZOI, V. 2003. Adaptive Dynamic Range Imaging: Optical Control of Pixel Exposures over Space and Time. *In Proceedings of the International Conference on Computer Vision*, 1-8.

NAYAR, S. and BRANZOI, V. 2004. Programmable Imaging Using a Digital Micromirror Array. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 436-443.

PATTANAIK, S. N., TUMBLIN, J., YEE, H. and GREENBERG, D. 2000. Time Dependent Visual Adaptation for Fast Realistic Image Display. *In Proceedings of ACM SIGGRAPH 2000*, ACM SIGGRAPH / Addison Wesley, Computer Graphics Proceedings, 47-54.

PERONA, P. and MALIK, J. 1990. Scale-Space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions of Pattern Matching and Machine Intelligence*, 12, 7, 629-639.

PETSCHNIGG, G., AGRAWALA, M., HOPPE, H., SZELISKI, R., COHEN, M.F., and TOYAMA, K. 2004. Digital Photograph with Flash and No-Flash Pairs. *ACM Transactions on Graphics*, 23, 3, 661-669.

RASKAR, R., ILIE, A., and YU, J. 2004. Image Fusion for Context Enhancement and Video Surrealism. *In Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering*, 85-94.

REIBEL, Y., JUNG, M., BOUHIFD, M., CUNIN, B., and DRAMAN, C. 2003. CCD or CMOS Camera Noise Characteristics. *In Proceedings of the European Physical Journal of Applied Physics*, 75-80.

SAND, P. and TELLER, S. 2004. Video Matching. *ACM Transactions on Graphics*, 23, 3, 592-599.

STOCKHAM, T.G. 1972. Image Processing in the Context of a Visual Model, *In Proceedings of the IEEE*, 60, 828-842.

TOMASI, C. and MANDUCHI, R. 1998. Bilateral Filtering for Gray and Color Images. *In Proceedings of the International Conference on Computer Vision,* 836-846.

TUMBLIN, J. and RUSHMEIER, H.E. 1993. Tone Reproduction for Realistic Images. *IEEE Computer Graphics and Applications*,13,6,42-48.

TUMBLIN, J. and TURK, G. 1999. LCIS: A boundary hierarchy for detail preserving contrast reductions. *In Proceedings of SIGGRAPH 1999*,83-90.

WARD, G. 1991. Real Pixels. *Graphics Gems II*. Academic Press. 80-83.

YEE, H., PATTANAIK, S, and GREENBERG, D. P. 2001. Spatio-Temporal Sensitivity and Visual Attention for Efficient Rendering of Dynamic Environments. *ACM Transactions on Graphics*, 20, 1, 39-65.

Figure 8: A frame from a video processed using virtual exposures. Upper Left: original frame; Upper Right: histogram stretched version; Bottom Left: red = number of temporal pixels integrated, green = number of spatial pixels integrated; Bottom Right: our result after filtering and tone mapping.

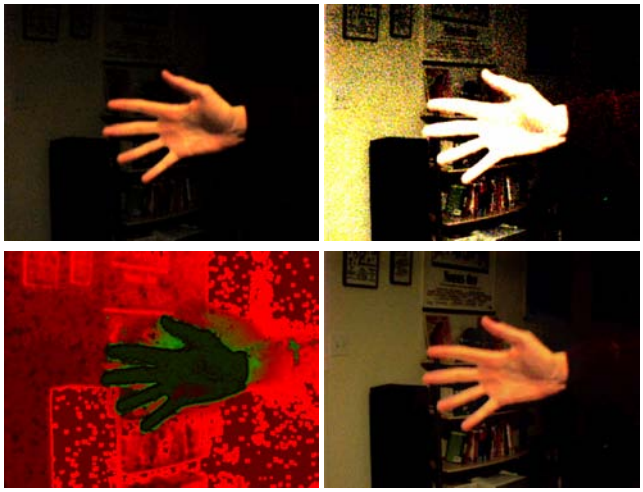


Figure 9: A frame from a video processed using virtual exposures. Upper Left: original frame; Upper Right: histogram stretched version; Bottom Left: red = number of temporal pixels integrated, green = number of spatial pixels integrated; Bottom Right: our result after filtering and tone mapping.

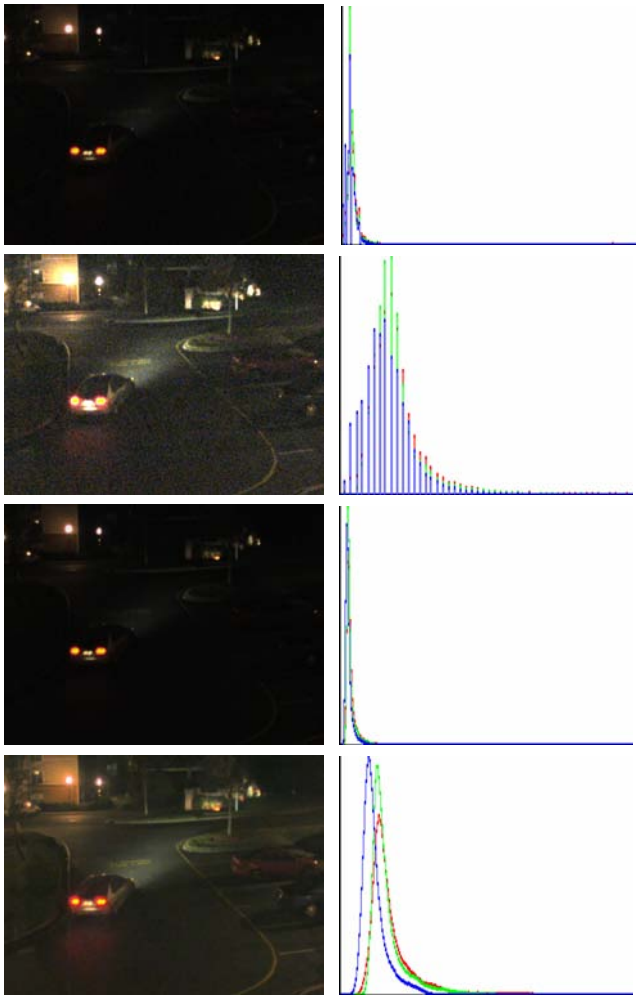


Figure 10: Inspection of color histograms in our process. From top to bottom: the original video frame and its histogram; a histogram stretched frame and its histogram showing quantization error; an ASTA processed frame and its histogram which is similar to the unfiltered histogram; the tone mapped ASTA frame and its stretched histogram without quantization error. Note the vertical scale in these histograms is vertically stretched to show maximum detail in each.

Fast Median and Bilateral Filtering

Ben Weiss†
Shell & Slate Software Corp.

Abstract

Median filtering is a cornerstone of modern image processing and is used extensively in smoothing and de-noising applications. The fastest commercial implementations (e.g. in Adobe® Photoshop® CS2) exhibit $O(r)$ runtime in the radius of the filter, which limits their usefulness in realtime or resolution-independent contexts. We introduce a CPU-based, vectorizable $O(\log r)$ algorithm for median filtering, to our knowledge the most efficient yet developed. Our algorithm extends to images of any bit-depth, and can also be adapted to perform bilateral filtering. On 8-bit data our median filter outperforms Photoshop's implementation by up to a factor of fifty.

CR Categories: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems – Sorting and Searching; I.4.3 [Image Processing and Computer Vision]: Enhancement – Filtering ; D.2.8 [Software Engineering]: Metrics – Complexity Measures; E.1 [Data Structures]: Arrays

Keywords: median filtering, bilateral filtering, rank-order filtering, sorting, image processing, algorithms, histograms, data structures, complexity, SIMD, vector processing

1 Introduction

1.1 Median Filtering

The median filter was introduced by Tukey [1977], and over the years tremendous effort has gone into its optimization and refinement. It provides a mechanism for reducing image noise, while preserving edges more effectively than a linear smoothing filter. Many common image-processing techniques such as rank-order and morphological processing are variations on the basic median algorithm, and the filter can be used as a steppingstone to more sophisticated effects. However, due to existing algorithms' fundamental slowness, its practical use has typically been restricted to small kernel sizes and/or low-resolution images.

†ben@shellandslate.com

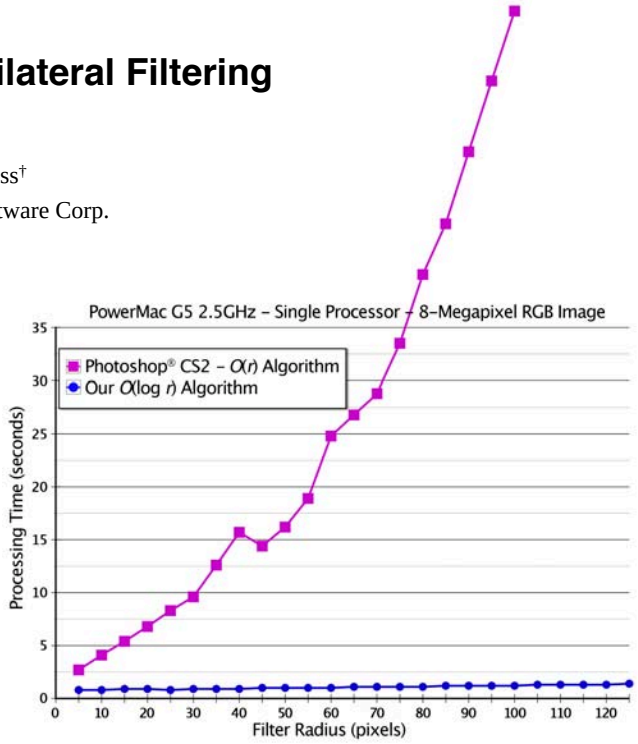


Figure 1: 8-Bit Median Filter Performance

Adobe® Photoshop® CS2 is the *de facto* standard for high-performance image processing, with a median filter that scales to radius 100. This filter exhibits roughly $O(r)$ runtime per pixel, a constraint which significantly reduces its performance for large filtering kernels. A variety of $O(r)$ algorithms are well known (e.g. Huang 1981), but it is not obvious that a faster algorithm should exist. The median filter is not separable, nor is it linear, and there is no iterative strategy for producing the final result, as there is with e.g. Gaussian Blur [Heckbert 1986], or the Fast Fourier Transform [Cooley et al. 1965]. A fast, high-radius implementation would be of considerable theoretical and practical value.

Gil et al. [1993] made significant progress with a tree-based $O(\log^2 r)$ median-filtering algorithm, but its per-pixel branching nature renders it ill-suited for deep-pipelined, vector-capable modern processors. Other efforts have resorted to massive parallelism on the presumption that a single processor is insufficient: according to Wu et al. [2003], "...designing a parallel algorithm to process [the median filter] is the only way to get a real-time response." Ranka et al. [1989] proposed a parallel algorithm with a processor-time complexity of $O(\log^4 r)$, but this curve actually scales worse than linear for $r < 55$ ($= e^4$), the point at which a 1% increase in radius corresponds to a 1% increase in computation.

Our algorithm overcomes all of these limitations and achieves $O(\log r)$ runtime per pixel on 8-bit data, for both median and bilateral filtering. It is fully vectorizable and uses just $O(r)$ storage. It also adapts as an $O(\log^2 r)$ algorithm to arbitrary-depth images, on which it runs up to twenty times as fast as Photoshop's 16-bit Median filter. To our knowledge, the presented $O(\log r)$ algorithm is the most efficient 2D median filter yet developed, and processes 8-bit data up to fifty times faster than Photoshop's Median filter.

Figure 2: Median Filter Variations. Top row: original; sharpened with Gaussian; sharpened with median (note fewer halo artifacts.) Middle row: Filtered at 20th; 50th [median]; and 80th percentiles. Bottom row: "High Pass" using median; bilateral smoothing filter; logarithmic bilateral filter.

1.2 Bilateral Filtering

The Bilateral filter was introduced by Tomasi et al. [1998] as a non-iterative means of smoothing images while retaining edge detail. It involves a weighted convolution in which the weight for each pixel depends not only on its distance from the center pixel, but also its relative intensity. As described, the bilateral filter has nominal $O(r^2)$ computational cost per pixel. Photoshop® CS2's 16-bit Surface Blur filter reflects this $O(r^2)$ complexity, and becomes unusably slow for even moderate radii. On 8-bit data, Photoshop's Surface Blur exhibits a performance curve nearly identical to its 8-bit Median filter, suggesting that they share the same core $O(r)$ algorithm.

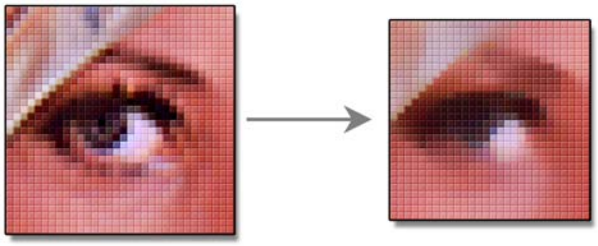
Durand et al. [2002] developed a much more efficient technique, refined and accelerated by Paris et al. [2006]. Durand's method approximates the bilateral by filtering subsampled copies of the image with discrete intensity kernels, and recombining the results using linear interpolation. It has the paradoxical property of becoming *faster* as the radius increases (due to greater subsampling), but also has some potential drawbacks. For one, it is not translation-invariant: the exact output is dependent on the phase of the subsampling grid. Also, the discretization may lead to a further loss of precision, particularly on high-dynamic-range images with narrow intensity-weighting functions.

Our bilateral filtering algorithm maintains high resolution in both space and intensity, and is translation-invariant. It is based on a box spatial kernel, which can be iterated to yield smooth spatial falloff. It is derived from the same core algorithm as our fast $O(\log r)$ median filter, and adapts to 16-bit and HDR data with minimal loss of precision.

1.3 Structure

Our approach in this paper will be first to illustrate the conventional $O(r)$ median algorithm for 8-bit images, and analyze its performance and limitations. Then we will show in steps how to improve it; first by constant factors, then into $O(\sqrt{r})$ and $O(\sqrt[3]{r})$ algorithms, and from there into an $O(\log r)$ algorithm. We will show how our approach adapts to higher bit-depth data, such as 16-bit and HDR floating-point. Finally, we will show how the algorithm can be adapted to perform bilateral filtering, and compare it with previous methods.

2 The Basic $O(r)$ Algorithm

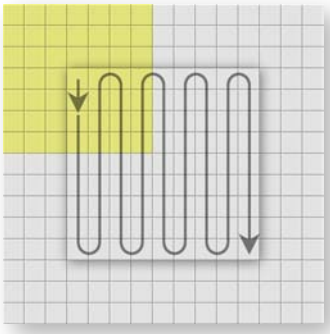


Consider the case of applying a radius-r median filter to an 8-bit image. Assume a source image that is larger than the destination by r pixels on all sides, to sidestep edge-related concerns. (In practice, we repeat edge pixels to fill undefined areas, and process color images on a per-channel basis.) Because the median filter is *local*, it can be applied to arbitrary-size images in tiles. As a consequence, its total runtime scales linearly with image area: $O(S^2)$ for an S-by-S image.

The fundamental property that concerns us here is *runtime per pixel*, as a function of filter radius. This corresponds to the performance a user will experience while adjusting the filter radius, and is the primary differentiating characteristic between median-filtering algorithms. For reference, a brute-force implementation can calculate each output pixel in $O(r^2 \log r)$ time, by sorting the corresponding $(2r + 1)^2$ -pixel input window and selecting the median value as output.

On discrete data, a radix-sort can be used to reduce the sorting complexity to $O(r^2)$ operations; this can be done for some floating-point data as well [Terdiman 2000]. In the case of 8-bit data, we use a 256-element histogram, H. Once the input values are added to H, the median value lies in the first index for which the sum of values to that index reaches $2r^2 + 2r + 1$. The median index can be found by integrating the histogram from one end until the appropriate sum is reached.

An improved algorithm was proposed by Huang [1981], based on the observation that adjacent windows overlap to a considerable extent. Huang's algorithm makes use of this sequential overlap to consolidate the redundant calculations, reducing the computational complexity to $O(r)$. A modified version of Huang's algorithm is below:



r: radius of median filter. (shown above as *r* = 3.)
H: 256-element histogram.
I: input image, *S* + 2*r* pixels square.
O: output image [inset], *S* pixels square.

```
initialize H to I[0 .. 2r][0 .. 2r].  // yellow region
find median value m in H, write m to O[0][0].
   for row = 1 to S - 1:
      add values I[2r + row][0 .. 2r] to H.
      subtract values I[row - 1][0 .. 2r] from H.
      find median value m in H; write m to O[row][0].
   step sideways to next column (and process bottom to top, etc.).
```

Figure 3: Pseudocode for Huang's $O(r)$ Algorithm

Huang's algorithm is a significant improvement over the brute-force method. However, the window-sliding step dominates the calculation with $O(r)$ runtime per pixel, while the histogram-scanning takes constant time per pixel. This suggests that we should look for a way to make the window-sliding faster, even at the expense of making the histogram-scanning slower.

Observe that as the window zigzags through the image, it passes through each region several times, performing nearly the same operations on each pass. (Picture mowing your lawn back and forth, shifting sideways one centimeter each time.) This redundancy is considerable, and mirrors the adjacent-window overlap that led to Huang's algorithm.

The difficulty is that these redundant calculations occur at widely spaced time intervals in the computation; perhaps tens of thousands of processor cycles apart, so they cannot be combined using the same sequential logic that led to the $O(r)$ technique. Yet, eliminating these redundancies is the key to a dramatically faster algorithm.

3 The *O*(log *r*) Algorithm

3.1 Synchronicity

The fundamental idea behind this paper, and the mechanism that enables our fast algorithm, is the observation that *if multiple columns are processed at once*, the aforementioned redundant calculations *become sequential*. This gives us the opportunity to consolidate them, resulting in huge increases in performance.

3.2 Distributive Histograms

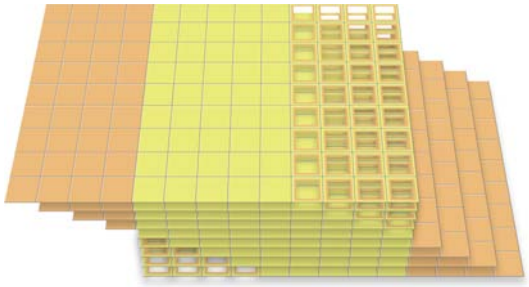
A straightforward adaptation of Huang's algorithm to process N columns at once involves the maintenance of N histograms, one per output column: H_0 .. H_{N-1}. This is essentially just a rearrangement of operations; the runtime complexity is unchanged. Each input pixel gets added to $2r + 1$ histograms over the course of filtering the image, leading to the $O(r)$ runtime complexity.

Fortunately, the explicit maintenance of each histogram H_n is unnecessary, due to the *distributive* property of histograms. This is where our approach diverges from Huang's algorithm. Histogram distributivity means that for disjoint image regions A and B:

$$H_{A\cup B}[v] \equiv H_A[v] + H_B[v] \qquad (1)$$

In other words, if an image window W is the union of two disjoint regions A and B, then its histogram H_W is equal to $H_A + H_B$. The median element of W can then be found by scanning the *implicit* histogram H_W, splicing it together from H_A and H_B on the fly. (This extends to signed linear combinations; $H_A \equiv H_W - H_B$, etc.)

In the case of median-filtering N columns, our approach is to form a set H^* of *partial* histograms P_0 .. P_{N-1} (whose elements may be signed), such that each histogram H_0 .. H_{N-1} is representable as the *sum* of T partial histograms from H^*. Figure 4 shows how a row of pixels v_0 .. v_{2r+8} is added to H^*, for the case $N = 9$, $T = 2$.



Huang: H_0 .. H_8	Our Method: P_0 .. $P_8 \subseteq H^*$	Mapping:
$H_0[v_0$ .. v_{2r}]++;	$P_0[v_0..v_3]$++; $P_0[v_{2r+1}..v_{2r+4}]$--;	$H_0 \equiv P_0 + P_4$
$H_1[v_1$ .. $v_{2r+1}]$++;	$P_1[v_1..v_3]$++; $P_1[v_{2r+2}..v_{2r+4}]$--;	$H_1 \equiv P_1 + P_4$
$H_2[v_2$ .. $v_{2r+2}]$++;	$P_2[v_2..v_3]$++; $P_2[v_{2r+3}..v_{2r+4}]$--;	$H_2 \equiv P_2 + P_4$
$H_3[v_3$ .. $v_{2r+3}]$++;	$P_3[\ v_3]$++; $P_3[\ v_{2r+4}]$--;	$H_3 \equiv P_3 + P_4$
$H_4[v_4$ .. $v_{2r+4}]$++;	$P_4[v_4$ ..... $v_{2r+4}]$++;	$H_4 \equiv P_4$
$H_5[v_5$ .. $v_{2r+5}]$++;	$P_5[v_4\ ]$--; $P_5[v_{2r+5}\ ]$++;	$H_5 \equiv P_5 + P_4$
$H_6[v_6$ .. $v_{2r+6}]$++;	$P_6[v_4..v_5]$--; $P_6[v_{2r+5}..v_{2r+6}]$++;	$H_6 \equiv P_6 + P_4$
$H_7[v_7$ .. $v_{2r+7}]$++;	$P_7[v_4..v_6]$--; $P_7[v_{2r+5}..v_{2r+7}]$++;	$H_7 \equiv P_7 + P_4$
$H_8[v_8$ .. $v_{2r+8}]$++;	$P_8[v_4..v_7]$--; $P_8[v_{2r+5}..v_{2r+8}]$++;	$H_8 \equiv P_8 + P_4$
$18r + 9$ operations	$2r + 41$ operations	$T = 2$

Figure 4: Adding a row of pixels to H^*, for the case $N = 9$, $T = 2$. Each layer shows how the corresponding histogram H_n is formed from partial histograms P_n in H^*. The pseudocode shows how a row of pixels v_0 .. v_{2r+8} is added to H^*. The "holes" represent pixels that are added to the central histogram P_4 but subtracted from partial histograms, canceling themselves out.

The histogram set H^* is arranged like a tree, with a central histogram (P_4) representing the input window for the central column, and the other partial histograms P_n representing the *difference* between the central and adjacent windows. The sum of each partial plus central histogram yields the full histogram for the corresponding square input window. By widening the yellow central region and fitting the partial histograms to its edges, the 9-column technique can be adapted to perform median filtering of arbitrary radius. The time spent modifying H^* is still $O(r)$, but with a much lower constant than Huang's algorithm. The median extraction time from H^* remains constant regardless of r.

The more fundamental improvement in efficiency comes when we allow the number of columns N to *vary* with r, conceptually adding more planes to Figure 4. For N output columns, the number of modifications to H^* per output pixel is $(N^2 + 4r + 1) / N$. (The graphic in Figure 4 show the case of $N = 9$, $r = 4$, requiring 98 adjustments to H^* per row or about 11 per output pixel.) Solving for N to minimize the number of adjustments gives $N \approx 2\sqrt{r}$, which yields $O(\sqrt{r})$ histogram modifications per pixel. Thus, the complexity of the $T = 2$, variable-N adaptive algorithm is $O(\sqrt{r})$.

3.3 Three Tiers and Beyond

Figure 5: H^* Histogram Layout for $N = 63$, $T = 3$.

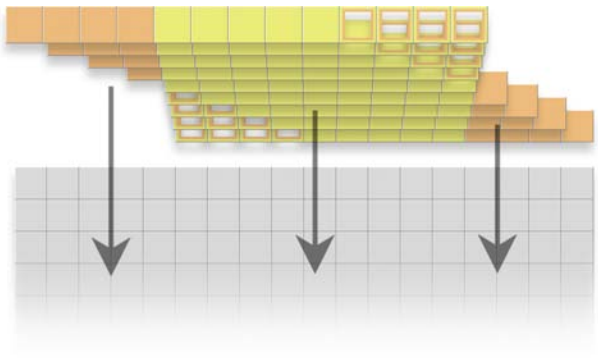
Figure 5 shows a layout for processing sixty-three columns at once. It is the three-tiered analogue of Figure 4, this time "viewed" from the side. There is a single shared histogram P_{31} [yellow] corresponding to the central window; eight partial histograms [orange] at seven-pixel intervals; and for each of these, six small partial histograms [red] at unit intervals; sixty-three histograms altogether. Each input pixel is added/subtracted to each histogram intersecting its column. In this example, a 63-by-1 block of output is produced at each iteration. The mapping of P_n to H_n becomes:

$$H_n = P_{31} + P_{7\lfloor n/7 \rfloor + 3} + P_n \qquad (2)$$

where the second and third terms are ignored if they match earlier terms (e.g., $H_{24} = P_{31} + P_{24}$.) The structure of H^* is recursive; the central yellow histogram forms a rough approximation to any particular H_n; the orange partial histograms refine that approximation, and the red histograms provide the final correction to make the sums exact. Once H^* is initialized, the full histogram of each of the 63 square input windows is expressible per Eq. 2 as the sum of one red histogram (or none), one orange histogram (or none), and the yellow central histogram. The illustrated case of $N = 63$, $T = 3$, $r = 31$ requires ~18 histogram modifications per output pixel. The median-extraction from H^* takes constant time, as the three partial histograms are spliced together on the fly.

For the general case of 3-tiered structures, processing N columns at once and with tier radix $\sqrt{N}$, the number of histogram adjustments per output pixel becomes $\sqrt{N} + ((4r + 2) / N)$. For radius r, solving for optimal N yields $N \approx 4r^{2/3}$, and the runtime of the three-tiered adaptive algorithm is therefore $O(\sqrt[3]{r})$.

In practice, three tiers covers the realistic range of implementation (into the hundreds), but our technique can be extended to arbitrary T. In the limit, a radius-r median filter can be computed across $N = O(r)$ columns at once, using N histograms arranged into $T = O(\log r)$ tiers of constant radix. For example, a radius *one-million* median filter can be computed across $N = 9^6 = 531{,}441$ columns at once, using 9^6 partial histograms arranged in seven tiers of radix 9, occupying roughly 500 megabytes of storage. Sliding the window from one row to the next requires $O(\log r) \approx 114$ histogram modifications per output pixel. Extracting each median takes $O(\log r)$ steps; in this case splicing up to seven partial histograms together to construct each H_n, counterbalancing the $O(\log r)$ complexity of writing to H^*. Therefore, the overall computational cost per pixel is $O(\log r)$. □



r: radius of median filter.
*H**: Array of partial histograms, processing *N* columns.
I: input image, *N* + 2*r* pixels square.
O: output image, *N* pixels square.

```
for each row in [0 .. 2r]:      // Initialize H*
    Add row, I[row][0 .. 2r + N - 1] to H*, as per Figure 4
for each output pixel in O[0][col]:  // compute first N median values
    scan H_col (implicit in H*) to the find the median m,
    write O[0][col] = m.
for row = 1 to N - 1: // step from top to bottom of image
    add new bottom row, I[row + 2r][0 .. 2r + N - 1], to H*.
    subtract old top row, I[row - 1][0 .. 2r + N - 1], from H*.
    find N new median values in H*; write to O[row][0 .. N - 1].
```

Figure 6: Pseudocode for $O(\log r)$ Algorithm

3.4 Implementation Notes

Scanning the histogram from index zero to find the median takes about 128 steps on average. Huang [1981] suggested using each output value as a "pivot" to find the next median value: as H is scanned to find m, we keep track of the number of values $v < m$ in H. Then as we add and remove pixels from H, we keep a running count of how many values satisfy $v < m$. This allows us to scan the updated histogram starting from m, which is typically much faster than starting from index zero.

Figure 7: Pivot Tracking. The middle image shows the approximation obtained using one pivot per sixteen columns to track the smallest median values. H* is then scanned upwards from these pivots (several columns at a time, vectorized) to yield the exact median result, right.

This heuristic adapts to the $O(\log r)$ algorithm by using $O(\log r)$ pivots across the N columns, with each pivot tracking the smallest median value in its respective columns. This approach obtains much of the benefit of the heuristic while preserving the $O(\log r)$ complexity. Since the pivot tracking involves many consecutive bytewise compares, it is ideally suited for vector optimization.

Finally, it is useful to *interleave* the partial histograms P_n in memory, so that multiple adjacent histograms can be modified simultaneously using vector loads and stores. This greatly accelerates the reading and writing of H*.

4 Higher-Depth Median Filtering

4.1 Adapting the 8-bit Algorithm

16-bit and HDR images have already become mainstream, so it is important that our median filter work with images of arbitrary bit-depth. A direct extension of the 8-bit algorithm is problematic, because the histograms must stretch to accommodate every possible value, growing exponentially with bit-depth. The algorithm still remains $O(\log r)$, but storage considerations render it impractical for 16-bit images and impossible for floating-point images.

4.2 The Ordinal Transform

H* is reduced to a manageable size through a technique we call the *ordinal transform*. This involves *sorting* the input image values, storing the sorted list, and replacing each cardinal value with its ordinal equivalent. (Duplicate cardinal values map to consecutive ordinal values.) The median filter is then applied to the ordinal image, and the transform is inverted to restore the cardinal-valued result. The ordinal transform operates on images of any depth, in logarithmic or constant time per pixel.

In this operation, the nonlinearity of the median filter is crucial. Any linear filter (e.g., Gaussian blur) would not be invariant under the ordinal transform, but the median filter is! That is because rank-order is preserved; the k^{th}-smallest cardinal value maps to the k^{th}-smallest ordinal value. After the ordinal transform is applied, the median filtering proceeds as in Section 3, this time using single-bit histograms P_n (sufficient here because each ordinal value is unique in the image), and the results are inverse-transformed to yield the final filtered image.

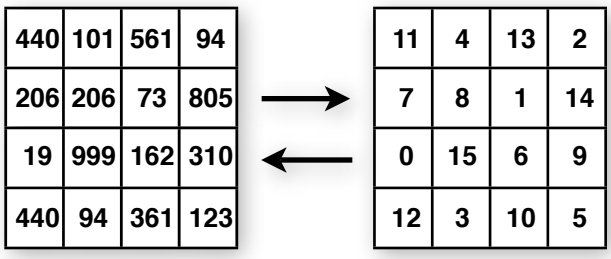


Figure 8: The Ordinal Transform. Duplicate cardinal values (e.g. 94, 94, left) map to consecutive ordinal values (2 and 3, right).

Recall that the histogram elements in H* can go negative. At first this appears problematic because the required range [-1, 0, 1] doesn't fit into a single bit. However, since each summed implicit histogram value $H_n[v]$ can only be either zero or one, only the lowest bit from each partial histogram must participate in the summation. Hence a single bit is sufficient for each element of P_n, and the splicing accomplished through a bitwise XOR.

4.3 The Compound Histogram

For processing N columns in parallel, this approach still requires the allocation and maintenance of N single-bit histograms. However, due to the uniqueness of values in the ordinal image, we can take advantage of a much more efficient encoding.

Consider the full histogram obtained by splicing the n^{th} set of partial histograms in H* (consisting of the central histogram plus one partial histogram from each tier), to yield the single-bit histogram for the n^{th} input window. Label this binary histogram B_n. By definition, the single bit $B_n[v]$ indicates whether the ordinal value v lies in the input window n.

Now, for $N <= \min(2r, 128)$, instead of allocating N binary histograms, we allocate a single *8-bit compound histogram* H^c. As rows of pixels $v = I$[row][col] are added, we adjust H^c as follows:

$$H^c[v] = \begin{cases} 0\text{xFF} - \text{col}, & \text{col} < N - 1 \\ 0\text{x80}, & N - 1 <= \text{col} <= 2r \\ 0\text{x80} - (\text{col} - 2r), & \text{col} > 2r \end{cases} \quad (3)$$

Since the ordinal values in I can have any arrangement, the compound histogram H^c is filled in arbitrary order. As rows of pixels are removed, the corresponding elements of H^c are zeroed. The power of this technique becomes clear when it comes time to scan the implicit histogram B_n to find the n^{th} median output value.

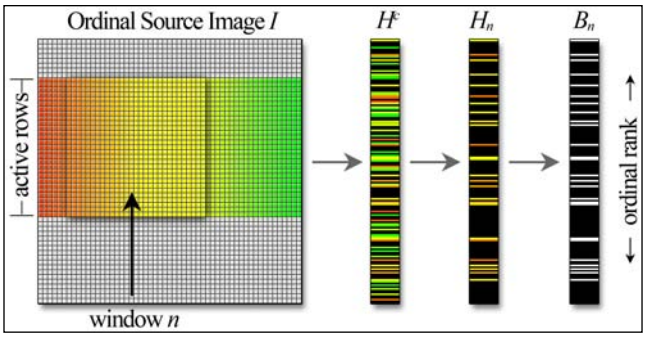


Figure 9: The Compound Histogram H^c

In our initial approach, each implicit histogram B_n was spliced together from $O(\log r)$ partial histograms, taking $O(\log r)$ time per element. With the compound histogram, using 8-bit modular arithmetic, elements of B_n can now be computed in *constant time:*

$$B_n[v] = (H^c[v] + n) >> 7. \quad (4)$$

For $N > 128$, this technique extends in a straightforward manner to 16-bit compound histograms, sufficient for $N <= 32768$, and so on. The computational complexity is independent of element size.

4.4 Coarse-To-Fine Recursion

There is one final detail. As the radius increases, the histogram size scales as $O(r^2)$, which directly affects the histogram scanning distance and thus the algorithm's time-complexity. This complication is addressed by computing the median in stages from coarse to fine precision. Alparone et al. [1994] applied a similar technique to the $O(r)$ algorithm, employing two levels of resolution to process 10, 12, or 14-bit images in faster (but still $O(r)$) time. Here we apply an analogous technique to our log-time algorithm.

In our case, the coarse-to-fine calculation is performed by right-shifting the ordinal image 8 bits at a time (or similar radix) until it reaches a fixed low resolution; e.g., 10 bits per pixel. Then the $O(\log r)$ algorithm from Section 3 is applied to the low-resolution data (whose values are no longer unique), storing not only the median values, but also the number of values *strictly below* the median. This result forms a pivot from which we calculate the median at the next-higher level of resolution. For example, if the lowest-resolution median value for a pixel is 0x84, and there are n values below 0x84 in its histogram, then there will be n values below 0x8400 in the next-higher-resolution histogram, and the median will be in [0x8400 .. 0x84FF]. This scanning is bounded by a constant [256] number of steps per iteration, with each iteration adding eight bits of precision to the output. The final iteration is performed using the compound histogram, which yields the full-precision ordinal result. The entire process requires $O(\log r)$ levels of recursion, each taking $O(\log r)$ time as shown in Section 3, for an overall computational complexity of $O(\log^2 r)$. □

4.5 Implementation Notes

Applying a radius-r median filter to an ordinal image cannot output any of the lowest $(2r^2 + 2r)$ ordinal values, because by definition the median must exceed that many values. The filter can thus treat all such values as a single low constant, and likewise the $(2r^2 + 2r)$ highest values as a single high constant, without affecting the final result. This "endpoint compression" can be incorporated into the ordinal transform, allowing input windows significantly larger than 2^{16} pixels to be filtered using 16-bit ordinal images.

Interestingly, since each ordinal value is unique, the median output for each pixel also tells us *where* in the source image that value came from, generating a vector field. On high-frequency images this field is quite noisy, but on smoother images it exhibits surprising structure. (Figure 14 on the last page is an emergent example of this structure.) Also, a variation of H^c where both row and column information is stored at each index can allow histogram elements of any computable region (e.g., a circle) to be determined in constant time. We have not fully explored these properties, but they suggest possible directions for future research.

For our implemented range of radii [1...127], the compound histogram is efficient enough not to require the coarse-to-fine recursion at all, except on carefully-constructed worst-case data. (Real-world images are invariably close to best-case.) In fact, the ordinal transform by itself is often the performance bottleneck. As shown in Figure 10, our implementation outperforms the 16-bit Median filter in Photoshop® CS2 by up to a factor of 20, with identical numerical results.

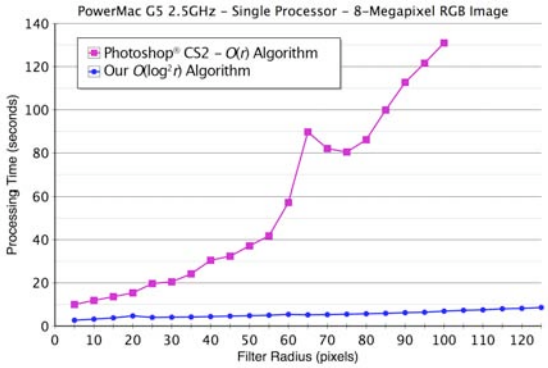


Figure 10: 16-Bit Median Filter Performance

5 The Bilateral Filter

The bilateral filter is a normalized convolution in which the weighting for each pixel p is determined by the spatial distance from the center pixel s, as well as its relative difference in intensity. In the literature (Tomasi et al. [1998] and Durand et al. [2002]), the spatial and intensity weighting functions f and g are typically Gaussian; Photoshop® CS2 implements a box spatial filter and triangular intensity filter. These functions multiply together to produce the weighting for each pixel. For input image I, output image J and window Ω, the bilateral is defined as follows:

$$J_s = \sum_{p \in \Omega} f(p-s)g(I_p - I_s)I_p \Big/ \sum_{p \in \Omega} f(p-s)g(I_p - I_s). \quad (5)$$

The special case of a spatial box-filter (with arbitrary intensity function) is worth studying, because the weighting function becomes *constant* for all pixels of a given intensity. Under this condition, the *histogram* of each spatial window becomes sufficient

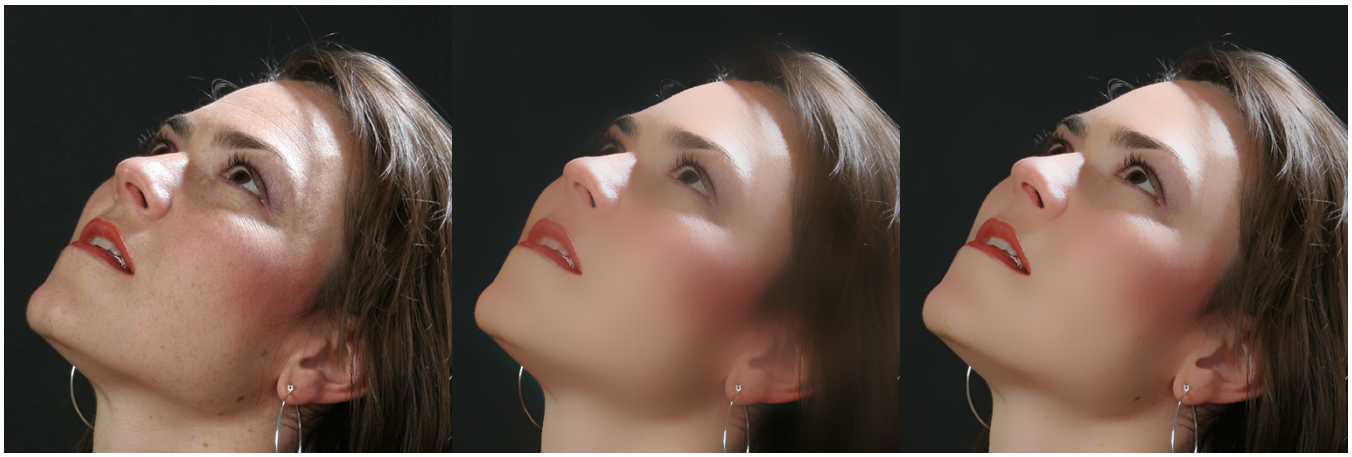


Figure 11: The Bilateral Filter. From left: 8-Bit Source Image; Linear-Intensity Bilateral (Eq. 5); Logarithmic-Intensity Bilateral (Eq. 6).

to perform the filtering operation. Our $O(\log r)$ median-filtering algorithm already generates these histograms, so the bilateral convolution can be appended in constant time per pixel, scaling with the support of the intensity function g.

For higher-precision data, one can either dither the source data into 8 bits before processing (which introduces surprisingly little error), or else downsample the source intensities into the histograms (along the lines of Paris et al. [2006]), which requires larger histogram elements but yields better accuracy. Durand et al. [2002] applied the bilateral to log-scaled images and re-expanded the result, but this approach can pose precision problems when filtering 8-bit data. Fortunately, this logarithmic approach can be approximated on linear data by scaling the width of g in proportion to the intensity of the center pixel while biasing the weight toward smaller values, yielding a new function g'. The rightmost image in Figure 11 shows the result of this logarithmic bilateral on 8-bit data, using a simple variable-width triangular function for g'. (Note the improved lip color and hair detail.) More sophisticated intensity functions can be precomputed for all (I_p, I_s). Our linear-data approximation to the logarithmic bilateral is as follows:

$$J_s = \sum_{p\in\Omega} f(p-s)g'(I_p/I_s)I_p \Big/ \sum_{p\in\Omega} f(p-s)g'(I_p/I_s). \quad (6)$$

where

$$g'(x) = g(\log x)/\sqrt{x}. \quad (7)$$

One potential concern with our histogram-based method is the imperfect frequency response of the spatial box filter. Visual artifacts may resemble faint mach bands, but these artifacts tend to be drowned out by the signal of the preserved image (e.g., the images in Figure 11 are box-filtered.) Still, smooth spatial falloff is achievable with our method, using an iterative technique. Direct iteration of the bilateral can yield an unintentionally cartoonish look [Tomasi 1998], but *indirect* iteration is more effective. At each step the output is re-filtered, while continuing to use the *original* data for the intensity windows. For homogeneous areas or with wide intensity kernels, this converges to a Gaussian without creating the cartoonish look:

$$I_s^{n+1} = \sum_{p\in\Omega} f(p-s)g(I_p^n, I_s^0)I_p^n \Big/ \sum_{p\in\Omega} f(p-s)g(I_p^n, I_s^0). \quad (8)$$

Figure 12: Original; One iteration; Three iterations (Eq. 8).

For the special case of the box-weighted bilateral, our technique achieves the discrete-segments result of Durand et al. [2002] in similar time, but with 256 segments instead of 10-20, and at full spatial resolution. This makes the result translation-invariant (avoiding artifacts due to the phase of the subsampling grid), and the high segment count allows high-dynamic-range images to be filtered with minimal loss of precision. Slight color artifacts may be introduced as a result of processing the image by channel, but we have found these also to be imperceptible on typical images.

With a single iteration and a fixed triangular intensity function (support 80 levels), our results numerically match Photoshop's Surface Blur output, with up to twenty-fold acceleration. The performance bottleneck (over 80% of the calculation) is the constant time spent multiplying each window's histogram by the intensity function, which accounts for the flatness of our performance curve. Reducing our implementation to 64 segments should nearly triple its speed, while maintaining very high quality results.

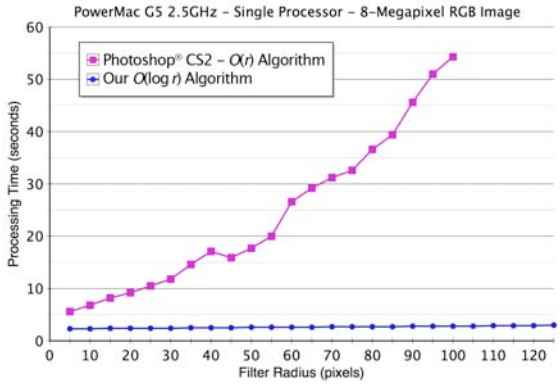


Figure 13: Bilateral Filter Performance

6 Conclusion

We have presented a logarithmic-time median filter algorithm, scalable to arbitrary radius and adaptable to images of any bit-depth. We believe this is the most efficient median algorithm yet developed, both in terms of theoretical complexity and real-world performance. Our algorithm can be extended to perform general rank-order filtering, and it is flexible enough to accomplish a wide variety of practical and creative tasks.

Significantly, we have shown that our algorithm can be adapted to perform bilateral filtering, where it becomes a highly effective noise-removal tool. Our algorithm provides a high-precision, translation-invariant, realtime implementation of the bilateral filter, and supports nonlinear intensity scaling, which greatly enhances the quality of the result.

Our algorithms have shown their advantage not only at high radii but across the spectrum. In the time it takes Photoshop® CS2 to process a 5x5 median or bilateral filter, our implementation can process any kernel up to 255x255. We have adapted our algorithm to multiple processors with near-linear performance gains, up to 3.2x faster on a four-processor system versus a single processor. The accompanying videos demonstrate the realtime performance of our median and bilateral filters.

Now that the speed of the median filter has been brought onto par with the workhorse filters of image-processing (e.g. Gaussian blur and FFT), we anticipate that the median filter and its derivatives will become a more widely used part of the standard image-processing repertoire. It is our hope that our algorithms spark renewed interest in this line of research, and we are confident that new applications and discoveries lie just around the corner.

Acknowledgments

Special thanks to Paul Heckbert for providing invaluable feedback in the early stages. Also thanks to Blaise Agüera y Arcas, Michael Herf, Klaus Schauser, Tobias Höllerer and Ian Gilman for their constructive critiques. To the talented Gretchen Elise for the use of her photo. To the reviewers for their time and insightful comments. Finally to Kai Krause, for the String Theory technique!

References

ALPARONE, L., CAPPELLINI, V., AND GARZELLI, A. 1994. A coarse-to-fine algorithm for fast median filtering of image data with a huge number of levels. Signal Processing, Vol. 39 No. 1-2, pp. 33-41.

COOLEY, J. H. AND TUKEY, J. 1965. An Algorithm for the Machine Calculation of the Complex Fourier series. Mathematics of Computation, vol. 19, pp. 297-301.

DURAND, F. AND DORSEY, J. 2002. Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. ACM SIGGRAPH 2002.

HECKBERT, P. 1986. Filtering by Repeated Integration. ACM SIGGRAPH 1986.

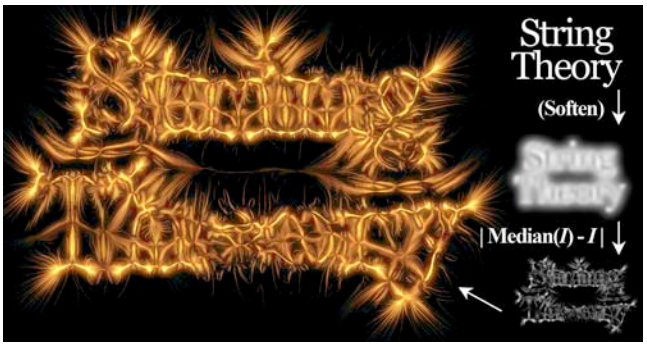


Figure 14: Kai's String Theory. A high-precision median filter is applied to a soft 16-bit mask, and the before-after ***difference*** (amplified 250x and colorized) is shown above. Far from the low-frequency result one might expect, the contours of the soft mask "beat" against the median filter's discrete sampling grid, producing an intricate filigree along rational field lines. This effect is ordinarily imperceptible, but with high amplification it lends itself to an unusual creative use of the median filter.

HUANG, T.S. 1981. *Two-Dimensional Signal Processing II: Transforms and Median Filters*. Berlin: Springer-Verlag, pp. 209-211.

GIL, J. AND WERMAN, M. 1993. Computing 2-D Min, Median, and Max Filters. IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 15 No. 5, pp. 504-507.

KABIR, I. 1996. *High Performance Computer Imaging.* Greenwich, CT. Manning Publications. pp. 181-192.

PARIS, S. AND DURAND, F. 2006. A Fast Approximation of the Bilateral Filter using a Signal Processing Approach. ECCV 2006.

PHA, T. Q. AND VLIET, L. J. V. 2005. Separable bilateral filtering for fast video preprocessing. IEEE Int. Conf. on Multimedia & Expo. CD1-4.

RANKA, S. AND SAHNI, S. 1989. Efficient Serial and Parallel Algorithms for Median Filtering. Proceeding 1989 International Conference on Parallel Processing, III-56 -- III-62.

TERDIMAN, P. 2000. Radix Sort Revisited. <http://www.codercorner.com/RadixSortRevisited.htm>

TANIMOTO, S. L. 1995. Fast Median Filtering Algorithms for Mesh Computers. Pattern Recognition, vol. 28, no. 12, pp. 1965-1972.

TOMASI , C. AND MANDUCHI , R. 1998. Bilateral filtering for gray and color images. In Proc. IEEE Int. Conf. on Computer Vision, 836–846.

TUKEY, J.W. 1977. *Exploratory Data Analysis*. Reading, MA. Addison-Wesley.

WEISS, B. 2006. Method and Apparatus for Processing Image Data. US Patent 7,010,163.

WU, C. H. AND HORNG, S. J. 2003. Fast and Scalable Selection Algorithms with Applications to Median Filtering. IEEE Transactions on Parallel and Distributed Systems, vol. 14, no. 10, pp. 983-992.

Real-Time Video Abstraction

Holger Winnemöller* Sven C. Olsen* Bruce Gooch*

Northwestern University

Figure 1: **Abstraction examples.** *Original*: Snapshots of a guard in Petra (left) and two business students (right). *Abstracted*: After several bilateral filtering passes and with DoG-edges overlayed. *Quantized*: Luminance channel soft-quantized to 12 bins (left) and 8 bins (right). Note how folds in the clothing and other image details are emphasized (stones on left and student's shadows on right).

Abstract

We present an automatic, real-time video and image abstraction framework that abstracts imagery by modifying the contrast of visually important features, namely luminance and color opponency. We reduce contrast in low-contrast regions using an approximation to anisotropic diffusion, and artificially increase contrast in higher contrast regions with difference-of-Gaussian edges. The abstraction step is extensible and allows for artistic or data-driven control. Abstracted images can optionally be stylized using soft color quantization to create cartoon-like effects with good temporal coherence. Our framework design is highly parallel, allowing for a GPU-based, real-time implementation. We evaluate the effectiveness of our abstraction framework with a user-study and find that participants are faster at naming abstracted faces of known persons compared to photographs. Participants are also better at remembering abstracted images of arbitrary scenes in a memory task.

CR Categories: I.3.3 [Computer Graphics]: Image Generation

Keywords: non-photorealistic rendering, visual perception, visual communication, image abstraction

1 Introduction

Many image stylization systems are designed for purely artistic purposes, like creating novel forms of digital art or helping laymen and artists with laborious or technically challenging tasks. Recently, several authors have proposed the goal of automatic stylization for efficient visual communication, to make images easier or faster to understand [DeCarlo and Santella 2002; Gooch et al. 2004; Raskar et al. 2004]. Although we also cater to artistic stylization (Figure 1, *Quantized*), our work focuses primarily on visual communication.

We present an automatic, real-time framework that abstracts imagery by modeling visual salience in terms of luminance and color opponency contrasts. We simplify regions of low contrast while enhancing high contrast regions. For adjusting contrasts, we employ several established image processing algorithms, which we modify for greater parallelism, temporal coherence, and directability.

We show that a separated approximation to a bilateral filter, applied iteratively, is an effective, parallelizable approximation to the process of simplifying images using anisotropic diffusion. We ensure that small input changes lead to similarly small output changes, on a frame-per-frame basis using several smooth quantization functions and avoid having to track object contours across frames.

A user study demonstrates the effectiveness of our framework for simple recognition and memory tasks, showing that our framework performs well even on small images, particularly on difficult subject matter like faces. We thus believe that visual communication applications will greatly benefit from our framework, as perceived fidelity is often paramount to actual fidelity for communication purposes. Possible applications include low-bandwidth videoconferencing and portable devices.

2 Related Work

Previous work in image-based stylization and abstraction systems varies in the use of scene geometry, video-based vs. static input, and the focus on perceptual task performance and evaluation.

Among the earliest work on image-based NPR was that of Saito and Takahashi [1990] who performed image processing operations on data buffers derived from geometric properties of 3D scenes. Our own work differs in that we operate on raw images, without requiring underlying geometry. To derive limited geometric infor-

*{holger|sven|bgooch}@cs.northwestern.edu

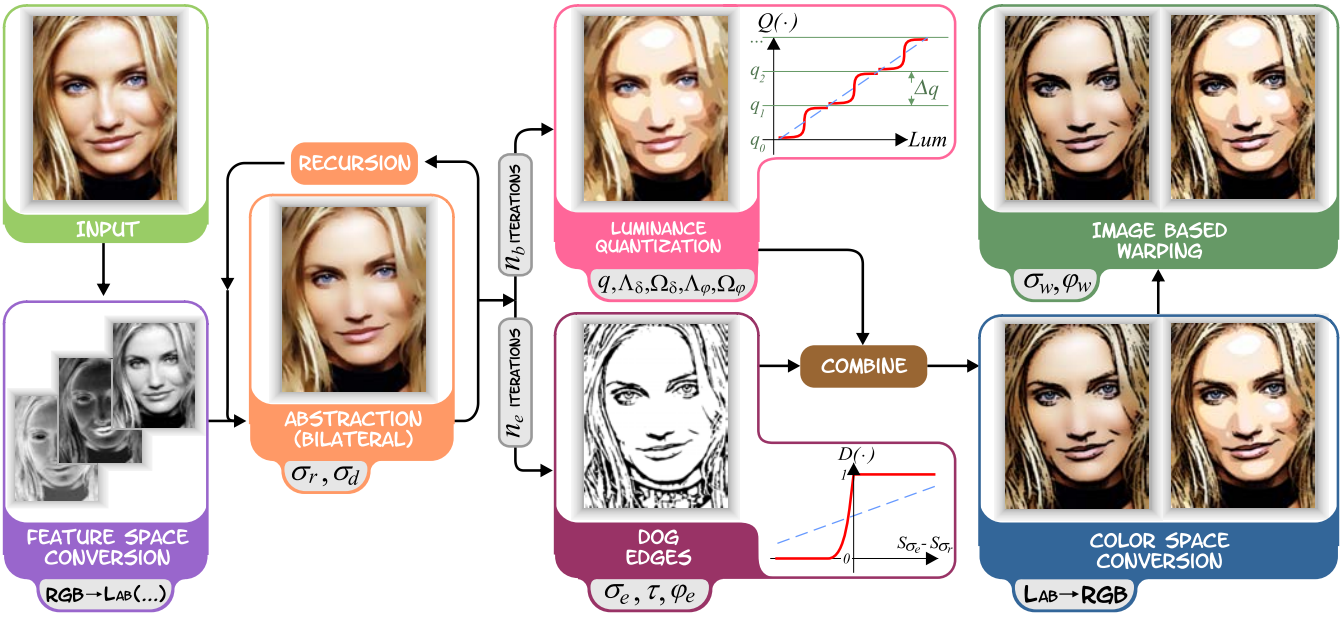


Figure 2: **Framework overview**. Each step lists the function performed, along with user parameters. The right-most paired images show alternative results, depending on whether luminance quantization is enabled (right) or not (left). The top image pair shows the final output.

mation from images, Raskar et al. [2004] computed ordinal depth from pictures taken with purpose-built multi-flash hardware. This allowed them to separate texture edges from depth edges and perform effective texture removal and other stylization effects. Our own framework does not model global effects such as repeated texture, but also requires no specialized hardware and does not face the technical difficulties of multi-flash for video.

Several video stylization systems have been proposed, mainly to help artists with labor-intensive procedures [Wang et al. 2004; Collomosse et al. 2005]. Such systems extended the mean-shift-based stylization approach of DeCarlo and Santella [2002] to computationally expensive three-dimensional video volumes. Difficulties with contour tracking required substantial user correction of the segmentation results, particularly in the presence of occlusions and camera movement. Our framework does not derive an explicit representation of image structure, thus limiting the types of stylization we can achieve. In turn, we gain a framework that is much faster to compute, fully automatic, and temporally coherent.

Fischer et al. [2005] explored the use of automatic stylization techniques in augmented reality applications. To make virtual objects less distinct from the live video stream, they applied stylization effects to both virtual and real inputs. Although parts of their system are similar to our own, their implementation is limited in the amount of detail it can resolve, and their stylized edges tend to suffer from temporal noise.

Recently, several authors of NPR systems have defined task-dependent objectives for their stylized imagery and tested these with perceptual user studies. DeCarlo and Santella [2002] use eye-tracking data to guide image simplification in a multi-scale system. In follow-up work, Santella and DeCarlo [2004] found that their eye-tracking-driven simplifications guided viewers to regions determined to be important. They also considered the use of computational salience as an alternative to measured salience. Our own work does not rely on eye-tracking data, although such data can be used. Our implicit visual salience model is less elaborate than the explicit model of Santella and DeCarlo's later work, but can be computed in real-time. Their explicit image structure representation allowed for more aggressive stylization, but included no provisions for the temporal coherence featured in our framework.

Gooch et al. [2004] automatically created monochromatic human facial illustrations from Difference-of-Gaussian (DoG) edges and a simple model of brightness perception. We use a similar edge model and evaluation study to Gooch et al. but additionally address color, real-time performance and temporal coherence.

3 Method

Our goal is to abstract images by simplifying their visual content while preserving or even emphasizing most of the perceptually important information.

Our framework is based on the assumptions that (1) the human visual system operates on different *features* of a scene, (2) changes in these features are of perceptual importance and therefore visually interesting (*salient*), and (3) polarizing these changes is a basic but useful method for automatic image abstraction.

Several image features are believed to play a vital role in low level human vision, among these are luminance, color opponency, and orientation [Palmer 1999]. A sudden spatial change (high contrast) in any of these features can represent boundaries of objects, subobject boundaries, or other perceptually important information. High contrast in these features is therefore linked to high visual salience, and low contrast to low salience. Based on this principle, several computational models of visual salience have been proposed [Privitera and Stark 2000; Itti and Koch 2001].

For our automatic, real-time implementation we implicitly compute visual salience with the following restrictions: we consider just two feature contrasts, luminance, and color opponency; we do not model effects requiring global integration; and we process images only within a small range of spatial scales. To allow for artistic control or more elaborate visual salience models, our framework alternatively accepts arbitrary scalar fields to direct abstraction.

The basic workflow of our framework is shown in Figure 2. We first exaggerate the given contrast in an image using nonlinear dif-

fusion. We then add highlighting edges to increase local contrast, and we optionally stylize and sharpen the resulting images.

3.1 Extended Nonlinear Diffusion

Perona and Malik [1991] defined a class of filters, called *anisotropic diffusion* filters, which have the desirable property of blurring small discontinuities *and* sharpening edges, as guided by a diffusion conduction function that varies over the image. Using such a filter with a conduction function based on feature contrast, we can amplify or subdue the given contrast in parts of an image. Barash and Comaniciu [2004] demonstrated that anisotropic diffusion solvers can be extended to larger neighborhoods, thus producing a broader class of *extended nonlinear diffusion* filters. This class includes iterated bilateral filters as one special case, which we prefer due to their larger support size and the fact that they can be approximated quickly and with few visual artifacts using a separated kernel [Pham and Vliet 2005].

Given an input image $f(\cdot)$, which maps pixel locations into some feature space, we define the following filter, $H(\cdot)$:

$$H(\hat{x},\sigma_d,\sigma_r) = \frac{\int e^{-\frac{1}{2}\left(\frac{\|\hat{x}-x\|}{\sigma_d}\right)^2} w(x,\hat{x}) f(x)\,dx}{\int e^{-\frac{1}{2}\left(\frac{\|\hat{x}-x\|}{\sigma_d}\right)^2} w(x,\hat{x})\,dx} \tag{1}$$

In this formulation, $\hat{x}$ is a pixel location, x are neighboring pixels, and σ_d is related to the blur radius. Increasing σ_d results in more blurring, but if σ_d is too large features may blur across significant boundaries. The range weighting function, $w(\cdot)$, determines where in the image contrasts are smoothed or sharpened by iterative applications of $H(\cdot)$.

$$w(x,\hat{x},\sigma_r) = (1-m(\hat{x}))\cdot w'(x,\hat{x},\sigma_r) + m(\hat{x})\cdot u(\hat{x}) \tag{2}$$

$$w'(x,\hat{x},\sigma_r) = e^{-\frac{1}{2}\left(\frac{\|f(\hat{x})-f(x)\|}{\sigma_r}\right)^2} \tag{3}$$

For the real-time, automatic case, we set $m(\cdot)=0$, such that $w(\cdot)=w'(\cdot)$ and Equation 1 becomes the familiar bilateral filter, where σ_r determines how contrasts will be preserved or blurred. Small values of σ_r preserve almost all contrasts, and thus lead to filters with little effect on the image, whereas for large values, $w'(\cdot) \xrightarrow{\sigma_r\to\infty} 1$, thus turning $H(\cdot)$ into a standard, linear Gaussian blur. For intermediate values of σ_r, iterative filtering of $H(\cdot)$ results in an extended nonlinear diffusion effect, where the degree of smoothing or sharpening is determined by local contrasts in $f(\cdot)$'s feature space. We use $\sigma_d=3$ throughout this paper and choose $\sigma_r=4.25$ for most images and the video.

With $m(\cdot)\neq 0$, the range weighting function, $w(\cdot)$, turns into a weighted sum of $w'(\cdot)$ and an arbitrary importance field, $u(\cdot)$, defined over the image. In this case, $m(\cdot)$ and $u(\cdot)$ can be computed via a more elaborate visual salience model [Itti and Koch 2001], derived from eye-tracking data (Figure 3, [DeCarlo and Santella 2002]), or painted by an artist [Hertzmann 2001].

Tomasi and Manduchi [1998] suggested computing the bilateral filter on a perceptually uniform feature space, such as *CIELab* [Wyszecki and Styles 1982], so that image contrast is adjusted depending on just noticeable differences. We follow this advice and our parameter values assume that $L\in[0,100]$ and $(a,b)\in[-127,127]$. Theoretically, the feature space could be extended to include additional features, such as orientation-dependent Gabor filters, although care would have to be taken to maintain perceptual uniformity of the combined feature space.

Figure 3: **Automatic vs. external abstraction**. *Top Row*: Original image by DeCarlo and Santella [2002] and their abstraction using eye-tracking data. *Bottom Row*: Our automatic abstraction, and data-driven abstraction based on the eye-tracking data.

3.2 Edge detection

In general, edges are defined by high local contrast, so adding visually distinct edges to regions of high contrast further increases the visual distinctiveness of these locations.

Marr and Hildreth [1980] formulated an edge detection mechanism based on zero-crossings of the second derivative of the luminance function. They postulated that retinal cells (*center*), which are stimulated while their *surrounding* cells are not stimulated, could act as neural implementations of this edge detector. A computationally simple approximation is the difference-of-Gaussians (DoG) operator. Rather than using a binary model of cell-activation, we define our DoG edges using a slightly smoothed step function, $D(\cdot)$ (bottom inset, Figure 2) to increase temporal coherence in animations. The parameter τ in Equation 4 controls the amount of center-surround difference required for cell activation, and φ_e controls the sharpness of the activation falloff. In the following, we define $S_{\sigma_e}\equiv S(\hat{x},\sigma_e)$ and $S_{\sigma_r}\equiv S(\hat{x},\sqrt{1.6}\cdot\sigma_e)$, with blur function $S(\cdot)$ given in Equation 5. The factor of 1.6 relates the typical receptive field of a cell to its surroundings [Marr and Hildreth 1980].

$$D(\hat{x},\sigma_e,\tau,\varphi_e) = \begin{cases} 1 & \text{if } (S_{\sigma_e}-\tau\cdot S_{\sigma_r})>0, \\ 1+\tanh(\varphi_e\cdot(S_{\sigma_e}-\tau\cdot S_{\sigma_r})) & \text{otherwise.}\end{cases} \tag{4}$$

$$S(\hat{x},\sigma_e) = \frac{1}{2\pi{\sigma_e}^2}\int f(x)c(\hat{x}-x,\sigma_e)\,dx \tag{5}$$

Here, σ_e determines the spatial scale for edge detection. The larger the value, the coarser the edges that are detected. The threshold level τ determines the sensitivity of the edge detector. For small values of τ, less noise is detected, but real edges become less prominent. As $\tau\to 1$, the filter becomes increasingly unstable. We use $\tau=0.98$ throughout. The falloff parameter, φ_e, determines the sharpness of edge representations, typically $\varphi_e\in[0.75,5.0]$. For n_b bilateral iterations, we extract edges after $n_e<n_b$ iterations to reduce noise. Typically, $n_e\in\{1,2\}$ and $n_b\in\{3,4\}$.

Canny [1986] devised a more sophisticated edge detection algorithm, which found use in several related works [DeCarlo and Santella 2002; Fischer et al. 2005]. Canny edges are guaranteed to lie on any real edge in an image, but can become disconnected for large values of σ_e and are computationally more expensive. DoG

Figure 4: **Parameter variations**. *Coarse*: Abstraction using coarse edges ($\sigma_e = 5$) and soft quantization steps ($q = 10, \Lambda_\varphi = 0.9, \Omega_\varphi = 1.6, \varphi_q = 3.1$). *Detailed*: Finer edges ($\sigma_e = 2$) and sharper quantization steps ($q = 14, \Lambda_\varphi = 3.4, \Omega_\varphi = 10.6, \varphi_q = 9.7$).

edges are cheaper to compute and not prone to disconnectedness but may drift from real image edges for large values of σ_e. We prefer DoG edges for computational efficiency and because their thickness scales naturally with σ_e.

Image-based warping (IBW) To fix small edge drifts linked to DoG edges and to sharpen the overall appearance of our final result we optionally perform an image-based warp (Figure 2, top-right). IBW is a technique first proposed by Arad and Gotsaman [1999] for image sharpening and edge-preserving expansion, in which they moved pixels along a warping field towards nearby edges. Loviscach [1999] proposed a simpler IBW implementation, in which the warping field is the blurred and scaled result of a Sobel filter of an input image. We use Loviscach's method with Gaussian blur $\sigma_w = 1.5$, and a scale factor of $\varphi_w = 2.7$.

3.3 Temporally coherent stylization

To open our framework further for creative use, we perform an optional color quantization step on the abstracted images, which results in cartoon or paint-like effects (Figures 1 and 4).

$$Q(\hat{x}, q, \varphi_q) \quad = \quad q_{nearest} + \frac{\Delta q}{2} \tanh(\varphi_q \cdot (f(\hat{x}) - q_{nearest})) \quad (6)$$

In Equation 6, $Q(\cdot)$ is the pseudo-quantized image, Δq is the bin width, $q_{nearest}$ is the bin boundary closest to $f(\hat{x})$, and φ_q is a parameter controlling the sharpness of the transition from one bin to another (top inset, Figure 2). Equation 6 is formally a discontinuous function, but for sufficiently large φ_q, these discontinuities are not noticeable.

For a fixed φ_q the transition sharpness is independent of the underlying image, possibly creating many noticeable transitions in large smooth-shaded regions. To minimize jarring transitions, we define the sharpness parameter, φ_q, to be a function of the luminance gradient in the abstracted image. We allow hard bin boundaries only where the luminance gradient is high. In low gradient regions, bin boundaries are spread out over a larger area. We thus offer the user a trade-off between reduced color variation and increased quantization artifacts by defining a target sharpness range $[\Lambda_\varphi, \Omega_\varphi]$ and a gradient range $[\Lambda_\delta, \Omega_\delta]$. We clamp the calculated gradients to $[\Lambda_\delta, \Omega_\delta]$ and then generate a φ_q value by mapping them linearly to $[\Lambda_\varphi, \Omega_\varphi]$. The effect for typical parameter values are hard, cartoon-like boundaries in high gradient regions and soft, painterly-like transitions in low gradient regions (Figure 4). Typical values for these parameters are $q \in [8, 10]$ equal-sized bins and a gradient range of $[\Lambda_\delta = 0, \Omega_\delta = 2]$, mapped to sharpness values between $[\Lambda_\varphi = 3, \Omega_\varphi = 14]$.

Another significant advantage of our pseudo-quantization implementation is temporal coherence. In standard quantization, an arbitrarily small luminance change can push a value to a different bin, thus causing a large output change for a small input change, which is particularly troublesome for noisy input. With soft quantization, such a change is spread over a larger area, making it less noticeable. Using our gradient-based sharpness control, sudden changes are further subdued in low-contrast regions, where they would be most objectionable.

Figure 5: **Sample images from evaluation studies.** The top row shows the original images and the bottom row shows the abstracted versions. All images use the same σ_e for edges and the same number of simplification steps, n_b. *Left*: Faces similar to those in Study 1. *Right*: Sample images from Study 2.

4 Evaluation

To verify that our abstracted images preserve or even distill perceptually important information, we performed two task-based studies to test recognition speed and short term memory retention. Our studies use small images because we see portable visual communication and low-bandwidth applications to practically benefit most from our framework and because small images may be a more telling test of our framework, as each pixel represents a larger percentage of the image.

Participants In each study, 10 (5 male, 5 female) undergraduates, graduate students or research staff acted as volunteers.

Materials Images in Study 1 are scaled to 176×220, while those in Study 2 are scaled to 152×170. These resolutions approximate those of many portable devices. Images are shown on a 30-inch Apple Cinema Display at a distance of 24 inches. The background of the monitor is set to white and the displayed images subtend a visual angle of 6.5 and 6.0 degrees respectively.

In Study 1, 50 images depicting the faces of 25 famous movie stars are used as visual stimuli. Each face is depicted as a color photograph and as a color abstracted image created with our framework. Five independent judges rated each pair of photograph and abstracted image as good likenesses of the face they portrayed. In Study 2, 32 images depicting arbitrary scenes are used as visual stimuli. Humans are a component in 16 of these images. Examples of stimulus images are shown in Figure 5.

Analysis For both studies, p-values are computed using two-way analysis of variance (ANOVA), with $\alpha = 0.05$.

4.1 Study 1: *Recognition Speed*

Study 1 assesses the recognition time of familiar faces presented as abstract images and photographs. The study uses a protocol [Stevenage 1995] demonstrated to be useful in the evaluation of recognition times for facial images [Gooch et al. 2004].

Procedure Study 1 consists of two phases: (1) reading the list of 25 movie star names out loud, and (2) a reaction time task in which participants are presented with sequences of the 25 facial images. All faces take up approximately the same space in the images and are three quarter views. By pronouncing the names of the people that are rated, participants tend to reduce the *tip-of-the-tongue* effect where a face is recognized without being able to quickly recall the associated name [Stevenage 1995]. For the same reason, participants are told that first, last or both names can be given, whichever is easiest. Each participant is asked to say the name of the person pictured as soon as that person's face is recognized. A study coordinator records reaction times, as well as accuracy of the answers. Images are shown and reaction times recorded using the *Superlab* software product for 5 seconds at 5-second intervals. The order of image presentation is randomized for each participant.

Results and Discussion In our study, participants are faster ($p < 0.018$) at naming abstract images ($M = 1.32$s) compared to photographs ($M = 1.51$s). The accuracy for recognizing abstract images and photographs are 97% and 99% respectively, indicating that there is no significant speed for accuracy trade-off. It can further be concluded that substituting abstract images for fully detailed photographs reduces recognition latency by 13%, a significant improvement not found by Sevenage [1995] and Gooch et al. [2004]. However, neither author used color images as stimuli.

4.2 Study 2: *Memory Game*

Study 2 assesses memory retention for abstract images versus photographs with a memory game, consisting of a grid of 24 randomly sorted cards placed face-down. The goal is to create a match by turning over two identical cards. If a match is made, the matched cards are removed. Otherwise, the cards are placed face down and another set of cards are turned over. The game ends when all pairs are matched. We created a Java program of the card game in which a user turns over a virtual card with a mouse click. The 12 images used in any given memory game are randomly chosen from the pool of 32 images without replacement, and randomly arranged. The program records the time it takes to complete a game and the number of cards turned over.

Procedure Study 2 consists of three phases: (1) a practice memory game with alphabet cards, (2) a memory game of photographs, and (3) a memory game of abstract images. All participants first play a practice game with alphabet cards to learn the interface and to develop a game strategy. No data is recorded for the practice phase. For the remaining two phases, half the participants are presented with photographs followed by abstracted images, and the other half is presented with abstracted images followed by photographs.

Results and Discussion In our study, participants are quicker ($p_{time} < 0.003$, $p_{clicks} < 0.004$) in completing a memory game using abstract images ($M_{time} = 59.95$ s, $M_{clicks} = 49.2$) compared to photographs ($M_{time} = 76.13$ s, $M_{clicks} = 62.4$). The study demonstrates that participants play the abstracted image version of the game faster than the version using photographs. In addition, using the abstracted images requires fewer cards to be turned over, possibly indicating that it is easier to remember previously revealed

Figure 6: **Failure case.** A case where our contrast-based importance assumption fails. *Left*: The subject of this photograph has very low contrast compared with its background. *Right*: The cat's low contrast fur is abstracted away, while the detail in the structured carpet is further emphasized. Despite this rare reversal of contrast assignment, the cat is still well represented.

abstractions. We thus conclude that the automatic image abstraction of our framework may produce more distinctive imagery.

5 Discussion and Conclusion

Performance We implemented and tested our framework in both a GPU-based real-time version, using OpenGL and fragment programs, and a CPU-version using OpenCV. Both versions were tested on an Athlon 64 3200+ with Windows XP and a GeForce GT 6800. Performance values depend on graphics drivers, image size, and framework parameters. Typical values for a 640×480 video stream and default parameters are $9-15$ frames per second (FPS) for the GPU version and $0.3-0.5$ FPS for the CPU version.

Limitations Our framework depends on local contrast to estimate visual salience. Images with very low contrast likely abstract too much and loose significant detail. Simply increasing contrast of the original image may reduce this problem, but can also increase noise. Figure 6 demonstrates an inversion of our general assumption, where important foreground objects have low contrast while background regions have high contrast. In practice we have obtained good results for many indoor and outdoor scenes.

Human vision operates at various spatial scales simultaneously. By applying multiple iterations of a non-linear blurring filter we cover a small range of spatial scales, but the range is not explicitly parameterized and not as extensive as that of real human vision.

Several high-contrast features that may be emphasized by our framework are actually deemphasized in human vision, among these specular highlights and repeated texture. Dealing with these phenomena using existing techniques requires global image processing, which is impractical in real-time on today's GPUs, due to their limited gather-operation capabilities.

Our fixed equidistant quantization boundaries are arbitrary, making it difficult to control results for artistic purposes. Constructing spatially varying boundaries to better account for underlying dynamic range might prove beneficial.

Compression A discussion of theoretical data compression and codecs exceeds the scope of the paper, but Pham and Vliet [2005] have shown that video compresses better when bilaterally filtered, judged by RMS error and MPEG quality score. Collomosse et al. [2005] list theoretical compression results for vectorized cartoon images. Possibly most applicable to this paper is work by Elder [1999], who describes a method to store the color information of an image only in high-contrast regions, achieving impressive compression results.

Indication *Indication* is the process of representing a repeated texture with a small number of exemplary patches and relying on

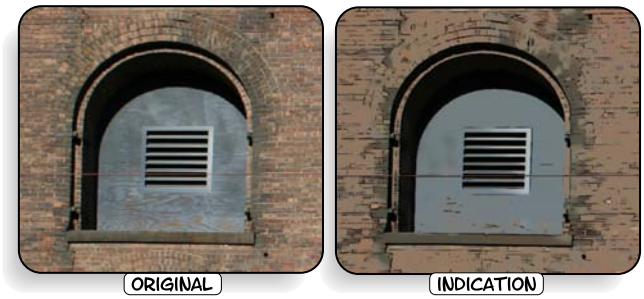


Figure 7: **Automatic indication**. The inhomogeneous texture of the bricks causes spatially varying abstraction. The resulting edges *indicate* a brick texture instead of depicting each individual brick.

an observer to interpolate between patches. For structurally simple, slightly inhomogeneous textures with limited scale variation, like the brick wall in Figure 7, our framework can perform simple automatic indication. As noted by DeCarlo and Santella [2002], such simple indication does not deal well with complex or foreshortened textures. Our automatic indication is not as effective as the user-drawn indications of Winkenbach and Salesin [1994], but some user guidance can be supplied via Equation 2.

Conclusion We have presented a simple and effective real-time framework that abstracts images while retaining much of their perceptually important information, as demonstrated in our user study. Our optional stylization step is temporally highly stable, results in effective color flattening and is much faster than the mean-shift procedures used in offline cartoon stylization for video [Collomosse et al. 2005; Wang et al. 2004]. Interestingly, several authors [Barash and Comaniciu 2004; Boomgaard and de Weijer 2002] have shown that anisotropic diffusion filters are closely related to the mean-shift algorithm. It is thus conceivable that various graphics applications that today rely on mean-shift could benefit from the much speedier anisotropic diffusion pre-process used in this paper.

Acknowledgements Many thanks to Amy Gooch, David Feng and our anonymous reviewers for their helpful writing suggestions; Jack Tumblin for inspiring discussions; Pin Ren for photographic assistance; Tom Lechner for modeling; the Northwestern GFX Group for their support; Rosalee Wolfe and Karen Alkoby for the deaf signing video; Douglas DeCarlo and Anthony Santella for proof-reading and supplying Figure 3 (top) and eye-tracking data; Marcy Morris and James Bass for acquiring image permission from Ms. Diaz. This material is based upon work supported by the National Science Foundation under Grant No. 0415083. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors. Figure 1 (left) and Figure 4 by Martina Winnemöller. Figure 2, Cameron Diaz, with permission. Figure 3 (top) by Anthony Santella, with permission of ACM. Figure 5, celebrities by Rita Molnár, Creative Commons License. Figure 7 by Hans Beushausen.

References

ARAD, N., AND GOTSMAN, C. 1999. Enhancement by image-dependent warping. *IEEE Trans. on Image Processing 8*, 9, 1063–1074.

BARASH, D., AND COMANICIU, D. 2004. A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift. *Image and Video Computing 22*, 1, 73–81.

BOOMGAARD, R. V. D., AND DE WEIJER, J. V. 2002. On the equivalence of local-mode finding, robust estimation and mean-shift analysis as used in early vision tasks. *16th Internat. Conf. on Pattern Recog. 3*, 927–390.

CANNY, J. F. 1986. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence 8*, 769–798.

COLLOMOSSE, J. P., ROWNTREE, D., AND HALL, P. M. 2005. Stroke surfaces: Temporally coherent artistic animations from video. *IEEE Trans. on Visualization and Computer Graphics 11*, 5, 540–549.

DECARLO, D., AND SANTELLA, A. 2002. Stylization and abstraction of photographs. *ACM Trans. Graph. 21*, 3, 769–776.

ELDER, J. H. 1999. Are edges incomplete? *Internat. Journal of Computer Vision 34*, 2-3, 97–122.

FISCHER, J., BARTZ, D., AND STRASSER, W. 2005. Stylized Augmented Reality for Improved Immersion. In *Proc. of IEEE VR*, 195–202.

GOOCH, B., REINHARD, E., AND GOOCH, A. 2004. Human facial illustrations: Creation and psychophysical evaluation. *ACM Trans. Graph. 23*, 1, 27–44.

HERTZMANN, A. 2001. Paint by relaxation. In *CGI '01:Computer Graphics Internat. 2001*, 47–54.

ITTI, L., AND KOCH, C. 2001. Computational modeling of visual attention. *Nature Reviews Neuroscience 2*, 3, 194–203.

LOVISCACH, J. 1999. Scharfzeichner: Klare bilddetails durch verformung. *Computer Technik 22*, 236ff.

MARR, D., AND HILDRETH, E. C. 1980. Theory of edge detection. *Proc. Royal Soc. London, Bio. Sci. 207*, 187–217.

PALMER, S. E. 1999. *Vision Science: Photons to Phenomenology*. The MIT Press.

PERONA, P., AND MALIK, J. 1991. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. on Pattern Analysis and Machine Intelligence 12*, 7.

PHAM, T. Q., AND VLIET, L. J. V. 2005. Separable bilateral filtering for fast video preprocessing. In *IEEE Internat. Conf. on Multimedia & Expo*, CD1–4.

PRIVITERA, C. M., AND STARK, L. W. 2000. Algorithms for defining visual regions-of-interest: Comparison with eye fixations. *IEEE Trans. on Pattern Analysis and Machine Intelligence 22*, 9, 970–982.

RASKAR, R., TAN, K.-H., FERIS, R., YU, J., AND TURK, M. 2004. Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging. *ACM Trans. Graph. 23*, 3, 679–688.

SAITO, T., AND TAKAHASHI, T. 1990. Comprehensible rendering of 3-D shapes. In *Proc. of ACM SIGGRAPH 90*, 197–206.

SANTELLA, A., AND DECARLO, D. 2004. Visual interest and NPR: an evaluation and manifesto. In *Proc. of NPAR '04*, 71–78.

STEVENAGE, S. V. 1995. Can caricatures really produce distinctiveness effects? *British Journal of Psychology 86*, 127–146.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proceedings of ICCV '98*, 839.

WANG, J., XU, Y., SHUM, H.-Y., AND COHEN, M. F. 2004. Video tooning. *ACM Trans. Graph. 23*, 3, 574–583.

WINKENBACH, G., AND SALESIN, D. H. 1994. Computer-generated pen-and-ink illustration. In *Proc. of ACM SIGGRAPH 94*, 91–100.

WYSZECKI, G., AND STYLES, W. 1982. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley, New York, NY.

Two-scale Tone Management for Photographic Look

Soonmin Bae Sylvain Paris Frédo Durand

Computer Science and Artificial Intelligence Laboratory

Massuchusetts Institute of Technology

(a) input (b) sample possible renditions: bright and sharp, gray and highly detailed, and contrasted, smooth and grainy

Figure 1: This paper describes a technique to enhance photographs. We equip the user with powerful filters that control several aspects of an image such as its tonal balance and its texture. We make it possible for anyone to explore various renditions of a scene in a few clicks. We provide an effective approach to æsthetic choices, easing the creation of compelling pictures.

Abstract

We introduce a new approach to tone management for photographs. Whereas traditional tone-mapping operators target a neutral and faithful rendition of the input image, we explore pictorial looks by controlling visual qualities such as the tonal balance and the amount of detail. Our method is based on a two-scale non-linear decomposition of an image. We modify the different layers based on their histograms and introduce a technique that controls the spatial variation of detail. We introduce a Poisson correction that prevents potential gradient reversal and preserves detail. In addition to directly controlling the parameters, the user can transfer the look of a model photograph to the picture being edited.

Keywords: Computational photography, high dynamic range, tone management, pictorial look, bilateral filter, image processing

1 Introduction

Much research has been dedicated to tone mapping for the display of high-dynamic-range images. These tools focus on contrast reduction, seeking a neutral reproduction, and are ideal when fidelity is needed. However, tone manipulation is also useful when the input has normal dynamic range, and many users seek to obtain a certain "look" for their pictures to convey a mood or an æsthetic. This is particularly significant for black-and-white photography where strikingly distinctive styles can be achieved. We present a new tone management approach that offers direct control over the "look" of an image for both high- and normal-dynamic-range inputs.

The "look" of images has been addressed in Non-Photorealistic Rendering and recent analogy approaches enable the imitation of texture or stylized images in a purely data-driven fashion, e.g. [Hertzmann et al. 2001]. However, to the best of our knowledge, no approach enables the imitation of a photographic "look" such as the ones achieved by master black-and-white photographers.

We argue that a large part of such a look deals with the management of tones, which advanced photographers perform through elaborate lighting, tedious work in the darkroom, or using photo editing software. Unfortunately, such painstaking work and advanced skill is out of reach of casual users. In addition, the issues of *workflow* and efficiency are becoming prevalent among professional users. The workflow describes the full process from image capture to printing and can include multiple software stages and manual retouching, all requiring much effort. Reducing the user work is critical to professionals, and many manuals and tools are dedicated to optimizing and automating all steps. For example, a wedding photographer takes hundreds of pictures and needs to give them a consistent look and to swiftly deliver them to clients. Individual retouching is hardly an option, and creative control over the look of each image is often sacrificed. Recently-introduced software such as Apple's Aperture and Adobe's Lightroom focuses on workflow optimization but offers little interactive editing capabilities.

To address these difficulties, we propose a tone-management technique dedicated to both casual and professional photographers. We focus on the tonal aspects of photos decoupled from their content. Issues such as framing and topic selection are out of the scope of our work. We nevertheless demonstrate the wide range of looks that our approach can produce. We provide simple controls and enable both global and local tone management. In addition to direct manipulation, users can transfer the look of a model picture,

(a) Clearing Winter Storm Ansel Adams,1942 or earlier
(reproduced with permission)

(b) Angkor #71, by Kenro Izu
(Original: 14"x20" film contact printed on Platinum/Palladium coated paper.)
Copyright 1994 Kenro Izu, reproduced with permission of the artist.

Figure 2: Typical model photographs that we use. Photo (a) exhibits strong contrast with rich blacks, and large textured areas. Photo (b) has mid-tones and vivid texture over the entire image.

thereby "showing" the desired look. This also allows professionals to apply the rendition of previous prints to new photographs.

This paper makes the following contributions.
Large-scale Tonal Balance Management: We control the large-scale spatial tonal variation over an image.
Spatial Detail Variation: We manipulate the amount of high-frequency detail or texture and its spatial variation. In particular, we introduce a computation of *textureness* that measures local high-frequency content while respecting strong edges.
Gradient Constraint: We employ a gradient reconstruction step to prevent gradient reversal and preserve detail.

Our exposition focuses on transfer between images because it demonstrates the relevance and robustness of the features we manipulate. However, direct control through the curve interface is equally powerful, though perhaps more suited to advanced users.

1.1 Related work

Tone Mapping Tone-mapping seeks the faithful reproduction of high-dynamic-range images on low-dynamic-range displays, while preserving visually important features [Reinhard et al. 2005]. Our work builds on local tone mapping where the mapping varies according to the neighborhood of a pixel [Pattanaik et al. 1998; Tumblin and Turk 1999; Reinhard et al. 2002; Durand and Dorsey 2002; Fattal et al. 2002; Li et al. 2005]. The precise characteristics of film have also been reproduced [Geigel and Musgrave 1997; Reinhard et al. 2002]. However, most techniques seek an objective rendering of the input, while we want to facilitate the exploration and transfer of particular pictorial looks.

Conversion to Grayscale Gooch et al. [2005] convert color images to grayscale while preserving salient features. They also seek fidelity to the original picture, whereas we explore stylistic variations. Their approach is nonetheless complementary to ours because it extracts compelling contrast from color images.

Gradient Image Processing A number of recent techniques have characterized images by their gradient and used Poisson reconstruction to perform tone mapping [Fattal et al. 2002] and montages [Pérez et al. 2003; Agarwala et al. 2004]. We also exploit the Poisson approach to ensure the quality of our result, because it naturally allows us to combat gradient reversal, a traditional plague of aggressive multi-scale manipulation.

Style Transfer and Stylization Style transfer has been explored for the textural aspects of non-photorealistic media, e.g. [Hertzmann et al. 2001; Drori et al. 2003], and DeCarlo et al. stylize photographs based on saliency [2002]. In contrast, we seek to retain photorealism and control large-scale effects such as tonal balance and the variation of local detail. In addition, our parametric approach leads to continuous changes supported by interactive feedback and enables interpolations and extrapolations of image look.

Visual Equalizer Our work is inspired by the ubiquitous visual equalizer of sound devices. Similarly, the modification of frequency bands can alter the "mood" or "style" of motion data [Bruderlin and Williams 1995]. The equivalent for images is challenging because of the halos that frequency decomposition can generate around edges. Our work can be seen as a two-band equalizer for images that uses non-linear signal processing to avoid halos and provides fine tonal and spatial control over each band.

1.2 Achieving a Photographic Look

The traditional darkroom offers remarkable global and local control over the brightness, contrast, and sharpness of images via a combination of chemical and optical processes [Rudman 1994; Adams 1995]. Black-and-white photographs vary in their tonal palette and how they deal with the dynamic range of a scene. A photographer like Adams (Fig. 2a) exhibits strong contrast with rich blacks, while an artist like Stieglitz (Fig. 15a) relies more on the mid-tones. This suggests the intensity histogram as a characterization of tonal look, but we show in this paper that the spatial distribution of tones must be taken into account because a histogram does not make the distinction between local and global contrast.

The amount of texture is crucial in photographs; some artists use vivid texture over the entire image (Fig. 2b), while other contrast large smooth areas with strong textures in other parts of the image (Fig. 2a). Furthermore, the human visual system is known to be more sensitive to local contrast than to low spatial frequencies.

Finally, a photograph is characterized by low-level aspects of the medium such as tone (e.g. sepia toning) and grain (controlled by the film and paper characteristics).

These observations drive our approach. We propose decompositions of an image that afford direct control over dynamic range, tonal distribution, texture and sharpness.

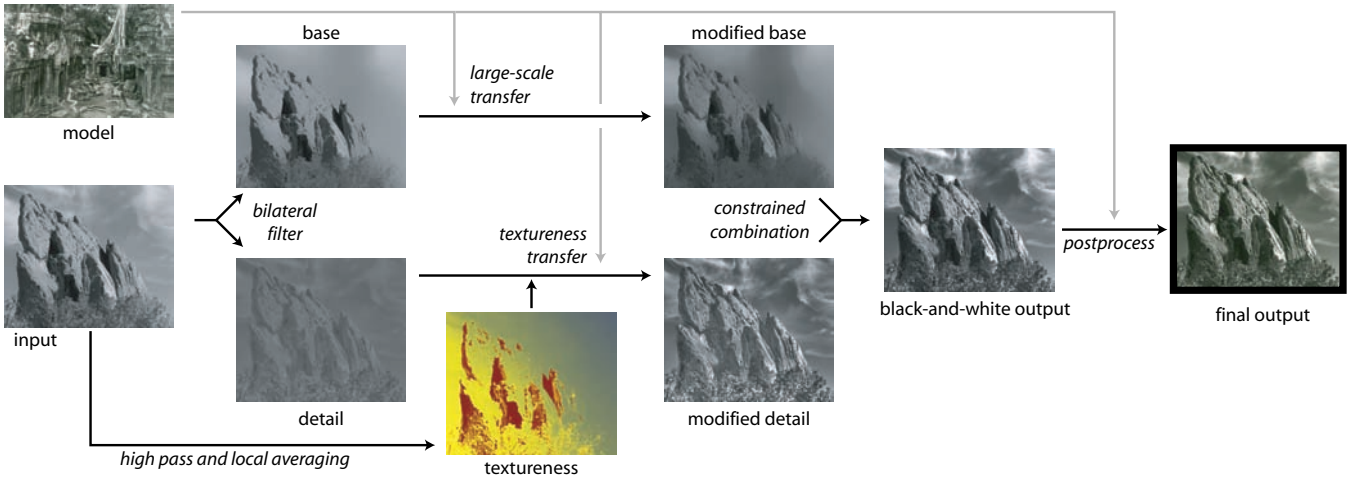


Figure 3: Overview of our pipeline. The input image is first split into base and detail layers using bilateral filtering. We use these layers to enforce statistics on low and high frequencies. To evaluate the texture degree of the image, we introduce the notion of *textureness*. The layers are then recombined and post-processed to produce the final output. The model is Kenro Izu's masterpiece shown in Figure 2b.

1.3 Overview

The previous discussion suggests that aspects such as the intensity distribution at different scales, spatial variations, and the amount and distribution of detail are critical to the look of a photograph. This inspires our use of a two-scale decomposition to control large-scale effects and the texture distribution. We quantify the look of an image using histograms over this decomposition, which affords both interactive control using a curve interface, and the ability to automatically transfer visual properties between images. In the latter, histograms of the components of a model image are forced upon a new input. Because we explore strong stylistic variations, we tend to perform larger modifications to the input than tone mapping. In particular, some looks require an increase in local contrast, which can produce halos if traditional techniques are used. We introduce a gradient constraint that prevents undesirable modifications. Finally, we post-process the image to achieve various effects such as soft focus, paper grain, and toning. Figure 3 summarizes this process.

2 Background

Before introducing our approach, we review important image-processing tools at the core of our technique.

Histogram Matching Matching histograms is the traditional solution to transferring an intensity distribution. Given an image I with histogram h_I and a reference histogram h_M, we seek a function $\ell_{I\triangleright M}$ such that applying $\ell_{I\triangleright M}$ to each pixel of I results in an image with histogram h_M. To build $\ell_{I\triangleright M}$, we use the cumulative histograms c_M and c_I defined by $c(x) = \int_{-\infty}^{x} h$. It can be shown that $c_I(I)$ produces an image with a uniform histogram. Thus, we define:

$$\ell_{I\triangleright M}(x) \;=\; c_M^{-1}[\,c_I(x)\,] \tag{1}$$

and $\ell_{I\triangleright M}(I)$ generates an image with the histogram h_M. More details can be found in image processing books, e.g. [Gonzales and Woods 2002] (p. 94). While histogram matching is a key tool in our approach, we observe that matching the pixel histogram is not sufficient to control the tonal look of an image (Fig. 11, 15 and 16).

Poisson Reconstruction Given a 2D field of 2D vectors $\mathbf{v}$, one can build an image I with a gradient ∇I as close as possible to $\mathbf{v}$, in the least square sense. This is achieved through a Poisson equation:

$$\partial I/\partial t = \Delta I - \operatorname{div}(\mathbf{v}) \tag{2}$$

Pérez *et al.* [2003] have shown impressive image manipulations using this tool. We refer to their paper for detail.

Bilateral Filtering The bilateral filter [Tomasi and Manduchi 1998] smooths the input image while preserving its main edges. Each pixel is a weighted mean of its neighbors where the weights decrease with the distance in space and with the intensity difference. With $g_\sigma(x) = \exp(-x^2/\sigma^2)$, a Gaussian function, the bilateral filter of image I at pixel $\mathbf{p}$ is defined by:

$$bf(I)_\mathbf{p} \;=\; \frac{1}{k}\sum_{\mathbf{q}\in I} g_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)\, g_{\sigma_\mathrm{r}}\left(|I_\mathbf{p}-I_\mathbf{q}|\right)\, I_\mathbf{q} \tag{3a}$$

$$\text{with:}\quad k \;=\; \sum_{\mathbf{q}\in I} g_{\sigma_\mathrm{s}}(\|\mathbf{p}-\mathbf{q}\|)\, g_{\sigma_\mathrm{r}}\left(|I_\mathbf{p}-I_\mathbf{q}|\right) \tag{3b}$$

where σ_s controls the spatial neighborhood, and σ_r the influence of the intensity difference, and k normalizes the weights. The bilateral filter is often used to create a two-scale decomposition where the output of the filter produces a large-scale layer (a.k.a. *base*) and the difference is called the *detail* layer [Durand and Dorsey 2002]. We use our fast version of the bilateral filter [Paris and Durand 2006].

3 Large-Scale Tonal Distribution

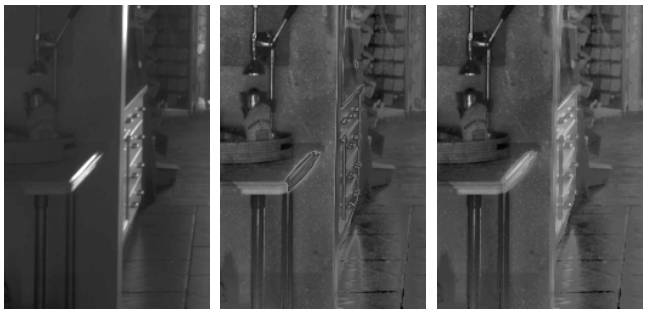
Our tone management relies on a two-scale decomposition based on the bilateral filter. We refine the standard usage of the bilateral filter in two ways: we introduce a gradient correction to prevent gradient reversals, and we apply histogram transformations instead of just scaling the large-scale component as in traditional tone mapping.

3.1 Bilateral Decomposition

We use a decomposition similar to that of Durand and Dorsey [2002]. Since contrast is a multiplicative effect, we work in the logarithmic domain. We define the base layer B and detail layer D from the input image I (where I, B and D have log values):

$$B \;=\; bf(I) \qquad \text{and} \qquad D \;=\; I \;-\; B \tag{4}$$

The choice of σ_s and σ_r is crucial. σ_s specifies spatial scales and $\sigma_\mathrm{s} = \min(\text{width}, \text{height})/16$ consistently produces good results. σ_r differentiates important edges from detail. We rely on the gradient



(a) input image (b) uncorrected detail (c) corrected detail

Figure 4: The bilateral filter can cause gradient reversals in the detail layer near smooth edges. Note the problems in the highlights (b). We force the detail gradient to have the same orientation as the input (c). Contrast is increased in (b) and (c) for clarity.

norm to estimate the edge amplitude in the input. With p_{90} denoting the 90th percentile[1], $\sigma_r = p_{90}(\|\nabla I\|)$ achieves consistently good results. These settings are robust to spatial and intensity scales.

Gradient Reversal Removal Durand and Dorsey [2002] note that artifacts can occur when edges are not sharp. They introduce a "fix" that detects uncertain pixels and uses a smoothed base layer, but they highlight that this solution is not entirely satisfying. The problem is more acute in our case because we may *increase* the amount of detail (by a factor as high as 6 in some examples), which requires a reliable halo-free detail layer.

We address this by directly constraining the gradient of the decomposition to prevent reversal. We force the detail derivatives $\partial D/\partial x$ and $\partial D/\partial y$ to have the same sign as the input derivatives and an amplitude no greater than them. For this, we build a gradient field $\mathbf{v} = (x_\mathbf{v}, y_\mathbf{v})$:

$$x_\mathbf{v} = \begin{cases} 0 & \text{if} \quad \text{sign}(\partial D/\partial x) \neq \text{sign}(\partial I/\partial x) \\ \partial I/\partial x & \text{if} \quad |\partial D/\partial x| > |\partial I/\partial x| \\ \partial D/\partial x & \text{otherwise} \end{cases} \tag{5}$$

The y component $y_\mathbf{v}$ is defined similarly. The corrected detail layer is obtained by solving the corresponding Poisson equation (Eq. 2). We update the base layer accordingly: $B = I - D$. This approach results in a high-quality detail layer because it directly addresses gradient reversal and preserves other subtle variations (Fig. 4).

3.2 Tonal Balance

The base layer contains the large-scale spatial distribution of tones (Fig. 5). In contrast to tone mapping where the base layer is simply scaled down [Durand and Dorsey 2002], we want to enforce a large-scale distribution of tones that matches a model image. This is why we perform histogram matching and transfer the histogram of the model base B_M onto the new base B_I.

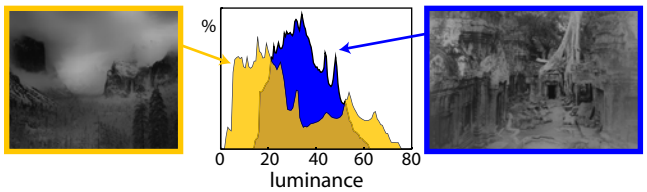


Figure 5: The luminance histogram of the base component is a good indicator of tonal balance. The photos are the same as in Figure 2.

[1]For an image I, $p_n(I)$ is the intensity value such that n% of the values of I are under it, *e.g.* $p_{50}(I)$ is the median. Percentiles are robust to outliers.

4 Detail and Texture Management

The amount and spatial distribution of high-frequency texture is the natural complement of the large-scale tonal palette. The core contribution of our work is a technique that manipulates the amount of high-frequency content and its spatial variation. This contrasts with tone mapping approaches that usually do not modify the detail layer.

This step involves additional challenges compared to the base transform. First, we show that the detail layer does not capture all the high frequency content of the image. Second, we need to modify the spatial variation of detail without creating artifacts. In particular, we introduce a new technique to measure and modify local frequency content in an edge-preserving manner.

4.1 Detail Management based on Frequency Analysis

While the bilateral filter provides a decomposition that facilitates halo-free manipulation, the edge-preserving term g_{σ_r} results in substantial high-frequency content in the base layer (Fig. 6). While the choice of different parameters or more advanced filters [Choudhury and Tumblin 2003] can affect this issue, the very nature of such filter calls for high-frequency content in the base. In particular, the influence of the range Gaussian g_{σ_r} means that patterns that are high-frequency but high-contrast will mostly be in the base. While this is not an issue for tone mapping where the detail is unaffected, it is critical for our detail management. On the other hand, the manipulation of the detail layer is a safe operation that does not lead to the halo artifacts caused by linear image processing.

Our solution combines linear frequency *analysis* with the *manipulation* of the detail layer obtained from our nonlinear filter. We analyze the amount of texture (or high frequency) using a high pass filter applied to *both* the detail and the base layer. This ensures that all the frequency content is taken into account. We use this information to decide how the detail layer should be modified. In a nutshell, we get the best of the two approaches: reliable analysis of the high-pass filter, and the safe manipulation of the detail layer.

4.2 Textureness

We seek to characterize the local amount of high frequency content over the image to distinguish regions with high levels of detail from

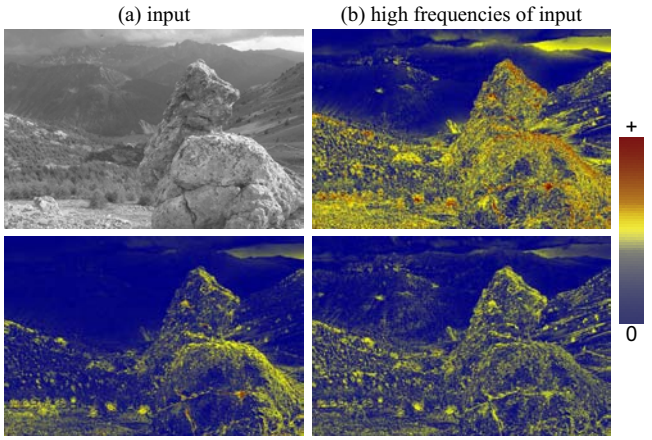


Figure 6: Because of the preserved edges, the high frequencies of an image (b) appear both in the base layer (c) and in the detail layer (d). This phenomenon has to be taken into account to achieve an appropriate analysis.

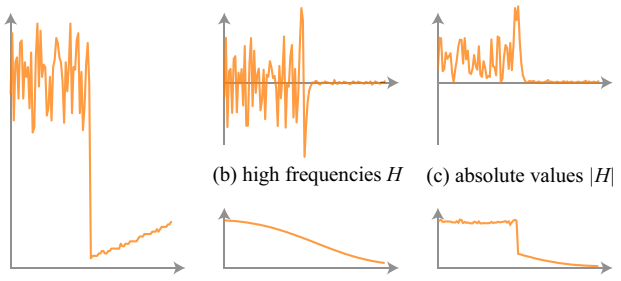


Figure 7: Textureness of a 1D signal. To estimate the textureness of the input (a), we compute the high frequencies (b) and their absolute values (c) . Finally, we locally average these amplitudes: Previous work based on low-pass filter (d) incurs halos (Fig. 8) whereas our cross bilateral filtering yields almost no halos (e) .

smooth regions. We build on the notion of power maps, e.g. [Su et al. 2005] and activity map [Li et al. 2005] where the local average of the amplitude of high frequencies is used. Figure 7 illustrates our computation of textureness for a 1D example where the left part has a high level of local contrast while the right part is smooth.

First, we compute a high-pass version H of the image using the same cutoff σ_s. Note that the local average of such a high-pass image is by definition zero: the low frequencies are removed. This is why we consider the *magnitude* (or absolute value) of H (Fig. 7c). Power maps or activity maps are then defined as the local average – obtained via low-pass filtering – of this magnitude (Fig. 7d). Such maps provide good characterization of highly-textured vs. smooth regions and the local level of detail can be altered by modifying the detail layer accordingly.

Unfortunately, such spatially-varying manipulation of detail can lead to artifacts at the boundary between highly detailed and smooth regions (Fig. 8). This is because the amount of detail on one side of the boundary influences the estimate on the other side, and the manipulation suffers from a halo effect similar to that observed in linear frequency decomposition of image intensity. This problem is the same as the one addressed by edge-preserving decomposition, except that we are dealing with a less spatially localized quantity, the magnitude of high frequency $|H|$. Strong edges are hard to characterize in $|H|$, which is why we define textureness using a *cross-bilateral filter* [Eisemann and Durand 2004; Petschnigg et al. 2004] where the intensity image defines the edge-preserving term to filter $|H|$. More precisely, our textureness is defined as

$$T(I)_{\mathbf{p}} = \frac{1}{k} \sum_{\mathbf{q} \in |H|} g_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, g_{\sigma_r}\left(|I_{\mathbf{p}} - I_{\mathbf{q}}|\right)\, |H|_{\mathbf{q}} \quad (6a)$$

$$\text{with:} \quad k = \sum_{\mathbf{q} \in I} g_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, g_{\sigma_r}\left(|I_{\mathbf{p}} - I_{\mathbf{q}}|\right) \quad (6b)$$

We set this cross filter with the same σ_r as for the base-detail computation, but with a larger σ_s (8 times larger in practice) to ensure smooth textureness variations on uniform regions (discontinuities can still happen at edges). Figure 9 shows how our textureness map captures the local amount of detail over the image.

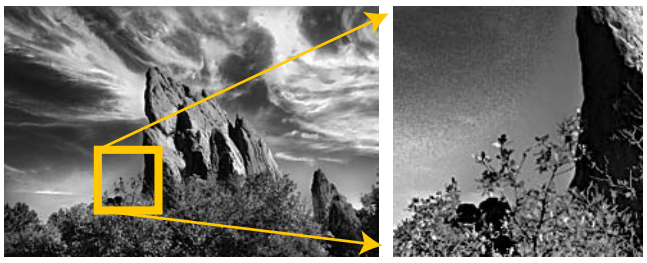


Figure 8: Using a Gaussian filter to locally average the high frequency amplitudes yields halos around strong edges. To prevent this defect, we use an edge-preserving filter.

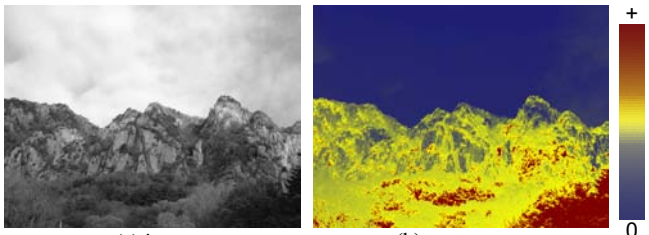


Figure 9: Our measure of textureness indicates the regions with the most contrasted texture.

Textureness Transfer The input I and model M have textureness maps $T(I)$ and $T(M)$, respectively. Using histogram transfer, we enforce the histogram of $T(M)$ onto $T(I)$ to build the desired textureness map T'. To prevent halos, we modify only the detail layer D to approximate T'. We scale the values of D by a ratio ρ to match T' values while accounting for the textureness of the base B' modified by the tonal balance of the previous section:

$$\rho_{\mathbf{p}} = \max\left(0, \frac{T'_{\mathbf{p}} - T(B')_{\mathbf{p}}}{T(D)_{\mathbf{p}}}\right) \quad (7)$$

We do not apply negative ratios, thus preventing gradient reversals. Although this computation is done pixel-wise, we found that the textureness maps are smooth enough to ensure a smooth transformation. We linearly recombine the layers to produce the output: $O = B' + \rho D$.

4.3 Detail Preservation

As illustrated by Figure 10b, the previous result ($O = B' + \rho D$) may result in saturated highlights and shadows. These bright and dark regions are nevertheless of higher importance for photographers who aim for crisp details everywhere. We preserve these details in two steps.

First, we enforce the intensity histogram of the model M to the current output O, which brings back the values within the displayable range. Second, we modify the gradient field to ensure that no details are removed or overly emphasized. Similarly to our shock removal, we build a gradient field $\mathbf{v}$ that satisfies these constraints. We aim at preserving a portion α of the variations of the

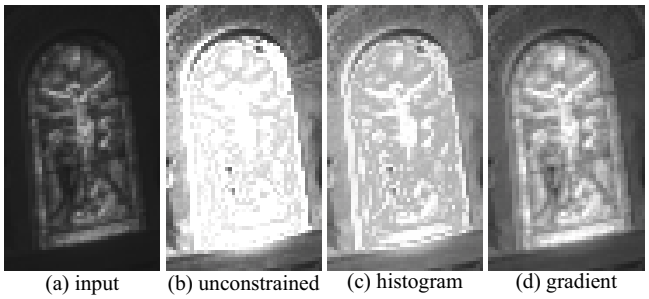


Figure 10: Without constraints, the result may lose valuable details (b) because the highlight are saturated. Enforcing the model histogram brings back the intensity values within the visible range (c). Finally, constraining the gradients to preserve some of the original variations (a) produces high quality details (d).

(a) model (608x785) (b) HDR input (512x768) (c) direct histogram matching (d) our result

Figure 11: Our system can seamlessly handle HDR images . We can turn a sharp picture (b) into a soft grainy and toned photograph (d). We have toned the histogram-transferred version (c) to prevent biased comparison due to different color cast. The model (a) is Accident at the Gare Montparnasse from the Studio Lévy and Sons, 1895. The input (b) is courtesy of Paul Debevec, USC

input image, and we prevent the gradient being increased by a factor greater than β to avoid over-emphasizing noise. We define:

$$x_{\mathbf{v}} = \begin{cases} \alpha\, \partial I/\partial x & \text{if} \quad |\partial O/\partial x| < \alpha\, |\partial I/\partial x| \\ \beta\, \partial I/\partial x & \text{if} \quad |\partial O/\partial x| > \beta\, |\partial I/\partial x| \\ \partial O/\partial x & \text{otherwise} \end{cases} \tag{8}$$

The y component $y_{\mathbf{v}}$ is defined similarly, and the image is reconstructed with the Poisson technique. All that remains is to set α and β. We use percentiles to define $\phi = [p_{95}(O) - p_5(O)]/[p_{95}(I) - p_5(I)]$, which robustly estimates the contrast change induced by our processing. We then use a constant $\alpha = \phi/4$, and we make β depend on intensity in order to avoid increasing noise. We use a smooth-step function $v_\tau(x) = 0$ if $x < \tau$, 1 if $x > 2\tau$, and $1 - [1 - (x-\tau)^2/\tau^2]^2$ otherwise. Setting $\beta = 1 + 3v_\tau\phi$ performs consistently well with $\tau = 0.1$. As a result, we successfully preserve the richness of the input images as shown on Figure 10.

5 Additional Effects

While our focus is on the management of the tonal palette and the variation of detail, we have also developed simple filters to control low-level aspects of the look of a photograph.

Soft Focus and Sharpness The level of sharpness of a picture is a strong aspect of style as exemplified by soft-focus effects. To characterize sharpness, we use difference-of-Gaussian filters and analyze three octaves of the current output O. We set the parameters so that the highest band captures the wavelengths shorter than $\lambda_h = \min(\text{width}, \text{height})/256$. For each band B_i^O, we evaluate the sharpness of the most contrasted edge with the 95th percentile $p_{95}(|B_i^O|)$. We divide this number by $p_{95}(O) - p_5(O)$ to make this measure invariant to intensity. The use of percentiles makes this estimation robust. To summarize, our sharpness estimator is a triplet of numbers $(\zeta_1, \zeta_2, \zeta_3)$ defined as $\zeta_i^O = p_{95}(|B_i^O|)/(p_{95}(O) - p_5(O))$. We compute the same measures for the model M and scale the bands B_i^O of the output by a factor ζ_i^M/ζ_i^O to transfer sharpness. See Figure 11, 15 and 17. In particular, in Figure 11, the intermediate frequencies are attenuated more than the highest frequencies, achieving a "soft-yet-sharp" rendition which is a convincing approximation of the effect produced by a soft-focus lens.

Film Grain and Paper Texture Some photographs exhibit a characteristic appearance due to the paper which they are printed on or because the film grain is visible. We reproduce this effect in two steps. First, since the grain is not part of the image content, we remove it from the model image with a bilateral filter on the luminance values, using $\sigma_r = p_{75}(\|\nabla M\|)$. Then, we crop a sample from the residual (detail) of this bilateral filter in a uniform region. We generate a grain layer using texture synthesis [Heeger and Bergen 1995] (Fig. 1, 11, and 15).

Color and Toning To handle color images, we can use the original a and b channels in the CIE-LAB color space. a and b can be used directly, or they can be scaled by L_O/L_I where L_I and L_O are the luminance of the input and current output. The latter alters color saturation and is useful for HDR images because their chromaticity is often out of the displayable gamut [Fattal et al. 2002; Li et al. 2005]. Figures 13 and 17 show color renditions.

We produce toned pictures (e.g. sepia) using a one-dimensional color map. We use the *Lab* color space to build the functions $a(L)$ and $b(L)$ from the model by averaging a and b for the pixels with a given L. These functions are then applied to the L values of the current result (Fig. 11).

6 Results

We demonstrate our technique using models by different artists on a variety of inputs, including pictures by beginners using point-and-

(a) low resolution (b) high resolution, twice of (a)

Figure 12: Results from lower resolution (a) provides quick previews and allow for interactive adjustments before rendering high resolution results (b). Limited differences are visible on the smallest details (*e.g.* in the background) because they are not well sampled in the low-resolution image.

(a) input image (800x424) (b) result from model (c) result after user adjustment

Figure 13: This rendition was obtained in two steps. We first used Kenro Izu's picture shown in Figure 2b as a model (b). Then, we manually increased the brightness and softened the texture to achieve the final rendition (c) that we felt is more suitable for the scene.

shoot cameras, photos by more advanced amateurs using SLRs, and high-dynamic-range images (Fig. 11).

Computation time varies roughly linearly with the number of pixels, thanks to our fast bilateral filter and a multigrid implementation of gradient reconstruction. For example, the full pipeline for a one megapixel image takes about six seconds on a 2.6GHz Opteron PC, and a four megapixel takes 23 seconds. However, note that we cache intermediate results such as the base, detail, and textureness map, which enables interactive feedback when using the user interface. In addition, results from downsampled images are faithful previews (Fig. 12) because our parameters are scale invariant, which enables fast interaction before a final computation at full resolution.

Our implementation enables interactive adjustment of the parameters through controls such as sliders for scalar parameters and, for the remapping function of the base layer, a spline interface inspired by the "curve" tool of photo-editing software. These adjustments can be saved and reused on subsequent inputs. We have also found that the interactive control is a great way to refine the result of an automatic transfer (Fig. 13).

Figures 15 and 16 shows a comparison of our results with a straightforward histogram matching from the model to the input. Histogram matching ignores the notion of texture and therefore overly increases or decreases the picture detail. In comparison, our technique yields results that are both more faithful to the model and higher quality, with rich shadows and detailed highlights.

Discussion The main cause of failure of our approach is poor input quality. In particular JPEG artifacts and noise noise can be amplified by our detail manipulation (Fig. 14). Apart from this, meaningful input/model couples (two landscapes, two trees, etc) consistently yield faithful transfers, close to our expectations. On more surprising pairs (*e.g.* a flower and a landscape), the process does not generate artifacts and the achieved mood is often pleasing, although one can always argue about the æsthetic quality of some results. Portraits are probably the most challenging type of input, and detail enhancement can lead to unflattering result because skin defects can be emphasized. It is then best to turn this feature off.

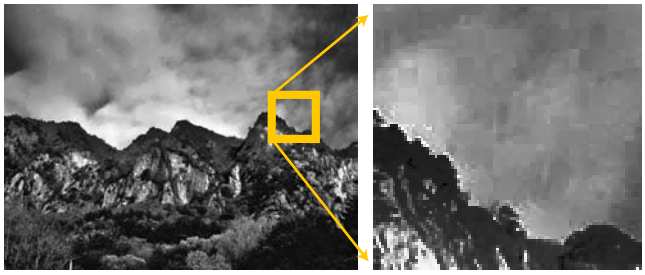


Figure 14: Our technique suffers from imperfections such as JPEG artifacts. In this example, the artifacts in the sky are not visible in the input image (Fig. 9a) but appear clearly after processing.

7 Conclusions

We have presented an approach to manipulate the tonal look of digital photographs. Using a combination of non-linear edge-preserving decomposition and linear analysis, we control both the large-scale tonal palette and the detail over an image. In particular, we manipulate the spatial variation of high-frequencies using a new textureness map that performs an edge-preserving analysis and manipulation of the high-frequency content. We have introduced a gradient constraint that preserves image content and prevents gradient reversal and halos.

Our method can be used to transfer the look of a model photograph or can be directly controlled using a simple interface. It allows for the exploration of a variety of styles and achieves high-quality results that are consistent from low-resolution previews to high-resolution prints.

This work opens several areas of future research. It should be combined with approaches to control the color components of pictorial style. While early experiments with videos have shown that our technique itself is stable, we have found that the biggest challenge is the fluctuation created by auto-exposure, autofocus and the variation of motion blur when the camera moves.

Acknowledgement We thank the reviewers of the MIT Computer Graphics Group and the SIGGRAPH reviewers for insightful feedback. We are especially grateful to Eugene Hsu and Eric Chan for their expert comments on our prints. This work was supported by a National Science Foundation CAREER award 0447561 "Transient Signal Processing for Realistic Imagery," an NSF Grant No. 0429739 "Parametric Analysis and Transfer of Pictorial Style," and a grant from Royal Dutch/Shell Group. Frédo Durand acknowledges a Microsoft Research New Faculty Fellowship. Sylvain Paris was partially supported by a Lavoisier Fellowship from the French "Ministère des Affaires Étrangères." Soonmin Bae is financially supported by the Samsung Lee Kun Hee Scholarship Foundation.

References

ADAMS, A. 1995. *The Print*. Little, Brown and Company.

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D. H., AND COHEN, M. F. 2004. Interactive digital photomontage. *ACM Trans. on Graphics 23*, 3. Proc. of ACM SIGGRAPH conf.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Proc. of ACM SIGGRAPH conf.*

CHOUDHURY, P., AND TUMBLIN, J. E. 2003. The trilateral filter for high contrast images and meshes. In *Proc. of Eurographics Symp. on Rendering*.

DECARLO, D., AND SANTELLA, A. 2002. Stylization and abstraction of photographs. *ACM Trans. on Graphics 21*, 3. Proc. of ACM SIGGRAPH conf.

(a) model (811x1044)

(b) input (795x532)

(c) direct histogram matching

(d) our result

Figure 15: A simple histogram matching from the model (a) to the input (b) increases the texture level of the image (c) whereas the model has little texture. In comparison, we successfully reduce the texture and the sharpness to achieve large uniform gray regions similar to those in the model. The model is Ring Toss by Clarence H. White.

DRORI, I., COHEN-OR, D., AND YESHURUN, H. 2003. Example-based style synthesis. In *Proc. of IEEE conf. on Comp. Vision and Pattern Recognition*.

DURAND, F., AND DORSEY, J. 2002. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. on Graphics 21*, 3. Proc. of ACM SIGGRAPH conf.

EISEMANN, E., AND DURAND, F. 2004. Flash photography enhancement via intrinsic relighting. *ACM Trans. on Graphics 23*, 3. Proc. of ACM SIGGRAPH conf.

FATTAL, R., LISCHINSKI, D., AND WERMAN, M. 2002. Gradient domain high dynamic range compression. *ACM Trans. on Graphics 21*, 3. Proc. of ACM SIGGRAPH conf.

GEIGEL, J., AND MUSGRAVE, F. K. 1997. A model for simulating the photographic development process on digital images. In *Proc. of ACM SIGGRAPH conf.*

GONZALES, R. C., AND WOODS, R. E. 2002. *Digital Image Processing*. Prentice Hall.

GOOCH, A. A., OLSEN, S. C., TUMBLIN, J., AND GOOCH, B. 2005. Color2gray: Salience-preserving color removal. *ACM Trans. on Graphics 24*, 3. Proc. of ACM SIGGRAPH conf.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proc. of ACM SIGGRAPH conf.*

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. In *Proc. of ACM SIGGRAPH conf.*

LI, Y., SHARAN, L., AND ADELSON, E. H. 2005. Compressing and companding high dynamic range images with subband architectures. *ACM Trans. on Graphics 24*, 3. Proc. of ACM SIGGRAPH conf.

PARIS, S., AND DURAND, F. 2006. A fast approximation of the bilateral filter using a signal processing approach. In *Proc. of Eur. Conf. on Comp. Vision*.

PATTANAIK, S., FERWERDA, J., FAIRCHILD, M., AND GREENBERG, D. 1998. A multiscale model of adaptation and spatial vision for realistic image display. In *Proc. of ACM SIGGRAPH conf.*

PETSCHNIGG, G., AGRAWALA, M., HOPPE, H., SZELISKI, R., COHEN, M. F., AND TOYAMA, K. 2004. Digital photography with flash and no-flash image pairs. *ACM Trans. on Graphics 23*, 3. Proc. of the ACM SIGGRAPH conf.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. on Graphics 22*, 3. Proc. of ACM SIGGRAPH conf.

REINHARD, E., STARK, M., SHIRLEY, P., AND FERWERDA, J. 2002. Photographic tone reproduction for digital images. *ACM Trans. on Graphics 21*, 3. Proc. of ACM SIGGRAPH conf.

REINHARD, E., WARD, G., PATTANAIK, S., AND DEBEVEC, P. 2005. *High Dynamic Range Imaging*. Morgan Kaufmann Publishers.

RUDMAN, T. 1994. *The Photographer's Master Printing Course*. Focal Press.

SU, S. L., DURAND, F., AND AGRAWALA, M. 2005. De-emphasis of distracting image regions using texture power maps. In *Proc. of Int. Workshop on Texture Analysis and Synthesis*.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proc. IEEE Int. Conf. on Comp. Vision*.

TUMBLIN, J., AND TURK, G. 1999. LCIS: a boundary hierarchy for detail-preserving contrast reduction. In *Proc. of ACM SIGGRAPH conf.*

(a) model (622x512)

(b) input with auto-levels (876x584)

(c) direct histogram transfer

(d) our result

Figure 16: Our approach is able to reproduce the level of texture observed in Adams' masterpiece (a) to achieve a compelling rendition (d). In comparison, Adobe® Photoshop® "auto-level" tool spans the image histogram on the whole intensity range. This reveals the small features of a picture but offers no control over the image look (b). And, a direct histogram transfer only adjusts the overall contrast and ignores the texture, thereby producing a dull rendition (c).

(a) input image (1200x900)

(b) our result

Figure 17: For color images, we process the luminance channel of the image and keep the original chrominance channels. In this example, the details are enhanced while the overall contrast and sharpness are increased. We used Adams' picture (Fig. 16a) as a model.