

GradientShop: A Gradient-Domain Optimization Framework for Image and Video Filtering

Pravin Bhat¹ C. Lawrence Zitnick²
¹University of Washington

Michael Cohen^{1,2} Brian Curless¹
²Microsoft Research

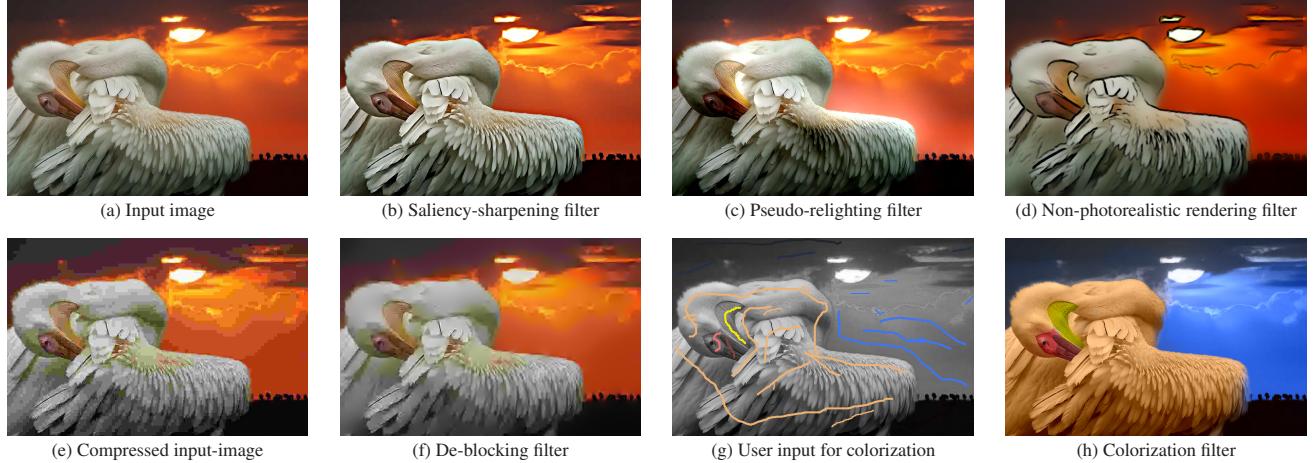


Figure 1: The figure shows some of the image-enhancement filters we have created using the GradientShop optimization-framework. GradientShop has been designed to allow applications to explore gradient-domain solutions for various image processing problems.

Abstract

We present an optimization framework for exploring gradient-domain solutions for image and video processing. The proposed framework unifies many of the key ideas in the gradient-domain literature under a single optimization formulation. Our hope is that this generalized framework will allow the reader to quickly gain a general understanding of the field and contribute new ideas of their own.

We propose a novel metric for measuring local gradient-saliency that identifies salient gradients that give rise to long, coherent edges, even when the individual gradients are faint. We present a general weighting-scheme for gradient-constraints that improves the visual appearance of results. We also provide a solution for applying gradient-domain filters to videos and video streams in a coherent manner.

Finally, we demonstrate the utility of our formulation in creating effective yet simple to implement solutions for various image-processing tasks. To exercise our formulation we have created a new saliency-based sharpen filter and a pseudo image-relighting application. We also revisit and improve upon previously defined filters such as non-photorealistic rendering, image de-blocking, and sparse data interpolation over images (e.g., colorization using optimization).

1 Introduction

Image filtering draws its theory from many different disciplines, and it is used by a diverse community of users that includes scientists, engineers, and artists. In the computer graphics and vision community image filters are used extensively in all the standard rendering and image processing pipelines. In fact, image filtering is often the very first topic covered in most introductory graphics and vision classes.

Over the years, numerous filters have been developed (e.g., sharpening, denoising, morphological operations, quantization), for tasks such as image enhancement, feature detection, and segmentation. Most prevalent image filters directly manipulate pixel values in the *spatial domain* or modulate frequencies in the *frequency domain*. Our hypothesis is that there are three main factors that influence the adoption of a particular domain for filtering:

- **Domain knowledge:** The first factor is accessibility of the knowledge required to design solutions in a domain. The easier it is for newcomers to find and understand the domain literature, the more widespread its adoption is likely to be.
- **Domain power:** Secondly, certain problems are simply better tackled in a particular domain because the solutions in this domain might be more effective, faster, simpler to implement, or more intuitive to understand when compared to possible solutions in other domains.
- **Domain tools:** Finally, the quality of tools available in a particular domain greatly affects its adoption rate. For example, a programmer might be inclined to implement a filter in the spatial domain for the convenience factor, since most programming languages make it easy to perform image input/output and pixel manipulation. However, given access to good frequency-domain tools (e.g., a fast Fourier transform library) the same programmer might prefer to implement the filter in the frequency domain if there are performance benefits to be had.

Gradient domain:

In this paper we focus on a particular form of spatial-domain filtering that has been popularized in recent years as *gradient-domain filtering*. Gradient-domain filters manipulate *pixel differences* (e.g., first order image-gradients) in addition to *pixel values* of an image. A motivation for filtering in the gradient domain is that gradients are integral to the way in which we perceive images. Studies in-

dicate that the human visual system (HVS) does not perceive absolute pixel values, but instead relies upon local contrast and ratios [Attneave 1954; Barten 1999], which more directly correlate with gradients in an image.

Primary goal: In the last decade, the computer graphics and vision community has made some excellent progress in expanding the domain knowledge, power, and tools available to gradient-domain users. Unfortunately, being a relatively young subfield, the key ideas in this literature are scattered across several seminal papers. The primary goal of this paper is to generalize this body of work under a common optimization framework. Our hope is that this generalized framework will make *gradient-domain knowledge* more accessible to newcomers, thus allowing them to quickly gain a general understanding of the field and contribute new ideas of their own. In addition, we hope the *gradient-domain tools* (i.e., the GradientShop API) released with this paper¹, will foster continued research in this area.

Secondary goal: A secondary goal of this paper is to increase adoption of the gradient domain among graphics and vision programmers by demonstrating its *domain power* in designing image filters that would otherwise be unintuitive or difficult to implement in other domains. Gradient-domain filtering provides an easy and intuitive way to alter the perception of an image by modifying its underlying gradient field to selectively emphasize or de-emphasize key image features. Moreover, this paper presents gradient-domain solutions for several image filtering problems that have been previously pursued in other domains. Such gradient-domain solutions can often be more effective, simpler to understand, and/or easier to implement than their state-of-the-art counterparts in other domains.

Contributions: In this paper we present GradientShop, an optimization framework for expressing gradient-domain solutions to image and video processing problems. GradientShop allows applications to tackle challenging image processing problems by simply specifying zeroth-order constraints (i.e., desired pixel values) and first-order constraints (i.e., desired pixel-gradients over space and time) in the optimization to compute the desired result.

Applications can exert further control by specifying weights for individual constraints, which are incorporated into GradientShop’s weighted least-squares solver. Applications may specify their own weighting scheme or, optionally, use GradientShop’s general weighting scheme that has been designed to automatically improve the visual quality of results for most applications when compared to results obtained when using a uniform weighting scheme (i.e., when using a standard, unweighted least-squares optimization).

Gradient-domain filters, when applied independently to individual frames in a video often suffer from flickering and temporal-incoherence artifacts. Our framework decouples the task of defining gradient-domain image filters from the task of applying these filters to a video coherently. GradientShop uses motion-compensated temporal constraints to cause the temporal characteristics of the input video (e.g., temporal coherence, illumination changes) to be enforced in the filtered result. In the case of video streams where all video frames are not available at once, GradientShop filters an incoming video frame while enforcing temporal constraints to the previously filtered video frame, thus helping alleviate the temporal-incoherence problem of filtering video streams.

In addition, we present a new measure for local gradient saliency that allows applications to better process and define gradient constraints. Our saliency measure is motivated by human perception studies that have shown long coherent edges in an image, even when

faint, are perceptually salient to the HVS [Beaudot and Mullen 2003; Elder and Goldberg 2001]. In previous methods the saliency of a local gradient is often approximated by its magnitude as in Lischinski et al. [2006] or by a response to a local filter as in Levin et al. [Levin et al. 2004a]. However, certain pixel gradients, even when faint, give rise to long coherent edges which demarcate object boundaries, shadows, surface creases, and other significant visual artifacts. To measure the perceptual importance of such gradients, our framework accounts for the length of the underlying dominant edge at each pixel and its local orientation. Thus we allow applications to use this length and orientation information to better estimate the saliency of local gradients when processing them. Needless to say, an application may choose to ignore our saliency measure or augment it by using an application-specific saliency detector like a face detector.

Finally, we show how several image processing tasks can be effectively expressed in our formulation. Among the many applications we explore include saliency sharpening, pseudo-relighting, de-blocking, sparse data interpolation over images (e.g., colorization), and non-photorealistic rendering. Using our framework, most filters that can be applied to a single image can also be automatically applied to videos coherently by enforcing simple first-order constraints along flow lines across time.

In summary, our contributions include:

- a general framework for exploring gradient-domain solutions for image & video filtering,
- a novel edge-length based measure for local gradient saliency,
- a general weighting scheme for gradient constraints that improves the visual appearance of results,
- a solution for applying gradient-domain filters to videos and video streams in a coherent manner, and
- a demonstration of our formulation in creating new, and improving upon existing, image and video processing applications.

2 Related work

Our work draws heavily from the rich body of work done on gradient-domain image processing by the computer graphics and vision community. We review some of the work in this literature that is most relevant to our optimization formulation. However, for a more extensive introduction to the gradient-domain literature, the reader is referred to Agrawal and Raskar’s [2007] excellent ICCV course on the topic.

One of the first gradient-domain image filters was proposed by Fattal et al. [2002]. Their work casts the tone-mapping problem as a pure gradient-field integration problem (i.e., no zeroth order terms) by attenuating large scale gradients in an HDR image and then solving for a LDR image that best approximates this attenuated gradient field.

Perez et al. [2003] also used a pure gradient field integration approach to create seamless image composites. The approach they take is similar to the technology used in Adobe Photoshop’s ‘healing brush’, the details of which were later published in an ECCV 2006 paper [Georgiev 2006]. Perez et al.’s algorithm copies the gradients from a source image region onto a target region selected by the user. This modified gradient field is then integrated while keeping pixel colors outside the target region fixed using Dirichlet boundary conditions. Levin et al. [2004b] used a similar approach for seamless image stitching. Levin et al. also showed that their work could be used for de-blocking compressed images. We

¹The GradientShop source code is freely available on www.GradientShop.com

present a similar method for image de-blocking (Section 6.4) that additionally includes zeroth order terms in the optimization to significantly improve the de-blocking quality.

Levin et al. [2004a] proposed a gradient-domain technique for interpolating colors from a sparse set of user-drawn color scribbles over a grayscale image in order to transform it into a photorealistic color image. Lischinski et al. [2006] showed that Levin’s method can also be used to interpolate various local edits such as modifications to colors, tonal values, and white balance of the image. This user-specified data is interpolated over the image in a piecewise-smooth manner with respect to the underlying gradient field of the luminance image. We present a simple improvement to this method (Section 6.5) that significantly reduces data bleeding across faint edges by leveraging our edge-length based measure for local gradient-saliency.

In their Color2Gray paper, Gooch et al. [2005] demonstrate a rather interesting application of gradient-domain techniques. They investigate the problem of converting a color image to grayscale while preserving the color saliency that is often lost in the standard color to gray mapping (i.e., isoluminant colors mapping to the same gray value). This work uses no zeroth order terms in its optimization. However, unlike most gradient domain techniques, this technique employs more than two first-order terms for each pixel in the optimization (i.e., each pixel has constraints defined with respect to the patch of pixels surrounding it).

Orzan et al. [2007] used a gradient-based approach to convert photographs into abstract renditions that capture their salient features. They analyzed the multiscale output of the Canny edge detector to determine both edge importance (measured by its lifetime along the scale axis) and the characteristic edge scale. We propose a filter for non-photorealistic rendering that in comparison to Orzan’s method uses zeroth order terms, a different edge saliency measure, and is temporally consistent when applied to videos. A more detailed comparison is provided in Section 6.2.

Agrawal et al. [2005] proposed a gradient projection technique to fuse gradients obtained from ambient and flash images in a manner that produces well-lit images without strong highlights. Later Agrawal et al. [2006a] generalized this work to a class of edge-suppressing operations on images. Agrawal et al. [2006b] further studied the problem of robust surface reconstruction from a non-integrable gradient field. In this last work, Agrawal et al. explored several different weighting schemes, including but not limited to robust statistics like M-estimators, for robustly satisfying gradient constraints. In Section 3.2 we propose a simple weighting scheme that is well-suited to finding robust solutions to our problem of integrating filtered gradient-fields. Black et al.’s [1998] work is a great resource on the relationship between robust statistics and problems similar to gradient-field integration like anisotropic diffusion.

The temporal constraints (i.e., first-order constraints over time) used in our formulation are inspired by the work of Levin et al. [2004a] and Bhat et al. [2007]. Bhat et al. showed that fusing temporal gradients defined along correspondence vectors from one video with the spatial gradients from another video can be used to combine the temporal characteristics of the former with the spatial characteristics of the latter. We use similar motion-compensated temporal-constraints to encourage the temporal characteristics of the input video (e.g., temporal coherence, illumination changes) to be enforced in the filtered result. We have found that defining temporal-constraints without compensating for motion, as in the works of Drori et al. [2004] and Wang et al. [2004], leads to artifacts like motion-trails and haloing in image filters that make large changes to the spatial appearance of the input video. Thus, our formulation decouples the task of defining a new image filter from the

task of applying that filter to a video coherently.

Our optimization formulation is most similar to the work of Zeng et al. [2006], which proposes a variational model for image editing. Similar to our work, Zeng et al. demonstrate the utility of using a data term in gradient-domain problems. Zeng et al. applied their data term to create a sharpen filter in the gradient domain and improve the results of previous gradient-domain filters like Perez et al.’s [2003] Poisson image editing work.

3 Optimization formulation for image processing

In this section we introduce our optimization formulation for image processing. In section 4 we extend this formulation to accommodate video processing.

The task of an image filter is to take an input image u and transform it into the final image f . Our formulation simplifies the task of writing image processing applications that can be expressed as an energy function involving zeroth and first order terms of the image f (i.e., Equation 2). For each pixel in f the application is allowed to specify a single zeroth-order constraint (i.e., desired pixel value) and two first order constraints (i.e., desired pixel gradients). The application can also specify a weight for each constraint in the optimization.

An application that wishes to use our formulation has to define a function of the following form:

$$F(u, \dots) \rightarrow [d, g, w] \quad (1)$$

Inputs: The function F takes as input the unfiltered image u and any metadata (e.g., parameter values, selection masks, edge statistics) which F may choose to use in its computation. These application-specific inputs to F will be described in further detail in the applications section (Section 6). The input image u may contain multiple channels (e.g., RGB, YUV, etc). However for simplicity of exposition we will treat u as a single-channel image in this section since each corresponding channel in the result f is solved for independently in practice.

Outputs: The function F returns three images – $[d, g, w]$. The image d is a single-channel image that provides the data constraint for each pixel in f . The image g is a two-channel image where channels g^x and g^y specify the desired x -derivative and y -derivative of f respectively. The image w is a three-channel image where channels w^d , w^x , and w^y provide the weights for constraints in d , g^x , and g^y respectively.

The final result f is generated by minimizing the following energy function:

$$E(f) = \sum_{p \in f} E_d(p) + E_g(p) \quad (2)$$

where p is a pixel in f , E_d is our data cost function, and E_g is our gradient cost function. The energy terms E_d and E_g are quadratic functions defined as follows:

$$E_d(p) = w^d(p) [f(p) - d(p)]^2 \quad (3)$$

and

$$E_g(p) = w^x(p) [f_x(p) - g^x(p)]^2 + w^y(p) [f_y(p) - g^y(p)]^2, \quad (4)$$

where f_x and f_y denote the x and y derivative of the final image f .

Thus, the energy terms E_d and E_g are the squared errors between the desired values specified by the function F and the actual values of the final image f . Each constraint also has a corresponding weight, w^d for the zeroth-order ‘data’ constraints and w^x and w^y for the gradient constraints. These weights control the amount of influence a constraint should have on the final image. As shown later, several effects can be achieved by varying these weights, including sparse data interpolation and the suppression of haloing artifacts common to gradient-domain techniques. Individual weights can also be set to zero to completely disable the effect of the corresponding constraint on the result.

Since our energy function E is quadratic, its minima can be found using standard, weighted least-squares techniques like the conjugate-gradient method [Shewchuk 1994]. To increase the runtime performance of the solver, various preconditioners may be used to better condition the optimization [Szeliski 2006]. Bhat et al. [2008a] have proposed a fast Fourier-domain solver that can be used to solve filters that do not use spatially vary weights for the constraints (i.e., w^d , w^x , and w^y are not used). Recently McCann and Pollard [2008] showed that a GPU accelerated conjugate-gradient solver can minimize energy functions like ours in real-time for megapixel-sized images.

For quick reference to the terms defined in this section see the glossary in Appendix A.

3.1 A simple sharpen-Filter

To build the reader’s intuition for image processing using zeroth and first order constraints and to provide further familiarity with our notation, in this subsection we will define a simple sharpen-filter F_{sharpen} using our formulation. This gradient-domain sharpen filter was first defined by Zeng et al. [2006] and was later proved by Bhat et al. [2008a] to subsume the Laplacian sharpen-filter (i.e., $f = u - \lambda \nabla^2 u$) commonly used in image processing. The outputs of F_{sharpen} are defined as follows:

$$d(p) = u(p); \quad g^x(p) = c_s \cdot u_x(p); \quad g^y(p) = c_s \cdot u_y(p);$$

$$w^d(p) = c_1; \quad w^x(p) = 1; \quad w^y(p) = 1$$

Here, the parameter c_s is a scalar constant set to a value greater than one. The sharpening behaviour of F_{sharpen} has an intuitive interpretation. To increase the local constraint of the input image the filter function F_{sharpen} sets the desired gradients of the result (i.e., g^x and g^y) to the gradients of the input (i.e., u_x and u_y) multiplied by a scaling factor c_s . The desired pixel values of the result are set to the input image (i.e., $d = u$). Setting the data constraints to the input image ensures the final result does not drift too far from the input image. Without the use of these this data constraint, the optimization would satisfy the gradient constraints by simply multiplying the input image by the scaling factor [i.e., $f = c_s \cdot u$]. The spatial extent of the sharpening kernel is a function of the ratio between the gradient scale factor and the data weight (i.e. c_s/c_1).

An example result of this sharpen filter can be seen in Figure 5.

3.2 A robust weighting scheme

The L_2 -norm is well known to be sensitive to outliers. This sensitivity causes the energy function E_g to produce visually unappealing results, because it prefers several small errors instead of a few large errors caused by local constraints (i.e., a desired gradient or pixel

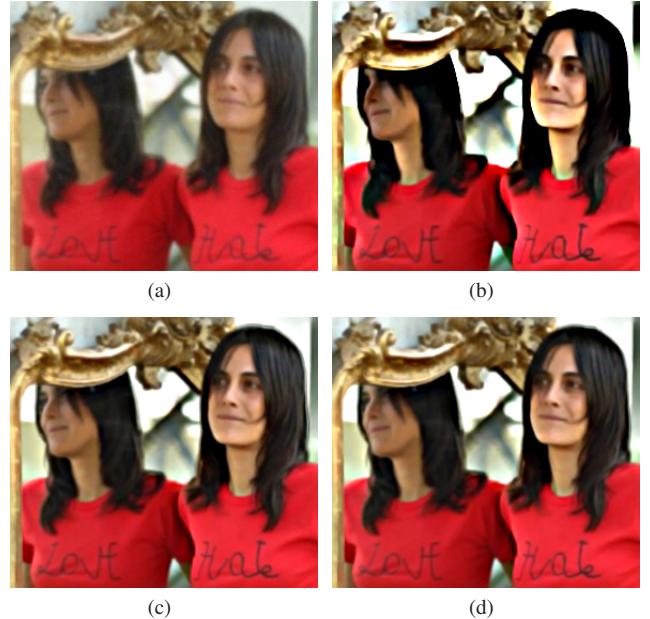


Figure 2: This figure shows the effect our robust weighting function has on the quality of the saliency sharpen filter defined in section 6.1. (a) Input image. (b) Image saliency sharpened with *uniform* weighting; notice the severe haloing artifacts on the woman’s face and neck. (c) Image saliency sharpened using our *robust* weighting. (d) IRLS result after solving ten weighted least-squares problems. See Figure 10 for a similar comparison with the NPR filter and pseudo re-lighting filter.

value) that are difficult to satisfy (i.e., outliers). The L_2 -norm reacts an outlier by distributing the error over a large region around the outlier, which can result in haloing or pinching artifacts in regions where the desired gradient field g is hard to satisfy (see the example in Figure 2).

One solution to this problem is to use a more robust metric such as the L_1 -norm, which would require slower, more complicated optimization techniques like linear programming, or iteratively re-weighted least squares (IRLS). Later in this section we compare against the IRLS algorithm, which is a general method for minimizing a robust norm (e.g., L_1 -norm) by solving successive weighted least squares problems. The method begins by weighting each constraint uniformly. In each successive weighted least squares problem the solution from the previous problem is used to downweight the outliers.

Instead, we introduce an alternative technique that involves solving a single weighted least squares problem. By applying the appropriate weights w^x and w^y to our gradient constraints, the visual artifacts mentioned above can be considerably mitigated. While an application may choose to define its own weights, we provide a default weighting function for the gradient constraints that works well for most applications.

Our robust weighting function is based on the simple prior that the gradient field of f is likely to deviate from g (thus leading to large errors in the L_2 -norm) in regions where g deviates *heavily* from the gradient field of u . By reducing the weights of these constraints we can lower their influence on the resulting image. The weighting functions are defined as follows:

$$w^x = \frac{1}{(|u_x - g^x| + 1)^b} \quad (5)$$

$$w^y = \frac{1}{(|u_y - g^y| + 1)^b} \quad (6)$$

Here the parameter b controls the sensitivity of Equation 4 to outliers and is typically set between 5 and 9; appendix B lists the value of parameter b used by various filters presented in this paper. Figure 2 demonstrates the effect our robust weighting scheme has on the visual quality of the saliency sharpening filter (described in Section 6.1).

Figures 2 and 10 show the effect our robust weighting-function has on the quality of various filters presented in this paper. Figure 2 shows a qualitative comparison of our single weighted least squares method to that produced by the IRLS method after it has solved ten weighted least-squares problems. In this example, our result looks visually identical to the result produced by IRLS, which took ten times longer to compute.

4 Generalization of the formulation to videos

To process an input video u using a filter function F defined using the image formulation in Section 3, one could apply F to each video frame independently. Unfortunately, the resulting video f generated using this approach often suffers from flickering artifacts and therefore looks temporally incoherent.

To alleviate this temporal incoherence problem we are going to use a technique proposed by Bhat et al. [2007]. Bhat et al. showed that fusing temporal gradients defined along motion correspondence vectors from one video with the spatial gradients from another video can be used to combine the temporal characteristics of the former with the spatial characteristics of the later. We use similar motion-compensated temporal constraints to cause the temporal characteristics of the input video (e.g., temporal coherence, illumination changes) to be enforced in the filtered result. Thus, our formulation decouples the task of defining a new image filter from the task of applying that filter to a video coherently.

In addition to the input video, our framework requires as input a set of motion vectors between each consecutive pair of video frames. These vectors are used to define the temporal constraints in the optimization. Although optical flow remains a difficult research problem, we have empirically found that if good motion vectors are available for 50-60% of the pixels and confidence values are available for the motion vectors, then our method produces temporally coherent results. For the streaming video results (Section 4.1) we rely on the blockwise motion vectors encoded in the video, which are obtained directly from the video decoder [Tomar 2006]. For all other results shown in the supplementary video, we use the optical flow algorithm proposed by Sand and Teller [2006] to generate motion vectors.

As in the image processing case, the application's filter function F is used to obtain the desired spatial constraints (i.e., d, g^x, g^y) for each video frame. However, in the video processing case, for every pixel in the video an additional first-order constraint g^v is used to influence what we call the flow gradient. The flow gradient is the difference between a pixel and its motion-compensated neighbor in the previous frame, and is computed as follows:

$$u_v = u(x, y, t) - u(x + v_x, y + v_y, t - 1) \quad (7)$$

Here the coordinate (x, y, t) gives the location of p in u and (v_x, v_y) is a motion vector that maps p to its corresponding pixel in the previous video frame. Thus $(x + v_x, y + v_y, t - 1)$ gives the coordinate of this corresponding pixel.

The value of the desired flow gradients (i.e., the temporal constraints specified by g^v) is computed as a function F_v of the cor-

responding flow gradient in the original video, i.e. $g^v(p) = F_v(u_f(p))$ (Equation 10).

Adding these constraints to our energy function $E(f)$, we get:

$$E(f) = \sum_{p \in f} E_d(p) + E_g(p) + E_v(p). \quad (8)$$

Similar to the data and gradient energy functions, $E_v(p)$ is defined as:

$$E_v(p) = w^v(p) [f_v(p) - g^v(p)]^2 \quad (9)$$

$$g^v(p) = F_v(u_v(p)) \quad (10)$$

where $f_v(p)$ represents the flow gradient in the result at pixel p . The term $w^v(p)$ controls the weight given to p 's temporal constraint (i.e., $g^v(p)$). A typical choice is to set $w^v(p)$ to the confidence in the accuracy of p 's motion vector thus effectively disabling the temporal constraints in regions with bad motion vectors. In all our experiments F_v was set to the identity function thus causing the result video f to mimic the temporal behaviour of the input video u – that is the result video preserves the temporal coherence and lighting changes seen in the input video. However, application designers may choose to modify the behaviour of F_v to better suit their needs.

4.1 Generalization to streaming videos

Though the energy function defined in Equation 8 can be optimized across an entire video, as videos increase in length this global optimization can become computation and memory intensive. It may also be the case that the input video is streaming and thus the entire video may not be available. In either of these cases the energy function may be approximately minimized by stepping through the video one frame at a time with the values of the previous frame fixed. That is, frame $t - 1$ is first computed. Its pixel values are then held fixed and frame t is computed while enforcing the temporal constraints. The very first video frame can be computed without using temporal constraints. The idea of using a variational approach for filtering an incoming frame in a video stream by treating pixels in the previously filtered video-frame as hard constraints was also concurrently proposed by Paris [Paris 2008].

In this paper, only the de-blocking results for streaming *YouTube* videos were created using this approach (see supplementary video).

5 Measuring local gradient saliency

Before we delve into the details of the various filters we have implemented using GradientShop, we first need to discuss our method for detecting local gradient saliency since it is used by a few of our applications. We present a new measure for local gradient saliency inspired by perception studies that show long coherent-edges, even when faint, are perceptually salient to the HSV [Beaudot and Mullen 2003; Elder and Goldberg 2001]. To account for the perceptual importance of gradients that give rise to such edges, our framework provides a long-edge detector that applications can use to measure the saliency of a local gradient.

Our long edge detector finds long, coherent edges instead of simply detecting edges with strong magnitude. The edge detector returns an image e with two channels: e^l and e^o . The e^l channel provides length of the dominant edge running through each pixel in the input image. The e^o channel represents the local orientation of the dominant edge at each pixel.

The main insight here is to factor in edge length instead of simply using the edge magnitude when measuring local gradient saliency. The local edge orientation is used to distribute the saliency weight over the two orthogonal-gradients in x and y directions. Thus our proposed saliency measure for local gradients is as follows:

$$s_x(p) = \cos^2(e^o(p)) \cdot e^l(p) \cdot u_x(p) \quad (11)$$

$$s_y(p) = \sin^2(e^o(p)) \cdot e^l(p) \cdot u_y(p) \quad (12)$$

where, $s_x(p)$ and $s_y(p)$ form our saliency enhanced gradients corresponding to $u_x(p)$ and $u_y(p)$. These saliency enhanced gradients are used by various applications as described in Section 6.

5.1 Implementation details

Our algorithm begins by computing local edge magnitude and orientation at each pixel, using steerable filters [Freeman and Adelson 1991]. Other filters besides steerable filters may also be used, such as finite differences between neighboring pixels. However, we found that the larger spatial extent of steerable filters leads to more robust results. Since we are only concerned with the primary orientation of the underlying image structure, the steerable-filter used is the second derivative of the Gaussian. For a pixel p in an image, this steerable filter estimates local edge magnitude p_m and edge orientation p_o . A visualization of p_m is shown in Figure 3(b).

Since our algorithm needs to detect long, coherent edges even when faint (i.e., not simply edges with strong magnitudes), the estimated local magnitude p_m for each pixel is normalized with respect to the magnitudes in its local neighborhood w (i.e., a 5×5 window).

$$\hat{p}_m = \frac{p_m - \mu_w}{\sigma_w + \epsilon},$$

where μ_w and σ_w denote the average and the standard deviation of edge magnitudes in the pixel neighborhood w . The result from this step is shown in Figure 3(c).

5.1.1 Length estimation using message passing

Having computed the local magnitudes \hat{p}_m and orientations p_o , the algorithm then estimates the length of the underlying edge at each pixel p_l using a message-passing scheme. The length of an edge is defined to be the sum of the normalized, local, edge magnitude for each pixel along the edge. The problem of approximating edge length at a pixel is broken into two sub-problems of estimating the lengths of two sub-edges that start at the pixel and proceed in two opposite directions given by the local edge orientation (i.e., p_o and $p_o + \pi$). During a single iteration of the message-passing scheme each pixel receives two messages that estimate the length of the two sub-edges for the current iteration:

$$m_0^t(p) = \sum_q w_\alpha(q) \cdot w_\theta(q) \cdot (\hat{q}_m + m_0^{t-1}(q)),$$

$$m_1^t(p) = \sum_q w_\alpha(q) \cdot w_\theta(q) \cdot (\hat{q}_m + m_1^{t-1}(q)).$$

The pixel q comes from the four pixels at integer coordinates nearest to the floating-point coordinate obtained when p is projected a distance of $\sqrt{2}$ along the edge orientation p_o for $m_0(p)$ and $p_o + \pi$ for $m_1(p)$.

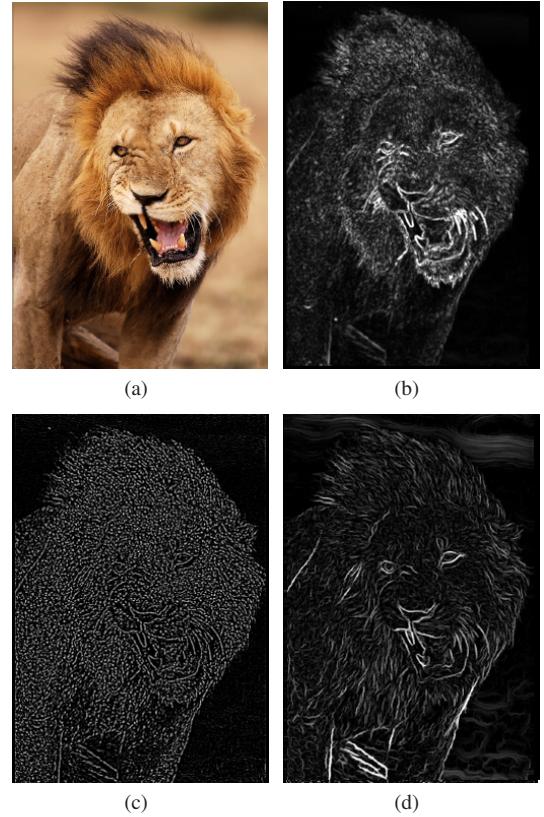


Figure 3: Example of our local gradient saliency measure: (a) original image, (b) local edge magnitudes given by the steerable filter, (c) normalized edge magnitudes, and (d) local gradient saliency (i.e., $\sqrt{s_x(p)^2 + s_y(p)^2}$) produced by our method.

The information at pixel q is weighted by two functions – w_α and w_θ . The function w_α computes the weights for bilinear interpolation between the integer coordinates to approximate the floating-point coordinate. The function w_θ measures the similarity of the local edge orientations at p and q as follows:

$$w_\theta(q) = \exp(-(p_\theta - q_\theta)^2 / 2\sigma_\theta^2),$$

where the variance σ_θ^2 is set to $\pi/5$.

Using this message-passing scheme, the normalized edge magnitudes are propagated along edges for a specified number of iterations. All results in this paper were computed using 60 iterations. The messages for the first iteration are initialized to zero. Note that this message-passing scheme only approximates the length of the underlying edge. For example, an edge that forms a loop with a perimeter that is greater than the number of iterations will be estimated to be longer than its actual length due to some edge magnitudes being counted more than once. The final edge length and orientation for each pixel is set as follows:

$$e^l(p) = m_0^t(p) + m_1^t(p) + \hat{p}_m$$

$$e^o(p) = p_o$$

A visualization of the local gradient saliency computed using the metric given in Equations 11-12 is shown in Figure 3(d).

5.2 Evaluation

In this subsection we will use the NPR filter introduced in Section 6.2 to compare the performance of our saliency measure against the performance of other methods. The NPR filter simply decreases the magnitude of gradients that have low saliency and boosts the magnitudes of gradients with high saliency. Thus the NPR filter exaggerates salient features in an image while abstracting away non-salient details to produce a non-photorealistic rendering of the input image.

Figure 4 compares the effect various saliency measures have on the result produced by the NPR filter. We use the following four saliency measures for comparison:

1. Gradient-magnitude measure: this measure simply uses the local gradient-magnitude computed using finite differences to measure the local gradient saliency.
2. Itti’s measure: this saliency measure was defined by Itti et al. [1998] to estimate the amount of visual attention drawn by an image region. Itti’s saliency measure is inspired by the behavior and the neuronal architecture of the early primate visual system.
3. Canny edge-detector based measure: this measure multiplies the local gradient-magnitude by the grayscale output of the Canny edge-detector.
4. Our measure; Section 6.2 describes how the NPR application uses our saliency measure.

In the scene shown in Figure 4a the person in the foreground has low contrast with respect to the background. Unlike our saliency measure, the other saliency measures fail to identify the gradients across the silhouette of the person to be salient. The NPR filter is best able to delineate the foreground while abstracting away the clutter from the background when using our saliency measure in comparison to the other measures described above. For the input image in Figure 4j, most saliency measures overemphasize the saliency of the short but strong edges around individual bricks in the wall. However, our saliency measure neither overemphasizes the saliency of these gradients nor does it fail to identify the salient but faint gradients in the sky region.

Figure 9 uses the sparse data interpolation (SDI) application defined in Section 6.5 to compare the performance of our saliency measure against that of the *gradient-magnitude* based measure. See Section 6.5 for details on how the SDI application uses our saliency measure. Our measure considerably reduces data bleeding in this application compared to the gradient-magnitude measure, thus requiring less user-input (i.e., data scribbles). In Figure 9 the input image contains semantic regions (e.g., ocean and sky) that are delineated by faint edges (e.g., horizon), which causes the gradient-magnitude based measure to leak data between the different semantic regions.

6 Applications

In this section we present new applications we have developed and a few previously defined applications that we have improved using our perceptually motivated formulation for image and video processing. Each of these applications was written in less than two hundred lines of C++ code using the GradientShop API². We hope that these simple-to-implement, yet effective, applications

²The GradientShop source code is freely available on www.GradientShop.com

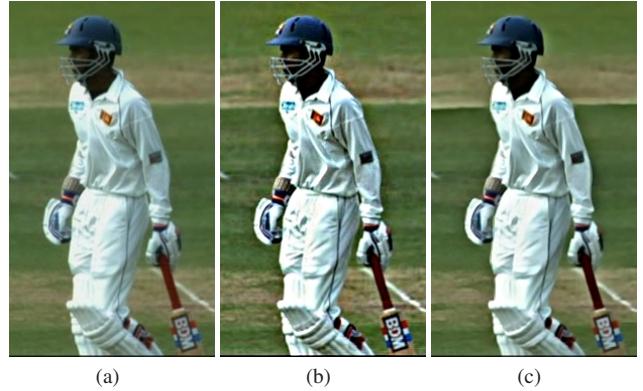


Figure 5: A qualitative comparison of our saliency sharpen filter (Section 6.1) to the simple sharpen filter (Section 3.1). (a) Original image. (b) Simple-sharpen result. (c) Saliency-sharpen result. Notice how saliency sharpen enhances the image without boosting the noise or background texture.

will demonstrate just how intuitive and simple the solutions to many image processing tasks can be when tackled in the gradient domain.

Note that all applications defined in this section, unless explicitly stated otherwise, use our robust weighting function (Section 3.2, equations 5 & 6) for defining the gradient constraint weights (i.e., w^x & w^y). The parameter values used in each application have been listed in Appendix B and can also be obtained from our source code available on the project website [Bhat et al. 2008b].

6.1 A saliency sharpen filter

Sharpening is one of the most commonly used image-enhancement filters. Unfortunately the simple sharpen filter (see Section 3.1) intensifies all gradients in an image, including gradients that arise from noise and background texture. A better sharpen filter would only intensify those gradients that correspond to salient image features. Our saliency sharpen filter uses the output of the long-edge detector (Section 5) to only boost the magnitude of gradients that lie *across* long edges thus enhancing salient gradients without boosting image noise or background clutter. Our saliency sharpen filter is defined as follows:

$$\begin{aligned} F_{\text{saliency_sharpen}}(u, e) &\rightarrow [d, g, w] \\ d(p) &= u(p); w^d(p) = c_1 \\ g^x(p) &= u_x(p) + c_2 \cdot s_x(p) \\ g^y(p) &= u_y(p) + c_2 \cdot s_y(p) \end{aligned}$$

$F_{\text{saliency_sharpen}}$ accepts as input the image to enhance (i.e., u) and the result of the long-edge detector (i.e., e). The data constraints and the parameter c_1 ($c_1 > 0$) keep the enhanced image from drifting too far from the input. The desired gradient field g is defined to intensify the magnitudes of gradients that lie *across* long edges and leave the other gradients unchanged. The parameter c_2 controls the overall sharpening amount ($c_2 > 0$). Here, $s_x(p)$ and $s_y(p)$ provide the saliency enhanced gradients at location p as per the definitions in the Equations 11-12.

See Figure 5 and the supplementary video for a qualitative comparison of our saliency sharpen filter to the simple sharpen filter.

6.2 A filter for non-photorealistic rendering

We will now present a filter for stylizing photographs and videos using non-photorealistic rendering (NPR). Our filter is inspired from a basic technique employed by illustrators when simplifying a scene – that is the ‘abstracting away’ of non-salient features in the scene and the exaggeration of salient features. Like most of our other filters, the NPR filter also measures the saliency of a region using the length of the underlying edge detected by the long-edge detector (Section 5). Specifically, the filter suppresses gradients in regions with short or no edges (i.e., abstraction) and intensifies gradients across long edges (i.e., exaggeration of local contrast). Our NPR filter F_{stylize} is defined as follows:

$$\begin{aligned} F_{\text{stylize}}(u, e) &\rightarrow [d, g, w] \\ d(p) &= u(p); w^d(p) = c_1 \\ g^x(p) &= u_x(p) \cdot \cos^2(e^o(p)) \cdot n(p) \\ g^y(p) &= u_y(p) \cdot \sin^2(e^o(p)) \cdot n(p) \\ n(p) &= c_2(1 - e^{\frac{(e^l(p))^2}{-2\sigma^2}}) \end{aligned}$$

Here, F_{stylize} accepts as input the image to stylize (i.e., u) and the result of the long-edge detector (i.e., e). The function n spatially varies the abstraction/exaggeration amount based on the underlying edge light $e^l(p)$. The parameter σ in function n controls the *abstraction* amount; large values of σ result in large scale features of the input image being abstracted away in the result. The parameter c_2 ($c_2 \geq 1$) controls the amount of *exaggeration* of local contrast across long edges. The data constraints and the parameter c_1 ($c_1 \geq 0$) control how much the stylized image is allowed to drift from the input image. As a postprocessing step, our system optionally overlays a simple visualization of the long edges detected in the input image on top of the result to make it look as if the artist outlined the salient edges using black brush strokes.

Now we will briefly compare our method for non-photorealistic rendering to that of Ozran et al. [2007] and Winnemöller et al. [2006]. Ozran’s method for NPR uses a pure gradient field integration approach (i.e., no data constraints) and as a side effect has to use more complicated contrast equalization and blurring steps to post process their results. In contrast, the data constraints used in our method cause the overall contrast and depth of field effects (e.g., spatially varying blur) in the input image to be automatically reproduced in the result to the amount desired by the user (i.e., using the control parameter c_1). Ozran’s method also does not address the problem of applying their effect to videos in a temporally coherent fashion. Unlike Ozran’s method and Winnemöller’s method, our method not only abstracts away non-salient image features but also exaggerates the contrast of salient image features, which can help accentuate the non-photorealistic look of the result. However, unlike Winnemöller’s method, our method does not currently perform in real-time. See figure 6 and the supplementary video for a qualitative comparison of our method to Ozran and Winnemöller’s methods.

6.3 A pseudo-relighting filter

Image relighting is the process of estimating what an image would have looked like had it been captured under different lighting conditions. Previous relighting algorithms rely on estimating scene geometry in order to produce photorealistic lighting effects [Marschner and Greenberg 1997]. Instead, our relighting filter is inspired by the observation that digital artists can often create

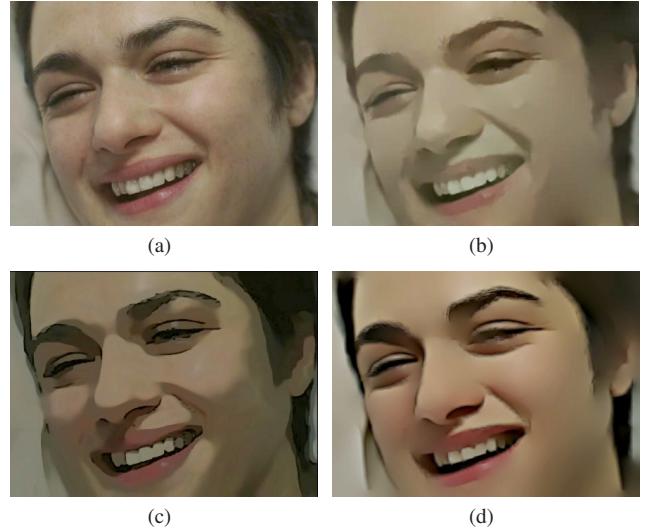


Figure 6: A comparison of our method for non-photorealistic rendering to Ozran and Winnemöller’s methods. (a) Original image. (b) Ozran et al.’s result. (c) Winnemöller et al.’s result. (d) Our result. Unlike Ozran and Winnemöller’s methods, our method not only abstracts away non-salient image features but also exaggerates the contrast of salient image features (e.g., the hairline in this case), which can help accentuate the non-photorealistic look of the result.

pseudo-relighting effects by cleverly adding a few handcrafted intensity ramps onto the original image (e.g., Figure 7g). Our relighting filter allows the user to specify a new lighting direction on the image plane, and then it simply boosts all intensity gradients that happen to be oriented along the specified direction. Integrating the gradient field modified in this manner creates the desired lighting effect by intensifying pre-existing ramps in the image that happen to be aligned with the desired lighting direction. As a result, the relit image looks natural even though the relighting is done without computing any scene geometry. The formal definition of our pseudo-relighting filter is as follows:

$$F_{\text{relight}}(u, o) \rightarrow [d, g, w]$$

Here, F_{relight} accepts as input the image to relight (i.e., u) and an image containing the desired lighting angle for each pixel (i.e., o). We could have used a single constant as the angle parameter instead of o , which is an image of angles. However, by allowing the lighting direction to vary spatially F_{relight} can be used to create a variety of relighting effects as shown in Figure 7. The following are the definitions used by F_{relight} to produce $[d, g, w]$:

$$\begin{aligned} d(p) &= u(p); w^d(p) = c_1 \\ g^x(p) &= u_x(p) + c_2 \cdot u_x(p) \cdot a(p) \\ g^y(p) &= u_y(p) + c_2 \cdot u_y(p) \cdot a(p) \\ a(p) &= \max(0, \frac{u_x(p) \cdot \cos(o(p)) + u_y(p) \cdot \sin(o(p))}{\sqrt{u_x(p)^2 + u_y(p)^2}}) \end{aligned}$$

The data constraints and the parameter c_1 ($c_1 \geq 0$) keep the relit image from drifting too far from the input. The desired gradient field g is defined to boost the local gradient if it happens to be oriented along the local lighting direction. The parameter c_2 controls the maximum gradient boost ($c_2 \geq 0$). The term $a(p)$ computes the

dot product (i.e., cosine of the angle) between the normalized local gradient and the local lighting direction, and then clamps all negative values of the dot product to zero using the max operator. Thus in effect the term $a(p)$ only boosts those gradients that are aligned with the lighting direction, leaving other gradients unchanged.

See Figure 7 and the supplementary video for various relighting results produced by our framework. Figures 7f and 7g compare a relighting result produced automatically by our system to a result produced manually in Photoshop using a radial intensity ramp.

6.4 A de-blocking filter

A common problem with highly compressed images and videos is that they appear blocky because each macroblock in the image/video is compressed independently without accounting for spatial coherence across block boundaries. Perception studies have found blocking to be one of the most distracting compression artifacts ranking alongside low resolution and ringing artifacts. A good de-blocking filter can therefore improve the perceived quality of highly compressed videos found on sites like *YouTube*.

Previous work in the spatial domain: There have been many attempts in the past to define high-quality de-blocking filters in the spatial domain [Averbuch et al. 2005; Castagno and Ramponi 1996; Hong et al. 1996]. Some of the best de-blocking filters in the spatial domain tend to be similar to a bilateral filter. They take the weighted average of pixels across block boundaries in order to suppress blockiness while trying not to over-blur the image. Most of the effort in designing these filters goes into crafting a weighting kernel that can suppress block edges but not affect the true edges in the image. There are three major limitations of these spatial-domain approaches for de-blocking:

1. The de-blocking effect of these filters is localized to a few pixels near the block boundaries. For severely compressed images such de-blocking filters are unable to fully suppress the blocking artifacts.
2. Increasing the size of the de-blocking kernel in order to increase the de-blocking effect invariably over-smoothes the image.
3. Applying these de-blocking filters to individual video frames results in the introduction of temporal artifacts (e.g., flickering).

De-blocking using optimization: Fortunately, the de-blocking problem can be tackled easily by using first order constraints in our formulation. In compressed images, the gradients across macroblock boundaries (i.e., inter-block gradients) are much less reliable than the gradients inside the macroblocks (i.e., intra-block gradients) since each macroblock is compressed independently. Thus, a straightforward de-blocking filter in our formulation would selectively edit inter-block gradients in a manner that suppresses the perceived blockiness. Our experiments show that inter-block gradients with large magnitudes usually correspond to true image gradients that simply happen to coincide with block boundaries. On the other hand, inter-block gradients with small magnitudes usually correspond to gradients with zero magnitude in the uncompressed image and form the major source of perceived blockiness in a compressed image. Therefore, our de-blocking filter selectively suppresses only those gradients that lie across block boundaries and have a small magnitude. The formal definition of our de-blocking filter is as follows:

$$F_{\text{deblock}}(u) \rightarrow [d, g, w]$$

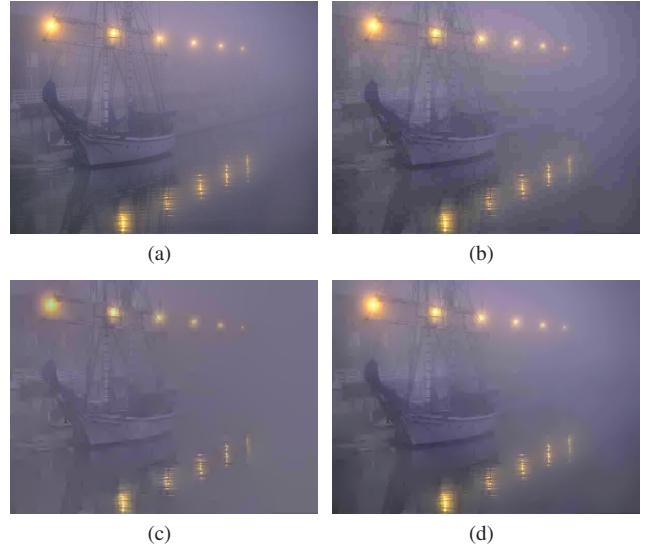


Figure 8: A demonstration of our improvement to Levin’s method for de-blocking images. (a) Original image. (b) Image after compression. (c) De-blocking result using gradient suppression but no data constraints (similar to Levin’s approach); Notice how the highly compressed regions get flattened in appearance. (d) Result produced by our de-blocking method, which uses gradient suppression to reduce blockiness and data constraints to maintain fidelity to the input.

$$\begin{aligned} d(p) &= u(p); w^d(p) = c_1 \\ g^x(p) &= G(u_x(p)) \\ g^y(p) &= G(u_y(p)) \\ G(h) &= \begin{cases} h \cdot S(h) & \text{if } h \text{ is an interblock gradient value} \\ h & \text{otherwise} \end{cases}, \\ S(h) &= 1 - e^{-\frac{h^2}{2\sigma^2}} \end{aligned}$$

The data constraints and the parameter c_1 ($c_1 > 0$) keep the de-blocked image from drifting too far from the input u . The function G suppresses only those gradients that lie across block boundaries (i.e., inter-block gradients). The location of the block boundaries can be easily determined by the file format (i.e., compression type) of u . The function S suppresses gradients with magnitudes close to zero. The parameter σ controls the amount of gradient suppression that happens at block boundaries and this parameter can be learned a priori by compressing a database of raw images to obtain training data. See figures 1 & 8 and the supplementary video for image and video de-blocking results.

Now we will briefly compare our method for de-blocking to that of Levin et al. [2004b], which also works by suppressing inter-block gradients. Their gradient suppression function requires access to the DCT coefficients of each macroblock, which might not be available to the application. More significantly, their approach is a pure gradient field integration approach (i.e., no data constraints). This severely affects their de-blocking quality in regions where the macroblocks only have a single DC coefficient (i.e., a single color). For example, several macroblocks in the sky and water regions of Figure 8b only have a single color. Without the use of data constraints (i.e., $c_1 = 0$), suppressing the inter-block gradients removes the image blockiness but also flattens the appearance of the result (See figure 8c). In contrast, our use of data constraints causes the colors

in the macroblocks to be smoothly interpolated over the sky and water regions as shown in Figure 8d.

Our de-blocking filter can also be thought of as a generalization of the image de-quantization work done by Kim et al. [2007]. In their work, Kim et al. look at the problem of restoring color variations inside uniform regions of a color-quanzitized image. Their algorithm works by minimizing the magnitude of gradients that lie across the boundary between two uniform regions. The flattening behaviour of these gradient constraints is balanced by a data term that encourages the pixel value in the result to be similar to the value in the input image. Our algorithm behaves identically in regions of the image that have been heavily quantized (e.g., macroblocks that contain a single color). However, for macroblocks that contain texture information our algorithm also enforces the underlying gradients that give rise to these textures, thus encouraging the de-blocking result to utilize this additional information.

6.5 A filter for sparse data interpolation

In their seminal work Levin et al. [2004a] demonstrated an optimization approach for colorizing grayscale images using a few user drawn color scribbles. Lischinski et al. [2006] observed that Levin's work was in fact a general and powerful technique for interpolating sparse data over images. Lischinski et al. pointed out that most data channels in images, and not just color channels, are best interpolated in a spatially piecewise-smooth manner with respect to the luminance channel of the image. They demonstrated the generality of Levin's work by interpolating a variety of data types including tonal values, blurring amounts, and white balance corrections specified by the user with a few paint strokes. Lischinski's method maps quite easily to our formulation as follows:

$$F_{\text{sparse_interp}}(u, d) \rightarrow [d, g, w]$$

$$w^d(p) = \begin{cases} \infty & \text{if } d(p) \text{ is defined} \\ 0 & \text{otherwise} \end{cases}$$

$$g^x(p) = 0; g^y(p) = 0$$

The function $F_{\text{sparse_interp}}$ accepts as input an image u that will guide the data interpolation and an image d that contains the user data (e.g., scribbles, paint strokes). The image u is grayscale or in log-luminance space depending on the data to be interpolated. The weights for the data constraints in w^d encourages the result to maintain fidelity to the user input where defined. The null gradient field in g in union with our default weighting function for gradient constraints (Section 3.2, equations 5 & 6) causes the data in d to be interpolated in a piecewise smooth manner with respect to u . In fact, the function $F_{\text{sparse_interp}}$ in union with our default weighting function causes the energy function in equation 2 to become equivalent to the energy function used by Lischinski's method. Substituting zero for the gradient constrains in Equation 4 we get:

$$E_g(p) = \frac{f_x(p)^2 + f_y(p)^2}{(|u_x(p)| + 1)^b + (|u_y(p)| + 1)^b}$$

Improvement: Lischinski's method interpolates sparse data in a piecewise-smooth manner with respect to the underlying image. However, their function for estimating regions where the smoothness constraints have to be softened (i.e., to create the piecewise smooth behavior) depends on the magnitude of a single, local gradient in the image (i.e., Equations 5 & 6). We make a simple modification to Lischinski's method by using our long edge detector to more robustly detect regions that should produce a break in

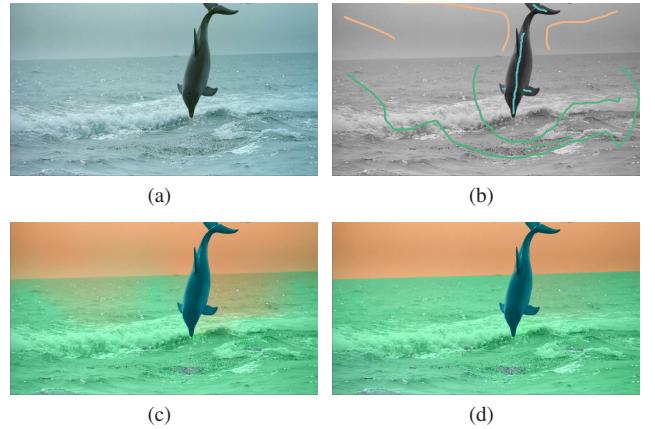


Figure 9: A demonstration of our improvement to Lischinski's method for sparse data interpolation over images. (a) Original color image. (b) User scribbles specifying the desired recolorizing of the image. (c) Colorization result produced by Lischinski's method; Notice the color bleeding between the sky and the ocean. (d) The colorization result produced by our method.

the smoothness of the interpolation. Thus our improvement significantly reduces the amount of data bleeding in the result (or conversely the number of user strokes required to produce the desired result). We *redefine* the weights for the gradient constraints in $F_{\text{sparse_interp}}$ as follows:

$$w^x(p) = \frac{1}{(|s_x(p)| + 1)^b}$$

$$w^y(p) = \frac{1}{(|s_y(p)| + 1)^b}$$

Here, $s_x(p)$ and $s_y(p)$ provide the saliency enhanced gradients at location p as per the definitions in the Equations 11-12. Parameter b controls the sensitivity of data interpolation to local gradient saliency.

Figures 1 & 9 show two results created using our data interpolation algorithm. Also, the supplementary video shows results interpolating data over an entire video where only a few frames have been marked by the user. Figure 9 compares our method to Lischinski's method. The sky and water regions in this example are separated by faint local gradients causing Lischinski's method to exhibit more data bleeding in comparison to our method.

7 Discussion

In summary, we have presented a gradient-domain optimization framework for image and video processing that unifies many of the ideas introduced in prior work under a general framework. Further, our work presents a novel edge-length based measure for local gradient saliency, which we have demonstrated to be a useful measure when processing local gradients in an image. We also presented a simple weighting scheme that is well suited for finding robust solutions to our problem of integrating filtered gradient fields.

In Section 6 we demonstrated the versatility of our formulation by designing and improving a variety of image-processing applications. The ease with which new solutions can be developed using

our framework should be apparent given the simple, intuitive solutions we have arrived at for the applications we considered, which include filters for:

- saliency sharpening,
- non-photorealistic rendering,
- pseudo-relighting,
- de-blocking, and
- sparse data interpolation.

Performance Performance is a major concern when it comes to least-squares based methods for image processing. Our unoptimized C++ code currently spends a few seconds for one megapixel images and nearly one minute per video frame (at 800x600 resolution) starting from the application specific filtering to the full 3D optimization. However, there is plenty of room of improvement since most of the computation time is spent in the weighted least-squares optimization³. Our software based conjugate gradient solver, can be significantly sped up using GPU acceleration and a preconditioner similar to the one proposed by Szeliski et al. [2006]. In fact, McCann and Pollard [2008] have recently shown that a GPU accelerated conjugate gradient solver can minimize energy functions like ours in realtime for megapixel-sized images.

Future Work There are several image processing applications that are likely to yield successful solutions when expressed using our formulation. For example, the LDR2HDR problem addressed by Rempel et al. [2007] could probably be solved with high quality results using our framework. Another interesting exploration of our formulation would be in removing compression artifacts like ringing and mosquito noise, which when combined with our de-blocking filter could significantly improve the perceived quality of streaming videos (e.g., YouTube and teleconferencing videos).

Our optimization framework also has much untapped potential for interactive image editing, especially when combined with learning algorithms that could automatically identify the type of pixels/gradients the user wants to manipulate given a few example brush strokes. Such interactive tools could be used to remove unwanted texture, glare, shadows, and other undesirable artifacts from an image by simply drawing a few rough strokes. Conversely, such tools could also be used to selectively enhance portions of the image for dramatic emphasis. In the coming years, we hope to see the graphics community use and extend our optimization framework to create exciting new image and video processing applications.

³In the case of applications presented in this paper 95% of the time is spent on the least-squares optimization

A Glossary of main terms

u	unfiltered image (i.e., input image)
f	filtered, result (i.e., result image)
p	an arbitrary pixel in f
F	filter function defined by application
$[d, g, w]$	output produced by F
g	desired gradient-field of f (i.e., gradient constraints)
g^x	channel in g specifying the desired x -derivate of f
g^y	channel in g specifying the desired y -derivate of f
d	desired pixel values of f (i.e., data constraints)
w	weights for constraints in g and d
w^d	channel in w specifying the weights for constraints in d
w^x	channel in w specifying the weights for constraints in g^x
w^y	channel in w specifying the weights for constraints in g^y
E	evaluates f 's fidelity to d and g
E_d	evaluates f 's fidelity to d
E_g	evaluates f 's fidelity to g
E_v	evaluates f 's fidelity to the temporal coherence seen in u
e	result of the long-edge detector on an image
e^l	channel in e specifying edge length at each pixel
e^o	channel in e specifying edge orientation at each pixel

B Parameter values for applications

Application	b	c_1
Simple sharpen	0	3.0E-002
Saliency sharpen	5	3.0E-002
Pseudo-relighting	9	1.0E-004
Non-photorealistic rendering	9	1.9E-002
De-blocking	0	2.0E-003
Sparse data interpolation	20	NA

References

- AGRAWAL, A., AND RASKAR, R., 2007. Gradient domain manipulation techniques in vision and graphics.
- AGRAWAL, A., RASKAR, R., NAYAR, S., AND LI, Y., 2005. Removing photography artifacts using gradient projection and flash exposure sampling.
- AGRAWAL, A., RASKAR, R., AND CHELLAPPAN, R. 2006. Edge suppression by gradient field transformation using cross-projection tensors. In *2006 Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, 2301–2308.
- AGRAWAL, A., RASKAR, R., AND CHELLAPPAN, R. 2006. What is the range of surface reconstructions from a gradient field. In *In ECCV*, Springer, 578–591.
- ATTNEAVE, F. 1954. Some informational aspects of visual perception. *Psychol Rev* 61, 3 (May), 183–193.
- AVERBUCH, A., SCHCLAR, A., AND DONOHO, D. 2005. De-blocking of block-transform compressed images using weighted sums of symmetrically aligned pixels. *IEEE Transactions on Image Processing* 14, 2 (February), 200–212.
- BARTEN, P. G. 1999. *Contrast Sensitivity of the Human Eye and Its Effects on Image Quality*. International Society for Optical Engineering.
- BEAUDOT, W., AND MULLEN, K. 2003. How long range is contour integration in human color vision? In *Visual Neuroscience*, vol. 15, 51–64.

- BHAT, P., ZITNICK, C. L., SNAVELY, N., AGARWALA, A., AGRAWALA, M., CURLESS, B., COHEN, M., AND KANG, S. B. 2007. Using photographs to enhance videos of a static scene. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*, J. Kautz and S. Pattanaik, Eds., Eurographics, 327–338.
- BHAT, P., CURLESS, B., COHEN, M., AND ZITNICK, L. 2008. *Fourier Analysis of the 2D Screened Poisson Equation for Gradient Domain Problems*.
- BHAT, P., CURLESS, B., COHEN, M., AND ZITNICK, L., 2008. Gradientshop - image and video processing. <http://www.GradientShop.com>, May 2008.
- BLACK, M. J., SAPIRO, G., MARIMONT, D., AND HEEGER, D., 1998. Robust anisotropic diffusion.
- CASTAGNO, R., AND RAMPONI, G., 1996. A rational filter for the removal of blocking artifacts in image sequences coded at low bitrate.
- DRORI, I., LEYVAND, T., FLEISHMAN, S., COHEN-OR, D., AND YESHURUN., H. 2004. Video operations in the gradient domain. Tech. rep., Tel-Aviv University.
- ELDER, J. H., AND GOLDBERG, R. M. 2001. Image editing in the contour domain. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 3, 291–296.
- FATTAL, R., LISCHINSKI, D., AND WERMAN, M. 2002. Gradient domain high dynamic range compression. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 249–256.
- FREEMAN, W. T., AND ADELSON, E. H. 1991. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence* 13, 9, 891–906.
- GEORGIEV, T. 2006. Covariant derivatives and vision. In *ECCV '06: Proceedings of the 9th European Conference on Computer Vision*.
- GOOCH, A. A., OLSEN, S. C., TUMBLIN, J., AND GOOCH, B. 2005. Color2gray: salience-preserving color removal. *ACM Trans. Graph.* 24, 3, 634–639.
- HONG, S., CHAN, Y., AND SIU, W. 1996. A practical real-time post-processing technique for block effect elimination. II: 21–24.
- ITTI, L., KOCH, C., AND NIEBUR, E. 1998. A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*.
- KIM, TAE-HOON, AHN, JONGWOO, CHOI, AND GYU, M. 2007. Image dequantization: Restoration of quantized colors. *Computer Graphics Forum* 26, 3 (September), 619–626.
- LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2004. Colorization using optimization. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM Press, New York, NY, USA, 689–694.
- LEVIN, A., ZOMET, A., PELEG, S., AND WEISS, Y. 2004. Seamless image stitching in the gradient domain. In *Eighth European Conference on Computer Vision (ECCV 2004)*, Springer-Verlag, 377–389.
- LISCHINSKI, D., FARBMAN, Z., UYTENDAELE, M., AND SZELISKI, R. 2006. Interactive local adjustment of tonal values. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM Press, New York, NY, USA, 646–653.
- MARSCHNER, S. R., AND GREENBERG, D. P. 1997. Inverse lighting for photography. In *Proceedings of the Fifth Color Imaging Conference, Society for Imaging Science and Technology*.
- MCCANN, J., AND POLLARD, N. S. 2008. Real-time gradient-domain painting. *ACM Transactions on Graphics (SIGGRAPH 2008)* 27, 3 (Aug.).
- ORZAN, A., BOUSSEAU, A., BARLA, P., AND THOLLOT, J. 2007. Structure-preserving manipulation of photographs. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*.
- PARIS, S. 2008. Edge-preserving smoothing and mean-shift segmentation of video streams. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, Springer-Verlag, Berlin, Heidelberg, 460–473.
- PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, ACM Press, New York, NY, USA, 313–318.
- REMPEL, A. G., TRENTACOSTE, M., SEETZEN, H., YOUNG, H. D., HEIDRICH, W., WHITEHEAD, L., AND WARD, G. 2007. Ldr2hdr: on-the-fly reverse tone mapping of legacy video and photographs. *ACM Trans. Graph.* 26, 3, 39.
- SAND, P., AND TELLER, S. 2006. Particle video: Long-range motion estimation using point trajectories. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Washington, DC, USA, 2195–2202.
- SHEWCHUK, J. R. 1994. An introduction to the conjugate gradient method without the agonizing pain.
- SZELISKI, R. 2006. Locally adapted hierarchical basis preconditioning. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM Press, New York, NY, USA, 1135–1143.
- TOMAR, S. 2006. Converting video formats with ffmpeg. *Linux J.* 2006, 146, 10.
- WANG, H., RASKAR, R., AND AHUJA, N. 2004. Seamless video editing. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3*, IEEE Computer Society, Washington, DC, USA, 858–861.
- WINNEMÖLLER, H., OLSEN, S. C., AND GOOCH, B. 2006. Real-time video abstraction. *ACM Trans. Graph.* 25, 3, 1221–1226.
- ZENG, Y., CHEN, W., AND PENG, Q. 2006. A novel variational image model: Towards a unified approach to image editing. *Journal of Computer Science and Technology*.

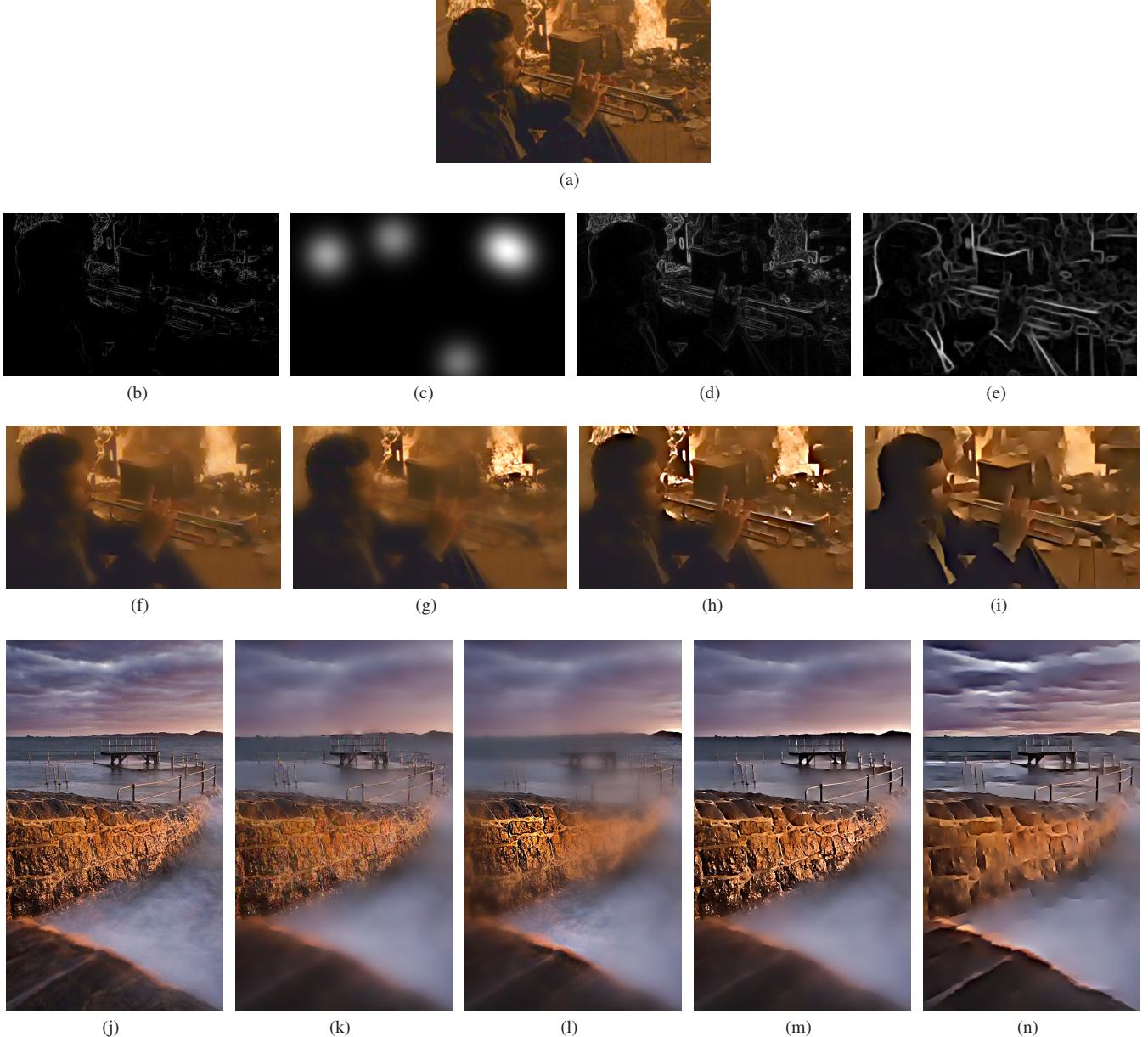


Figure 4: A performance comparison for various gradient saliency measures when used in the NPR filter. (a) Input image. Images in the second row (b-e) show visualizations of the saliency mask generated by various saliency measures – (b) gradient-magnitude measure, (c) Itti's measure, (d) Canny edge-detector, and (e) our saliency measure. Third row images (f-i) show the effect each saliency measure has on result produced by the NPR filter. The fourth row shows the effect of each saliency measure (k-n) for another input image (j). See the images at full resolution to observe the differences. Also, see Section 5.2 for details.

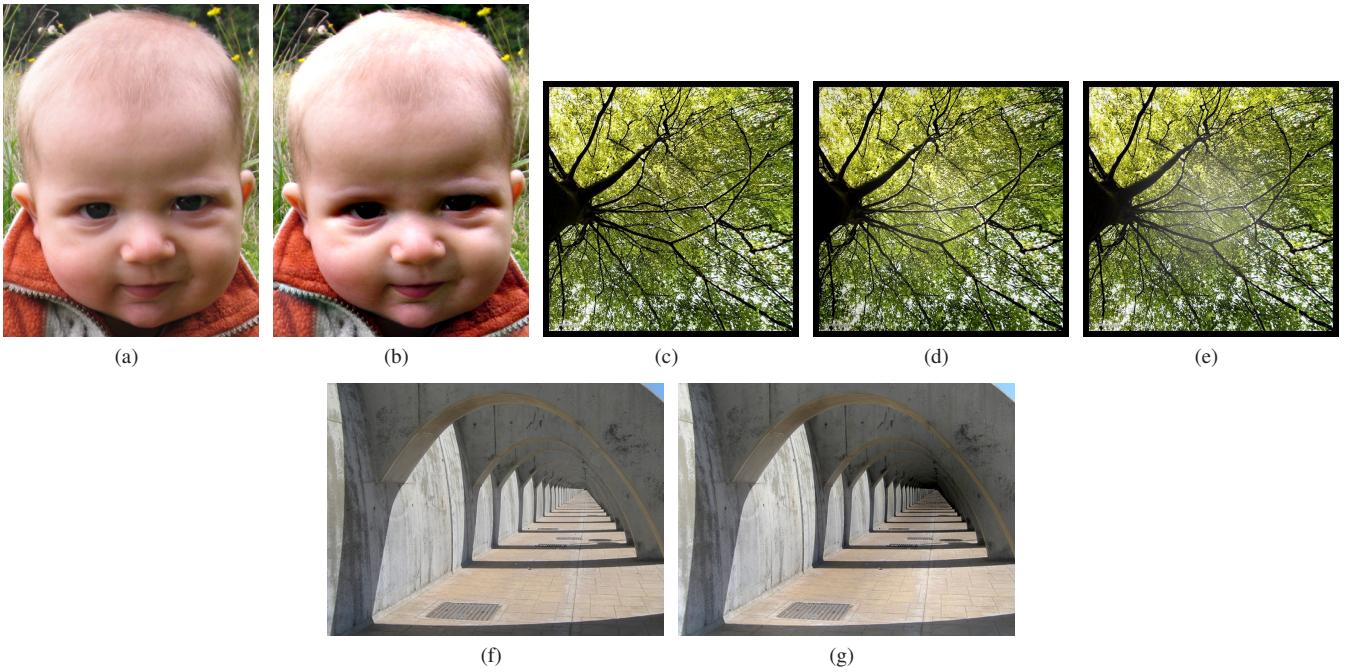


Figure 7: The figure shows some of the pseudo-relighting effects created using the function F_{relight} . (a) Input image. (b) Image relit to simulate an additional light source to the top of the face. The effect is achieved by setting the local lighting direction for every pixel (i.e., $o(p)$ in F_{relight}) to point north. (c) Input image. (d) Image relit to simulate overhead sun light. (e) The same relighting effect attempted in Photoshop by using a radial intensity ramp. Notice that our result (d) looks more realistic in comparison to the Photoshop edit (e). (f) Input image. (g) Image relit to simulate the light fading into the vanishing point, thus adding more depth to the image. The effect is achieved by setting the local lighting direction for every pixel to point away from the vanishing point. Also see the supplementary video to observe the relighting effects more clearly.

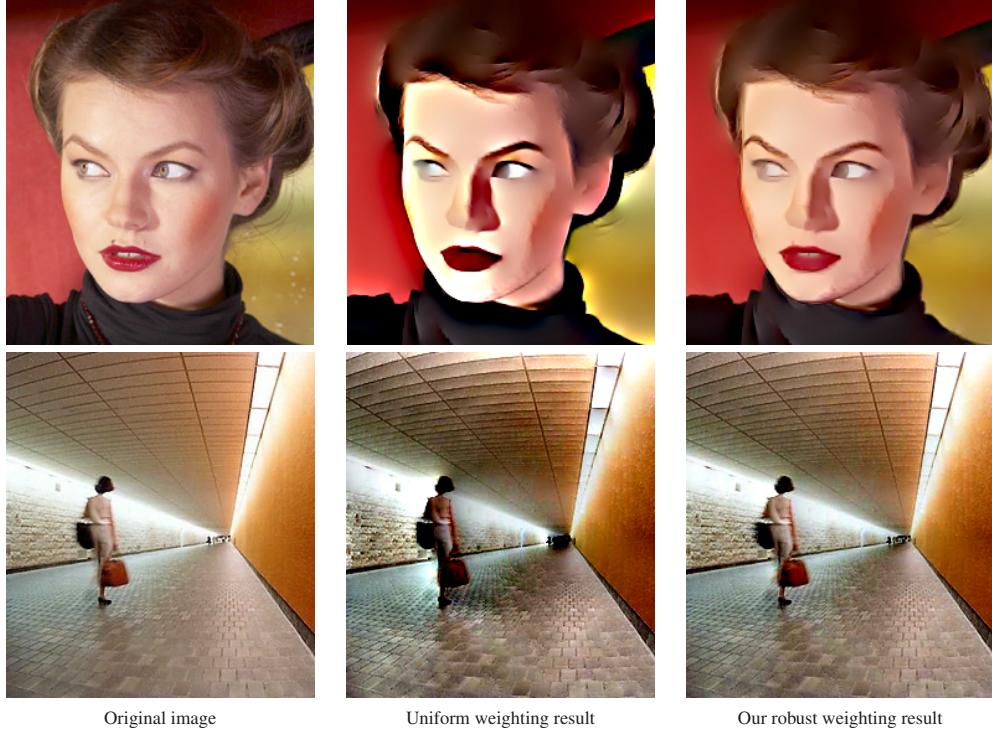


Figure 10: The figure shows the effect our robust weighting function (Section 3.2) has on the quality of the NPR filter (top row) and pseudo re-lighting filter (bottom row). The left column shows the original image; the middle column shows the result when using *robust* weighting; and the right column shows the result when using *uniform* weighting. Notice the absence of haloing and pinching artifacts in the middle column. See Figure 2 for a similar comparison with the saliency sharpen filter.