Antialiasing Recovery

LEI YANG and PEDRO V. SANDER
The Hong Kong University of Science and Technology

JASON LAWRENCE
University of Virginia

and

HUGUES HOPPE
Microsoft Research

We present a method for restoring antialiased edges that are damaged by certain types of nonlinear image filters. This problem arises with many common operations such as intensity thresholding, tone mapping, gamma correction, histogram equalization, bilateral filters, unsharp masking, and certain non-photorealistic filters. We present a simple algorithm that selectively adjusts the local gradients in affected regions of the filtered image so that they are consistent with those in the original image. Our algorithm is highly parallel and is therefore easily implemented on a GPU. Our prototype system can process up to 500 megapixels per second and we present results for a number of different image filters.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture/Image Generation—*Antialiasing*; I.4.3 [**Image Processing and Computer Vision**]: Enhancement—*Filtering*

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Aliasing, Image, Post-processing, Nonlinear filter

ACM Reference Format:

Yang, L., Sander, P. V., Lawrence, J. and Hoppe, H. 201X. Antialiasing Recovery. ACM Trans. Graph. XX, X, Article XXX (XXXXXX 201X), XX pages.
DOI = 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

Authors' addresses: L. Yang, P. V. Sander, Department of Computer Science and Engineering, HKUST, Hong Kong; web: http://www.cse.ust.hk/∼{yanglei, psander}; J. Lawrence, Department of Computer Science, University of Virginia, Charlottesville, VA 22904 USA; web: http://www.cs.virginia.edu/∼jdl; H. Hoppe, Microsoft Research, Redmond, WA 98052 USA; web: http://research.microsoft.com/∼hoppe.

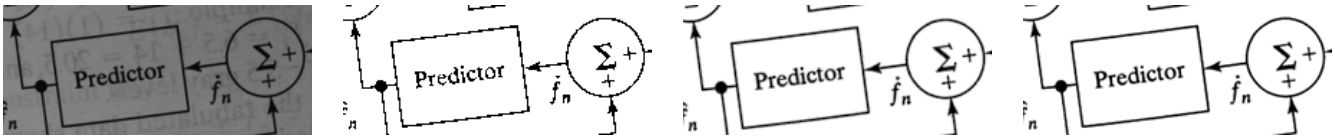
DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

1. INTRODUCTION

Desirable images usually have smooth, antialiased edges. These edges are a fortunate byproduct of capturing an image with a camera, or are computed at a significant cost in off-line and interactive rendering systems. In either case, sharp discontinuities in the scene, such as object silhouettes or material boundaries, are properly filtered to create smooth antialiased transitions. However, this important aspect of an image is easily damaged by many common nonlinear transformations including simple intensity thresholding, tone mapping, gamma correction, color transfer, histogram equalization, applying a bilateral filter, or unsharp masking. Figure 1 shows an example for intensity thresholding. After these filters are applied, edges that were originally nicely antialiased (Figure 1a) often show jagged transitions (Figure 1b).

We present a simple and effective method for restoring antialiased edges that are damaged by these types of filters (Figure 1c). Our approach assumes that the original image is available (Figure 2), that it is free of aliasing artifacts itself, and that the filtered image is in one-to-one pixel correspondence with the original. Thus, our algorithm works in conjunction with any filter that applies a nonlinear transfer function to pixel values independently. It also works for a limited class of nonlinear filters that replace the value at each pixel with a weighted sum of the values in its local neighborhood, such as bilateral filters and unsharp-masking filters. In this way, our method can easily be incorporated into existing image editing software systems (e.g., GIMP, Adobe Photoshop), which provide access to the image data both before and after the application of a particular filter. This offers a compelling alternative to the current state-of-the-art which involves a filter-specific ad-hoc approach to repair damaged antialiased edges. However, our algorithm does not apply to transformations that geometrically distort the original image such as magnification, rotation, or free-form deformations.

Our algorithm has two basic steps. First, in the source image, at each pixel that straddles an edge, we examine the colors adjacent to the edge, and estimate a blending model that reproduces the observed antialiased color from these neighboring colors. More precisely, we estimate the fractional coverage together with the colors on either side of the scene discontinuity, with the assumption that these colors are locally uniform. Second, we adjust the value of each corresponding pixel in the filtered image such that it exhibits the same blending relationship with respect to the colors in its local neighborhood. This has the effect of modifying the local gradients in the filtered image so that they are consistent with the corresponding gradients in the original image. However, we apply this correction adaptively and, guided by an edge detector, modify



(a) Original image *(b) Intensity threshold applied* *(c) Recovered edges (our result)* *(d) Reference*

Figure 1. *Applying simple thresholding to (a) a properly antialiased image results in (b) undesirable jagged edges. We present an algorithm that (c) restores edges that were damaged by these types of nonlinear filters while remaining faithful to the intended effect. We compare this result to (d) a reference image obtained by thresholding a higher-resolution version of this image and then downsampling.*

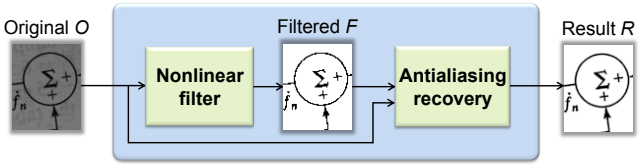


Figure 2. *Schematic of the overall algorithm.*

only those pixels in the filtered image that straddle an edge that is also present in the original image.

Due to the simplicity and parallel structure of our algorithm, it can be easily implemented within a GPU pixel shader. We describe an implementation that is able to process more than 500 megapixels per second. This fast speed makes our method useful as a lightweight post-processing step to accompany existing image filters. We present results of applying our algorithm to images processed with several common filters, and compare it to the related technique of *morphological antialiasing* (MLAA) [Reshetov 2009].

2. PRIOR WORK

Numerous sampling and filtering techniques have been proposed for antialiasing images rendered using ray tracing or rasterization (Pharr and Humphreys [2004] and Akenine-Möller et al. [2008] survey recent techniques). However, far less attention has been paid to the challenging problem of removing aliasing artifacts within an image when there is no access to the underlying 3D scene. One strategy is to perform dimensionality reduction on local image patches using a method such as principal components analysis (PCA) in order to isolate and attenuate high oblique frequencies that typically accompany this phenomenon [Hyvärinen et al. 2009; van Hateren and van der Schaaf 1998].

Reshetov [2009] recently introduced *morphological antialiasing* (MLAA), an image-based method for antialiasing rendered images. MLAA estimates the location and orientation of discontinuous "geometry" edges in the underlying scene using a template-based search for jagged edge patterns in the raster image. These areas are replaced with smooth transitions computed using an analytic edge model. Unlike these prior methods, we focus on a different problem: repairing edge artifacts created by applying a certain class of nonlinear filters to a properly antialiased image. Such artifacts are usually different from those introduced by point-sampling during image synthesis, and do not satisfy the assumptions of previous methods. Our approach is unique in that it uses information from a properly antialiased input image to repair damaged edges in a filtered version. We believe we are the first to propose a solution to this *antialiasing recovery* problem.

Our technique is also related to image matting algorithms (see Wang and Cohen [2007] for a recent survey) in that both involve determining the partial pixel coverage α of overlapping image layers. Image matting attempts to separate the image into two layers, foreground and background, with non-trivial α values occurring only along a closed boundary containing the foreground. However, edges that are damaged by a filter may occur anywhere in the image and often have a complex topology. Our method identifies the complete set of edges even in these difficult cases and does so using only local information at each pixel. This provides a more efficient optimization as compared to most matting algorithms, which must employ a global search. Furthermore, our method is automatic and does not require, for example, the user to specify a trimap.

Our algorithm builds on recent work that shows that the colors within small patches in natural images often occupy a 1D subspace of the full 3D color space [Omer and Werman 2004]. This "color line" model has been successfully used for a number of similar inverse problems including alpha matting [Levin et al. 2006; Bando et al. 2008], denoising [Liu et al. 2007; Joshi et al. 2009], demosaicing [Bennett et al. 2006] and deblurring [Joshi et al. 2009].

3. ANTIALIASING RECOVERY

Figure 3 provides an illustration of the antialiasing recovery problem in 1D. We assume that the portion of the scene imaged at a single pixel is either uniform, in which case no edge is present, or straddles a discontinuous transition between two regions of uniform color. We denote the original image O, the filtered image F, and the recovered image produced by our algorithm R. Let $O[p_i]$ denote the color at pixel p_i. At a pixel that straddles an edge, we assume that the original antialiased color value is a simple linear combination of the colors on either side:

$$O[p_i] = \alpha O[p_{i-1}] + (1-\alpha)O[p_{i+1}], \quad (1)$$

where α is the fractional coverage on the left side of the edge. This is consistent with a smooth antialiased rasterization of this boundary (Figure 3b). Furthermore, we assume that the result of applying a nonlinear filter to this antialiased image is correct in uniform regions, but may be incorrect at edge pixels (Figure 3c). For example, imagine applying a non-monotonic transfer function at each pixel independently. This could replace the value at an edge with a value that is inconsistent with the transition across the boundary (marked in red in Figure 3c). Our goal is to identify the edge pixels in the filtered image F that may have been damaged and to adjust their values so that they exhibit the same transition that was observed in O (Figure 3d). Effectively, we combine O and F to approximate the effect of having applied the nonlinear filter to the colors in the underlying scene and then properly rasterizing the result.

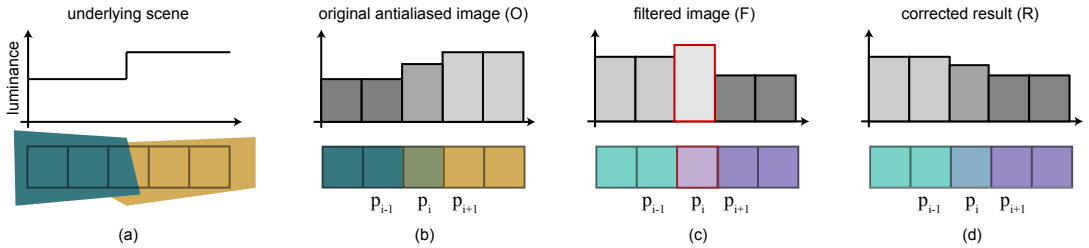


Figure 3. *The image formation model and the stages of our antialiasing recovery algorithm. (a) The 3D scene imaged at each pixel is assumed to be either uniform or to contain a discontinuous edge that separates two regions of locally uniform colors; (b) a properly antialiased and focused image of this simple scene shows a smooth color transition; (c) the result of applying a nonlinear transfer function to the pixel values gives an incorrect result at edge pixels; (d) our algorithm identifies these errant pixels and corrects them to be a consistent weighted combination of the surrounding colors.*

Our method is a generalization of this basic framework from 1D to 2D. It operates independently at each pixel p, and consists of two major steps: (1) estimating an antialiased edge model at each edge pixel p in the original image O, and (2) modifying the corresponding edge pixel p in the filtered image F to reproduce that antialiased edge. We next describe these steps in detail.

3.1 Fitting the antialiased edge model

For each edge pixel p in O, we seek to estimate the two uniform colors c_a and c_b adjacent to the edge, as well as the fractional coverage α, that together best explain the (antialiased) color $c = O[p]$ according to the image formation model in Figure 3. We accomplish this as follows.

Determining the color-space endpoints. Whereas in the 1D case of Figure 3 we could immediately determine the two neighboring colors $c_a = O[p_{i-1}]$ and $c_b = O[p_{i+1}]$, for 2D images the 3×3 neighborhood offers 8 neighbors to choose from. Our solution to this ill-posed problem relies on two assumptions. First, we assume that the two neighboring colors c_a, c_b are extrema in the direction of maximum variance in color space. This not only gives preference to sharp transitions between dissimilar colors, but also respects the principal color distribution of the neighborhood. Second, we assume that in color space, the color c of the center pixel is approximately a linear combination of c_a and c_b, as in Equation 1.

We gather the set of 9 pixel colors c_i in the neighborhood of pixel p (Figure 4a), represented in linear RGB space, and compute their first principal component x. This vector x corresponds to the direction of greatest variance in color space. We obtain x using the iterative Expectation Maximization (EM) scheme of Roweis [1997], which has the benefit of requiring little storage — a crucial property for efficient GPU implementation. This EM scheme reliably converges after 2-3 iterations. We then form the line $l = c + x\,t$ passing through c along the principal component direction, as illustrated (in 2D) in Figure 4b. For each neighboring color c_i, we compute both its distance d_i to the line l, and the parametric coordinate t_i of its projection on l.

Next, we determine c_a, c_b as the two extrema colors along the direction x and within a distance $3\sigma_d$ from line l. Formally, we solve

$$c_a = O[p_a], c_b = O[p_b] \text{ where}$$

$$(a, b) = \arg\max_{(i,j)} (t_i - t_j) \text{ such that } d_i, d_j < 3\sigma_d.$$

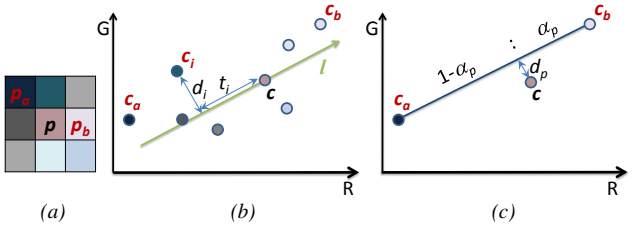


Figure 4. *We examine the 3×3 neighborhood around each edge pixel to determine the two extrema colors c_a and c_b that are assumed to be blended together at the antialiased center pixel.*

The parameter σ_d establishes a threshold on the distance from l that we are willing to tolerate in these endpoints. Its purpose is to remove outliers caused by excessive noise or the presence of a third scene feature.

Determining pixel coverage values. Once the extrema colors c_a, c_b are computed, it is straightforward to determine the blending weights that best reproduce the antialiased center pixel $c = O[p]$. This amounts to solving a linear least-squares system that minimizes

$$d_p = \|(\alpha_p c_a + (1 - \alpha_p)c_b) - c\|\,, \tag{2}$$

subject to the constraint that $0 \le \alpha_p \le 1$. Figure 4c visualizes the residual color error d_p for the optimal value of α_p.

3.2 Correcting the filtered image

The second step in our algorithm is to correct each edge pixel p in F that was potentially damaged by the filter. Specifically, we would like each edge pixel in the recovered image R to exhibit the same local blending that was observed in the input image:

$$R[p] = \alpha_p R[p_a] + (1 - \alpha_p)R[p_b], \tag{3}$$

where the blending factor α_p and the pixel locations p_a and p_b are the same as those computed for $O[p]$. Furthermore, at uniform (non-edge) pixels, we want the corrected image to retain its value in the filtered image:

$$R[p] = F[p]. \tag{4}$$

To achieve this, we use an edge detection algorithm to compute the presence of edges in O and F and balance these competing goals according to the per-pixel edge strength.

GPU	THROUGHPUT	FULL HD IMAGES
NVidia 8800 GTX	98 MP/s	47 fps
AMD ATI HD4870	194 MP/s	94 fps
NVidia 285 GTX	218 MP/s	105 fps
AMD ATI HD5870	559 MP/s	270 fps

Table I. *Performance of our algorithm on different architectures. We present both the throughput in megapixels per second and the average framerate for a* 1920×1080 *Full HD image.*

Determining edge strength. Instead of making a binary decision as to whether a pixel p in F straddles an edge or not, we define a per-pixel edge strength e_p as the product of the Sobel edge detector at p in both O and F (Figure 5). Locating pixels that are near edges in both the original and filtered images lets us restrict antialiasing recovery to regions that were potentially damaged and for which we have information necessary to repair them. More precisely, we define a continuous confidence value between 0 and 1 by combining this edge strength with the residual distance d_p in Equation 2:

$$\beta_p = G(d_p, \sigma_d)(1 - G(e_p, \sigma_e)), \tag{5}$$

where $G(y, \sigma) = \exp(-y^2/\sigma^2)$, and the parameter σ_e controls the sensitivity of our algorithm to the strength reported by the Sobel filter. If $e_p > 3\sigma_e$, we set $\beta_p = 0$ to prevent modifying $F[p]$.

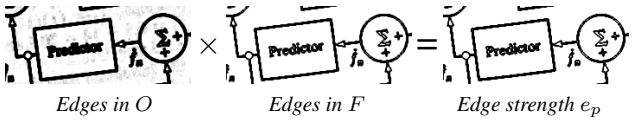


Figure 5. *The Sobel operator applied to the images from Figure 1 and our edge strength measure* e_p*. Larger values of* e_p *are darker.*

Computing the recovered image. To compute the recovered image R, we combine Equations 3 and 4 according to the confidence value in Equation 5:

$$R[p] = \beta_p(\alpha_p R[p_a] + (1 - \alpha_p)R[p_b]) + (1 - \beta_p)F[p]. \tag{6}$$

We obtain R by solving a large sparse linear system that encodes this set of constraints. Note that pixels near edges ($\beta_p > 0$) interact with one another in this system of equations, whereas those in uniform regions as well as local extrema (both with $\beta_p = 0$) act to condition the solution. This formulation has the benefit of tolerating thick edges (wider than one pixel) due to its ability to capture a chain of interactions across several adjacent pixels.

3.3 Real-time GPU implementation

The fact that each step in the algorithm described in the previous sections can be executed in parallel allows implementing a fast solver on modern graphics hardware. Our implementation renders a full-screen quadrilateral in order to process image pixels in parallel in a fragment shader. We apply several iterations of the Jacobi method to solve the linear system in Equation 6 with $R = F$ serving as the starting point. This entails updating the value at each pixel $R[p]$ using values in the previous solution. Because aliased edges tend to span only a few pixels, this process converges quickly within only a few iterations. In our implementation, we always apply exactly 3 iterations. Table I summarizes the running times on various GPUs.

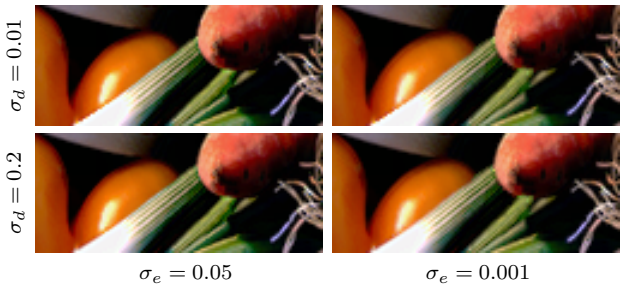


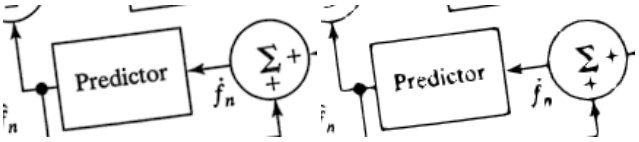
Figure 6. *A demonstration of the insensitivity of our algorithm to its two parameters. The output images are nearly identical over a relatively broad range of values.*

Parameters. The two parameters in our algorithm control its sensitivity to noise in the image σ_d and its sensitivity to the Sobel edge detector σ_e. For a broad class of images (e.g. low-noise digital photographs), a single setting of these parameters performed well in almost every case we considered. We used $\sigma_d = 0.1$ and $\sigma_e = 0.01$ to generate all the results in Section 4. Furthermore, we found that the algorithm is rather insensitive to changing these parameters within a reasonable range (Figure 6). In other situations that include extremely noisy or low-contrast images, or if the user would prefer more aggressive correction to damaged edges, our real-time algorithm permits interactive adjustment of these values.

4. RESULTS

We evaluate our algorithm by using it to repair the output of several common image processing filters. We also compare our technique to applying morphological antialiasing (MLAA) [Reshetov 2009] directly to the filtered image F. This comparison is not altogether fair, since MLAA does not consider the input image and is designed to repair point-sampling artifacts in rendered images rather than damaged edge gradients. However, we include this comparison to emphasize the importance of using the underlying edge information in the input image as is done in our approach.

- **Intensity thresholding** It is common to apply intensity thresholding to a grayscale image to convert it to black-and-white. In the example of Figure 1 this is done to remove noise and elements from the underside of a scanned document. Although this successfully isolates the foreground text, it damages the smooth antialiased edges. The repaired image produced by our algorithm agrees closely with a reference image obtained by applying the threshold to the original image at 16 times the resolution and then downsampling the result. Figure 7 shows a comparison of our algorithm and MLAA for this image. By considering both the input and filtered images, our algorithm does a better job of preserving the structure of the digitized text.
- **Image abstraction** A number of image filters, such as anisotropic diffusion and color quantization, achieve a stylized effect by blurring gradual transitions in an image while emphasizing strong boundaries. The first two rows in Figure 14 show images produced using the technique of Kyprianidis et al. [2009]. Note that smooth edges in the input often become jagged in these stylized renditions and that our algorithm is able to restore the edges while remaining faithful to the effect.



Our result *MLAA applied to F*

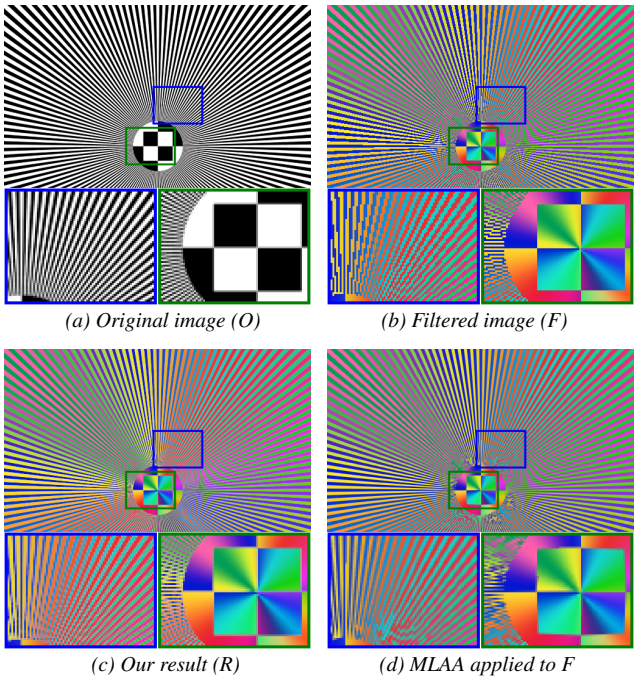
Figure 7. *Comparison of our algorithm with MLAA for the application shown in Figure 1.*

- **Bilateral filter** Bilateral filtering is effective at removing noise in uniform image regions without blurring strong boundaries. However, this can sometimes produce staircase artifacts near edges [Paris et al. 2009] where the transitional colors are biased towards the closer side of the uniform regions, as shown in the third row of Figure 14. This undesired effect is often more pronounced when the bilateral filter is applied iteratively. Joint bilateral upsampling [Kopf et al. 2007] occasionally exhibits a similar problem as seen in the last row of Figure 15. Our algorithm greatly diminishes these artifacts by reproducing edges with smooth transitions like those present in the input.
- **Detail enhancement** Algorithms for enhancing image details and contrast can damage smooth edges. The fourth row of Figure 14 shows the result of applying an unsharp masking filter to a cluttered image of groceries. The bottom row of Figure 14 shows another result of detail enhancement using an off-the-shelf multiscale decomposition system [TopazLabs 2010]. In both examples, our algorithm successfully repairs jagged or distorted edges without undermining the desired enhancement.
- **Color replacement** Replacing colors in an image, as is done in the top row of Figure 15, often leads to artifacts along object boundaries. We found that our algorithm is also useful in these situations as well. Note that this presents a particularly challenging case for MLAA since these edges are not aliased, but rather exhibit an incorrect color transition along the boundary.
- **Color to gray** The second row in Figure 15 shows the color-to-gray filter of Kim et al. [2009] applied to a synthetic image with a combination of strong and weak edges. This filter successfully preserves the weak edges at the cost of damaging the smooth transitions across the strong ones. This image presents a difficult case for our algorithm since it includes corners where multiple colors meet, which violates our color line model. This results in undesired blending between originally similar colors in some of these regions, although our algorithm does restore smooth transitions elsewhere in the image. Furthermore, we have found that these cases are rare in natural images.
- **Gradient mapping** A number of common operations (including image colorization, color transfer and tonal adjustment) map image intensity to a user-specified color gradient. This process often produces artifacts near edges as demonstrated in the images in the third and fourth rows of Figure 15. Our algorithm successfully repairs these jagged transitions even for edges that border regions of detailed texture.

5. DISCUSSION

5.1 Advantage over single-image approaches

Our algorithm uses the original antialiased image to estimate information about how colors are blended across edges in the scene. This information is important and cannot always be accurately reconstructed from the filtered image alone. Our results indicate that incorporating information from both the input and filtered images leads to superior reconstruction, especially in the following cases:



(a) Original image (O) *(b) Filtered image (F)*

(c) Our result (R) *(d) MLAA applied to F*

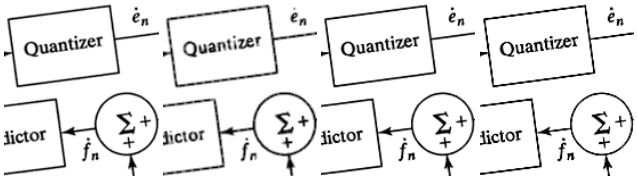
Figure 8. *Our algorithm and MLAA used to repair the edges in an image of a resolution chart after an intensity threshold and a colorization filter were applied.*

- **Partially and already antialiased features** It is difficult to estimate a precise model of the way colors should be blended across edges from the filtered image alone. This is particularly true in instances where the edge is already partly antialiased and a perfect staircasing transition is not present. This explains why MLAA struggles in Figure 14, rows 1/2/3 and Figure 15, row 3, particularly near vertical and horizontal lines. Another situation worth highlighting is that when edges are already antialiased in the filtered image, techniques like MLAA may falsely classify these as edges in need of repair and consequently overblur them (note the face closeup in Figure 14, row 3).
- **Thin and crowded features** Thin lines, especially when they are parallel and close to one another, often lead to jaggies and moiré patterns with many image filters. Thin features are difficult to isolate using pattern matching techniques like those employed by MLAA (note the comparison in Figure 14, row 4 and Figure 15, row 4). Additionally, aliasing patterns may be misidentified as features and can even be reinforced with techniques like MLAA that only consider the filtered image. A striking example of this is shown in Figure 8. An intensity threshold and colorization filter have been applied to a properly antialiased resolution chart. Note the superior job that our algorithm does of repairing the damaged edges and eliminating the aliasing patterns by virtue of having a much more accurate model of the location and characteristics of the scene boundaries. Inferring these boundaries from the filtered image alone is much more difficult and often leads to distortions near thin or small features such as the text in Figure 7.

- **Incorrect chromatic transition** The application of filters such as color replacement and gradient mapping can result in an abrupt colored edge rather than smooth transition (see Figure 3). These types of edges cannot be repaired by analyzing the corrupted image alone (for instance, see MLAA results in Figure 15, row 1/2/3). However, by reproducing the transition present in the original image, our method successfully addresses this case.

5.2 Comparison with a gradient-domain method

The process of extracting the gradient from one image and applying it to another image has become a standard tool for seamless image editing [Fattal et al. 2002; Pérez et al. 2003; Agrawal et al. 2005]. We experimented with using gradient-domain processing for antialiasing recovery. This operates on grayscale images and involves solving a Poisson equation at the edge pixels in the filtered image, imposing that they exhibit the same gradients as in the original image. The values at non-edge pixels in the filtered image are held fixed and serve as boundary conditions. We found that the result of this approach is very susceptible to the overall scale of the target gradient. For example, if a filter changes the image contrast or inverts its intensities, then the gradient must be scaled accordingly to achieve an acceptable solution. Figure 9 shows several results obtained using this gradient-domain approach for repairing the edges damaged by a simple intensity threshold (Figure 1). Even for this simple case, a suitable scale factor for the gradient was determined only through manual trial-and-error. Additionally, this approach is unsuitable for filters that change the scale of the image gradient non-uniformly over the image plane, or combine multiple channels in a color image. In such cases, there is no single scale factor that can be used (Figure 10). Because our method estimates the local blending in 3D color space independently at each pixel, it doesn't suffer from these drawbacks. The local formulation used by our algorithm also results in much greater computationally efficiency compared to a global Poisson solver.

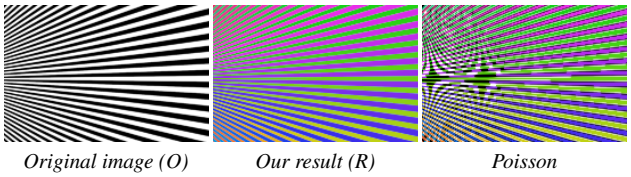


(a) Our result *(b) Poisson* *(c) Poisson* $3 \cdot \nabla$ *(d) Poisson* $5 \cdot \nabla$

Figure 9. *Results of using a gradient-domain method to restore edge pixels damaged by the application of an intensity threshold. (b) Requiring that edge pixels in the filtered image have the same local gradient as those in the original, lower-contrast image leads to excessive smoothing. (c) Only when the gradient values are scaled by an appropriate factor (in this case 3.0) do we achieve a result comparable to ours (reproduced in (a)). (d) Applying a scale factor to the gradient that is too large leads to excessive sharpening.*

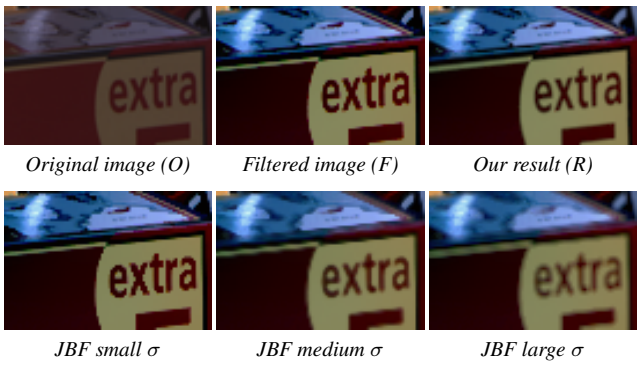
5.3 Comparison with a joint bilateral filter

The notion of using the edge information in one image to guide a filtering process or optimization in another image is related to the the joint/cross bilateral filter (JBF) [Petschnigg et al. 2004; Eisemann and Durand 2004; Kopf et al. 2007]. Intuitively, a JBF can be applied by replacing a damaged edge pixel with a weighted average of its neighboring pixels, where weights are dictated by a Gaussian range filter of color distance in the original image. However, a JBF aims to keep colors separate across edge boundaries instead of smoothly bridging them as is required for antialiasing recovery. Technically speaking, the bilateral weights of pixels that neighbor an edge pixel are not proportional to the fractional contribution they make to the center edge pixel. Therefore, a JBF cannot be directly applied to the problem considered here, as illustrated in Figure 11. Note that using a smaller range variance (σ) exaggerates jaggy artifacts near edges, whereas a larger variance leads to excessive blurring. There is no single setting of this parameter that properly repairs the antialiased edges.



Original image (O) *Our result (R)* *Poisson*

Figure 10. *Using a gradient-domain technique to recover antialiased edges in a color image (cropped from Figure 8). Each color channel is solved separately, using the gradient of a grayscale input. The result contains many artifacts because the scale of the gradient is not uniform. It differs from one channel to the next and even within a single channel at different locations in the image.*



Original image (O) *Filtered image (F)* *Our result (R)*

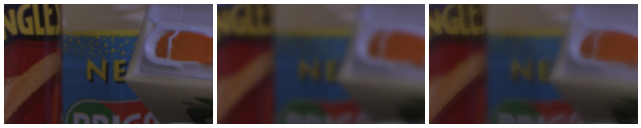
JBF small σ *JBF medium* σ *JBF large* σ

Figure 11. *Comparison of our method to using a joint bilateral filter (JBF), which is not designed for this task. These images show the result of using three different range variances. In some cases, staircase artifacts are exaggerated and in the others the features are excessively blurred. No setting of the variance parameter produces a result that is competitive with our technique.*

5.4 Limitations and future work

A limitation of our technique is that it assumes the input and filtered images are in perfect correspondence. Therefore, our method is not suitable for filters that introduce geometric distortions or intentionally affect the fidelity of the edges. One example of this is a simple Gaussian blur filter. Attempting to use our method to restore the original edges contradicts the intention of such a smoothing filter and may produce artifacts (see Figure 12 for an example). Additionally, our technique is not designed to repair artifacts introduced

by the filter in uniform regions of the input. Figure 13 shows a common haloing artifact produced by an unsharp masking filter. Based on the original image alone, it is impossible to tell whether or not this effect is desirable, and our algorithm does not attempt to alter these regions.



Original image (O) *Filtered image (F)* *Our result (R)*

Figure 12. *Our method used with a standard Gaussian blur. This produces visible artifacts since reproducing sharp edges runs against the goal of this filter.*

Original image (O) *Filtered image (F)* *Our result (R)*

Figure 13. *Haloing artifacts near edges caused by an unsharp masking filter. These types of artifacts, which are introduced by the filter in uniform regions of the input, are not treated by our technique which restricts itself to known edge pixels.*

Future work might consider the more challenging case of filters that geometrically distort the input image. This would require computing a correspondence between the input and output that may not be possible in all cases. Recently, image inpainting techniques based on sparse coding have been proposed [Aharon et al. 2006; Mairal et al. 2008]. These methods may offer an alternative approach for antialiasing recovery that involves building a dictionary of small image patches that straddle edges in the input image and using that dictionary to restore edge information in the filtered image. However, we expect that this approach may share the same "scale factor problem" that we encountered with gradient-domain techniques (Section 5.2).

Another assumption worth relaxing in future work is the color-line assumption. It is violated near corners or in image regions that receive contributions from more than two scene elements. The synthetic color-to-gray result in Figure 15 shows slight over blending that our algorithm produces in these regions with weak edges. However, we consider this an extreme "stress test" of our system and note that at many of these junctions the adjacent colors are very similar in the original image. In more common natural images, we observed that these problems almost never arise, as demonstrated by the many other examples in Figures 8, 14 and 15. Furthermore, we believe this limitation could be addressed through a more general image formation model, although this would likely increase the number of parameters in our optimization and might also require imposing regularization conditions on the solution.

6. CONCLUSION

We have introduced a simple and effective technique for *antialiasing recovery*: repairing antialiased edges that are damaged by certain types of image filters. We presented results that demonstrate the utility and robustness of this technique for a number of common image operations. The fact that our algorithm is embarrassingly parallel allows a real-time implementation on modern graphics hardware. Our method is the first solution to this problem and we anticipate a number of improvements and extensions in future work.

ACKNOWLEDGMENTS

The first two authors were partly supported by RGC GRF grant #619509. We thank the anonymous reviewers for their constructive feedback. Thanks to Diego Nehab and Tian Fang for fruitful discussions. We are grateful to Yongjin Kim, Martin Čadík, Johannes Kopf, Jan Eric Kyprianidis, Giuseppe Papari and Phillip Greenspun, as well as the following Flickr® users for sharing their images: `paullew`, `shoshonasnow` and `thaneeya`.

REFERENCES

AGRAWAL, A., RASKAR, R., NAYAR, S. K., AND LI, Y. 2005. Removing photography artifacts using gradient projection and flash-exposure sampling. *ACM Trans. Graph. 24,* 3, 828–835.

AHARON, M., ELAD, M., AND BRUCKSTEIN, A. 2006. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Process. 54,* 11 (11), 4311 –4322.

AKENINE-MÖLLER, T., HAINES, E., AND HOFFMAN, N. 2008. *Real-Time Rendering*, 3rd ed. AK Peters.

BANDO, Y., CHEN, B.-Y., AND NISHITA, T. 2008. Extracting depth and matte using a color-filtered aperture. *ACM Trans. Graph. 27,* 5, 134.

BENNETT, E. P., UYTTENDAELE, M., ZITNICK, C. L., SZELISKI, R., AND KANG, S. B. 2006. Video and image Bayesian demosaicing with a two color image prior. In *LNCS: Proceedings of ECCV 2006*. Vol. 3951. 508–521.

EISEMANN, E. AND DURAND, F. 2004. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph. 23,* 3, 673–678.

FATTAL, R., LISCHINSKI, D., AND WERMAN, M. 2002. Gradient domain high dynamic range compression. *ACM Trans. Graph. 21,* 3, 249–256.

HYVÄRINEN, A., HURRI, J., AND HOYER, P. O. 2009. *Natural image statistics: A probabilistic approach to early computational vision*. Springer.

JOSHI, N., ZITNICK, C., SZELISKI, R., AND KRIEGMAN, D. 2009. Image deblurring and denoising using color priors. In *Proceedings of IEEE CVPR*. 1550–1557.

KIM, Y., JANG, C., DEMOUTH, J., AND LEE, S. 2009. Robust color-to-gray via nonlinear global mapping. *ACM Trans. Graph. 28,* 5, 161.

KOPF, J., COHEN, M., LISCHINSKI, D., AND UYTTENDAELE, M. 2007. Joint bilateral upsampling. *ACM Trans. Graph. 26,* 3, 96.

KYPRIANIDIS, J. E., KANG, H., AND DÖLLNER, J. 2009. Image and video abstraction by anisotropic Kuwahara filtering. *Computer Graphics Forum 28,* 7, 1955–1963.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2006. A closed form solution to natural image matting. In *Proceedings of IEEE CVPR*. Vol. 1. 61–68.

LIU, C., SZELISKI, R., KANG, S. B., ZITNICK, C. L., AND FREEMAN, W. T. 2007. Automatic estimation and removal of noise from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30,* 2, 299–314.

MAIRAL, J., ELAD, M., AND SAPIRO, G. 2008. Sparse representation for color image restoration. *IEEE Trans. Image Process. 17,* 1 (1), 53 –69.

OMER, I. AND WERMAN, M. 2004. Color lines: Image specific color representation. In *Proceedings of IEEE CVPR*. Vol. 2. 946–953.

PARIS, S., KORNPROBST, P., TUMBLIN, J., AND DURAND, F. 2009. Bilateral filtering: Theory and applications. *Foundations and Trends in Computer Graphics and Vision 4,* 1, 57–62.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. Graph. 22,* 3, 313–318.

PETSCHNIGG, G., SZELISKI, R., AGRAWALA, M., COHEN, M., HOPPE, H., AND TOYAMA, K. 2004. Digital photography with flash and no-flash image pairs. *ACM Trans. Graph. 23,* 3, 664–672.

PHARR, M. AND HUMPHREYS, G. 2004. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann.

RESHETOV, A. 2009. Morphological antialiasing. In *Proceedings of ACM Symposium on High Performance Graphics*. 109–116.

ROWEIS, S. R. 1997. EM algorithms for PCA and SPCA. In *Advances in neural information processing systems*. Vol. 10. MIT Press, 626–632.

TOPAZLABS. 2010. Topaz Detail 2. Software Package by Topaz Labs. http://www.topazlabs.com/detail/.

VAN HATEREN, J. H. AND VAN DER SCHAAF, A. 1998. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society B 265*, 359–366.

WANG, J. AND COHEN, M. 2007. Image and video matting: A survey. *Foundations and Trends in Computer Graphics and Vision 3,* 2.

Received May 2010; accepted February 2011

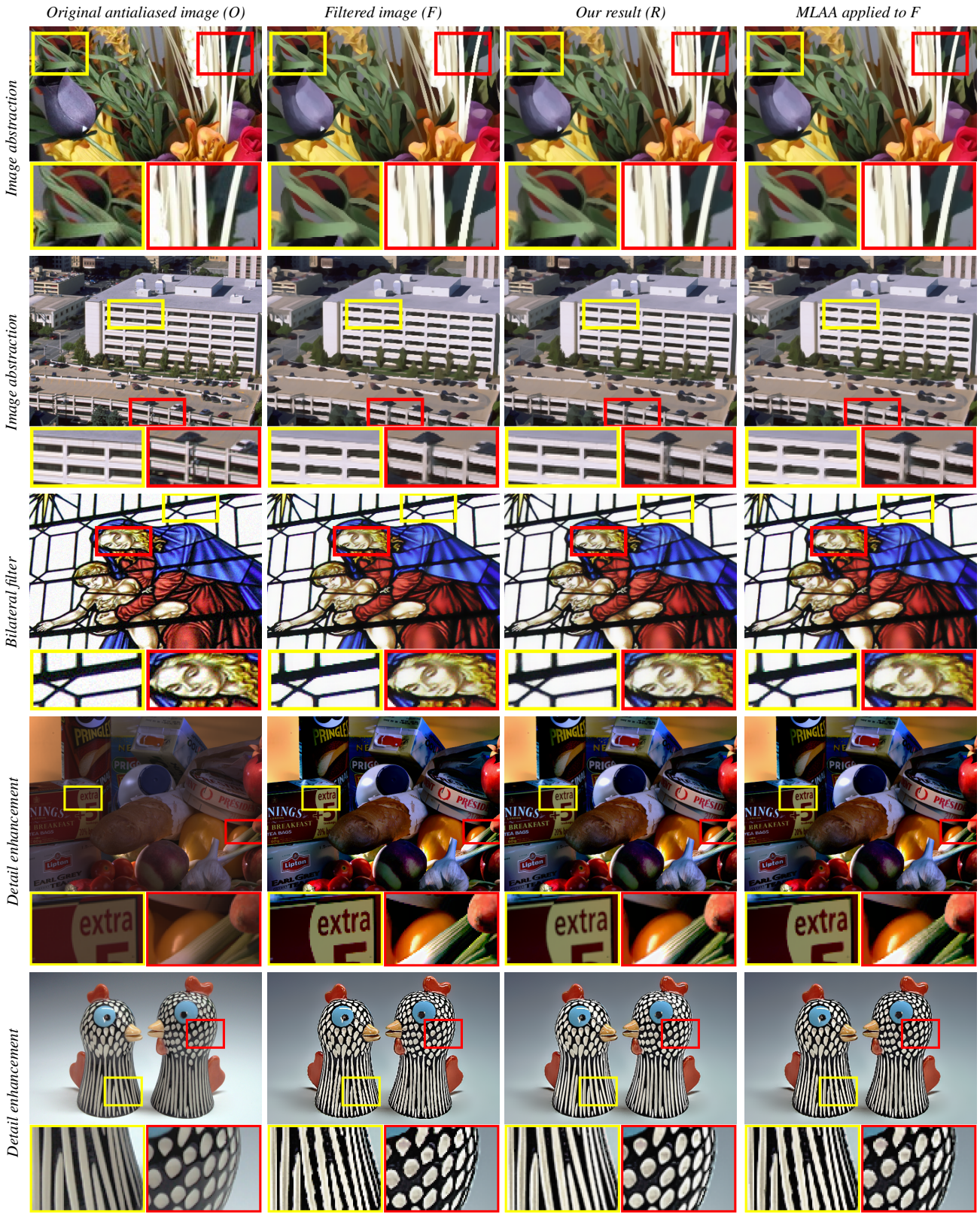


Figure 14. *Results with image abstraction, bilateral filtering, and detail enhancement filters. (Distortions are introduced by some PDF readers. To see the differences most clearly, please zoom in, or refer to the full-resolution images in the supplemental material.)*

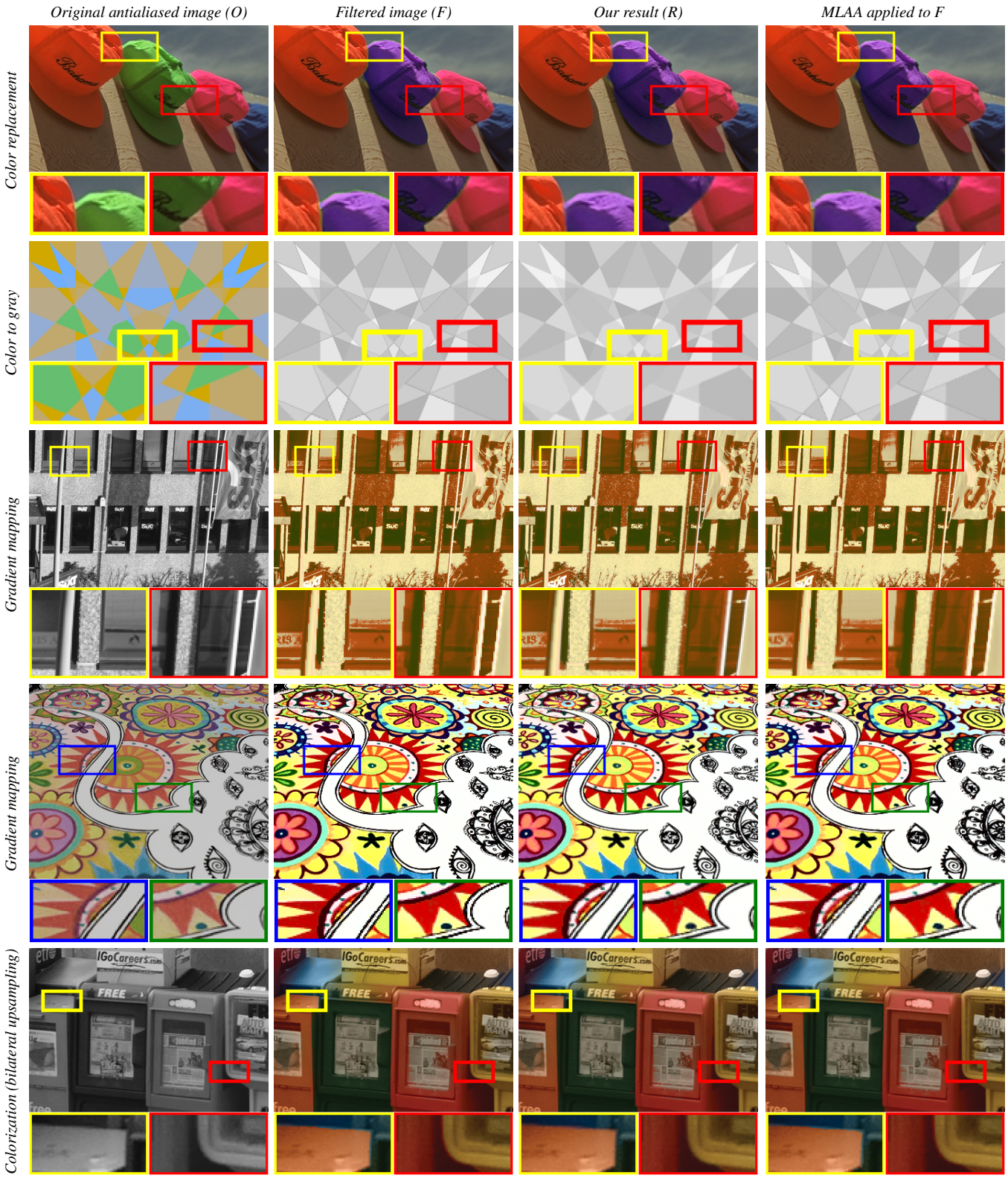


Figure 15. *Results of our algorithm used with color replacement, color to gray, gradient domain filtering, and a colorization technique based on joint bilateral upsampling.*