2014/07/01 v0.5 IPOL article class

Published in Image Processing On Line on 2012–03–24.
Submitted on 2012–00–00, accepted on 2012–00–00.
ISSN 2105–1232 © 2012 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2012.g-dwcs

# Image Demosaicking with Contour Stencils

Pascal Getreuer

Yale University (pascal.getreuer@yale.edu)

**Abstract**

*Demosaicking* (or demosaicing) is the problem of interpolating full color information on an image where only one color component is known at each pixel. Most demosaicking methods involve some kind of estimation of the underlying image structure, for example, choosing adaptively between interpolating in the horizontal or vertical direction. This article discusses the implementation details of the method introduced in Getreuer, "Color Demosaicing with Contour Stencils," 2011. *Mosaicked contour stencils* first estimate the image contour orientations directly from the mosaicked data. The mosaicked contour stencils are then used to guide a simple demosaicking method based on graph regularization.

**Source Code**

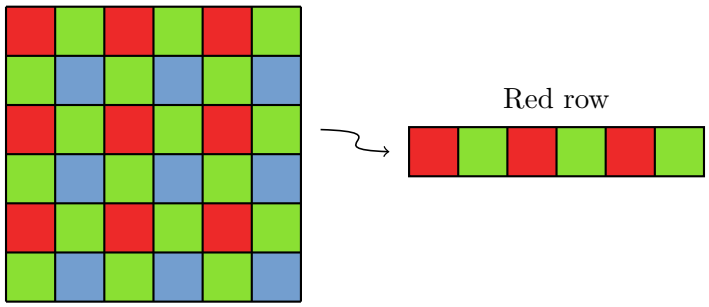
ANSI C source code to produce the same results as the demo is accessible on the article web page http://dx.doi.org/10.5201/ipol.2012.g-dwcs. Future software releases and updates will be posted at http://dev.ipol.im/~getreuer/code.

**Keywords:** image demosaicking, image level-line analysis

## 1 The Bayer Pattern

Most digital cameras use only a single sensor at each pixel location. A color filter array (CFA) is placed in front of the sensor to allow light of one color, usually red, green, or blue, to pass through to each pixel. Therefore, the camera observes a mosaicked image.

The Bayer pattern [1] is the most commonly used pattern for the CFA.




Green pixel locations are arranged in a quincunx lattice and cover half the array. The red and blue pixel locations are spaced uniformly every two pixels and each cover a quarter of the array. The pattern alternates between "red rows" and "blue rows." In a red row the pattern is $R, G, R, G, \ldots$, and in a blue row it is $G, B, G, B, \ldots$.

## 2 Mosaicked Contour Stencils

Contour stencils [7, 8, 12] are a method for estimating the contours of a uniformly sampled grayscale or color image. The structure of a grayscale image is determined by computing variations of the form

$$(\mathcal{S} \star [v]) := \sum_{m,n \in \mathbb{Z}^2} \mathcal{S}(m,n)|v_m - v_n|, \tag{1}$$

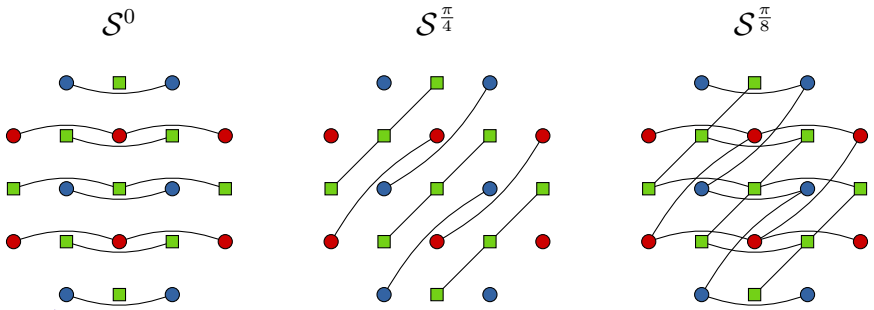
where $\mathcal{S} : \mathbb{Z}^2 \times \mathbb{Z}^2 \to \mathbb{R}^+$" is the *contour stencil*, a function describing weighted edges between the pixels of $v$. The image contours are estimated by finding the best-fitting stencil

$$\mathcal{S}^\star = \arg\min_{\mathcal{S} \in \Sigma} (\mathcal{S} \star [v]) \tag{2}$$

where $\Sigma$ is a set of candidate stencils.

Mosaicked contour stencils [13] are a modification of contour stencils for the Bayer pattern. To extend contour stencils to mosaicked data, the key difference from the case with grayscale images is that absolute differences are only meaningful between pixels of the same color channel. Therefore, the stencil should be selected according to the CFA so that $\mathcal{S}(m,n)$ is nonzero only where $m$ and $n$ sample the same channel.

For developing stencils on the Bayer pattern, there are two distinct cases: stencils centered on a green location and stencils centered on a non-green location. The proposed stencils for stencils centered on a green location are illustrated below.




*Stencils centered on a green location.*

These stencils are the result of a design optimization described in [13]. The edge weights are nonzero where an edge is drawn, indicating that an absolute difference is to be computed between the two linked pixels. For the horizontally-oriented stencil $\mathcal{S}^0$, the nonzero edge weights $\mathcal{S}(m,n)$ are $1/22$. For the diagonally-oriented stencil $\mathcal{S}^{\frac{\pi}{4}}$, the nonzero edge weights are $2^{1/2}/28$. The stencil at orientation $\pi/8$ is formed as a linear combination of the other two,

$$\mathcal{S}^{\frac{\pi}{8}} = \frac{\mathcal{S}^0 + \mathcal{S}^{\frac{\pi}{4}}}{1 + \frac{1}{\sqrt{2}}(\cot\frac{\pi}{16} - 1)}. \tag{3}$$

For example, the horizontal variation with $\mathcal{S}^0$, is computed as

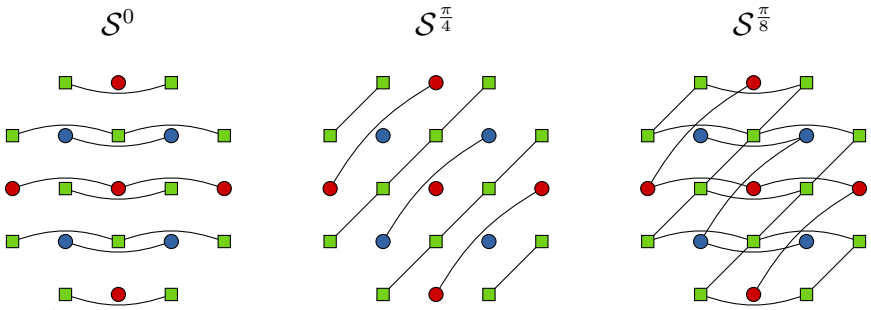
$$\begin{aligned}(\mathcal{S}^0 \star [v])(i,j) &= \sum_{m,n \in \mathbb{Z}^2} \mathcal{S}^0(m,n)|v_m - v_n| \\ &= \tfrac{1}{22}\big(|v_{i-1,j+2} - v_{i+1,j+2}| \\ &\quad + |v_{i-2,j+1} - v_{i,j+1}| + |v_{i-1,j+1} - v_{i+1,j+1}| + |v_{i,j+1} - v_{i+2,j+1}| \\ &\quad + |v_{i-2,j} - v_{i,j}| + |v_{i-1,j} - v_{i+1,j}| + |v_{i,j} - v_{i+2,j}| \\ &\quad + |v_{i-2,j-1} - v_{i,j-1}| + |v_{i-1,j-1} - v_{i+1,j-1}| + |v_{i,j-1} - v_{i+2,j-1}| \\ &\quad + |v_{i-1,j-2} - v_{i+1,j-2}|\big).\end{aligned} \tag{4}$$

The notation $v_{i,j}$ denotes the value of image $v$ at pixel location $(i, j)$, where the horizontal index $i$ increases to the right and the vertical index $j$ increases upwards.

By rotations of these stencils, we define eight stencils $\mathcal{S}^{k\frac{\pi}{8}}$ at orientations $k\pi/8$, $k = 0, \ldots, 7$. The values of the edge weights are such that the mosaicked contour stencils satisfy an approximate rotational invariance: let $f(x) = x_1 \sin\theta - x_2 \cos\theta$, then

$$|\theta - \tfrac{\pi}{16}j| < \tfrac{\pi}{16} \quad \Rightarrow \quad \mathcal{S}^{\frac{\pi}{8}j} = \operatorname*{arg\,min}_{\mathcal{S}\in\Sigma} (\mathcal{S} \star [f]). \tag{5}$$

The following figure shows the stencils centered on a red location. The nonzero edge weights are determined in the same way.




*Stencils centered on a red location (similarly for blue).*

For every pixel in the mosaicked image, mosaicked contour stencils are applied to compute the variation in eight orientations. The stencil yielding the smallest variation is selected as the best-fitting stencil $\mathcal{S}^\star$. The orientation modeled by $\mathcal{S}^\star$ is an effective estimate of the true contour orientation.

Hamilton–Adams [2], Chung–Chan [6], and some other existing methods for demosaicking use sums of absolute differences for orientation detection as well. Compared to contour stencils, they are more limited in the number of possible orientations, many of them only considering horizontal vs. vertical.

# 3 Demosaicking

As a proof of concept that the improved angular resolution of contour stencils is useful for demosaicking, we introduced [13] a demosaicking method based on minimization. The objective function is a graph regularization, which imposes weighted penalties on the differences between neighboring pixels according to the edge weights of the graph. The orientations estimated by the contour stencils are used to construction the graph. Additionally, the minimization is constrained such that the result agrees with the input mosaic. A similar graph regularization approach was applied to image interpolation in [8].

Demosaicking is performed by solving the constrained minimization

$$\begin{cases} \displaystyle\operatorname*{arg\,min}_{u} \sum_m \Big(\sum_n \big(w_{m,n}\|u_m - u_n\|_L\big)^2\Big)^{1/2} + \alpha \sum_m \Big(\sum_n \big(w_{m,n}\|u_m - u_n\|_C\big)^2\Big)^{1/2} \\ \text{subject to } u^k_m = f_m,\ m \in \Omega^k, k \in \{R, G, B\}, \end{cases} \tag{6}$$

where

- $f$ denotes the observed mosaicked image,
- $u$ is the demosaicked image that is found by the minimization,
- $\Omega^k$ denotes the set of pixel locations where color component $k$ is known, $k \in \{R, G, B\}$,

- $\alpha$ is a positive parameter,
- $\|\cdot\|_L$ and $\|\cdot\|_C$ are seminorms in the color space proposed by Condat [9],

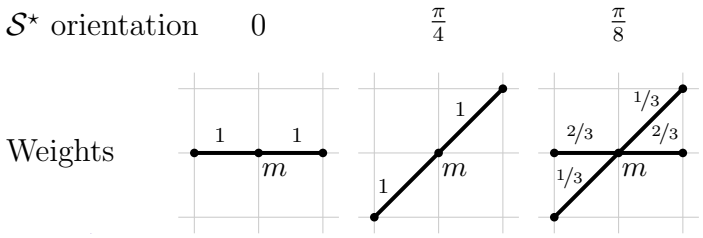
$$\|x\|_L := \tfrac{1}{\sqrt{3}}|x^R + x^G + x^B|, \tag{7}$$

$$\|x\|_C := \sqrt{\tfrac{1}{2}(x^R - x^B)^2 + \tfrac{1}{6}(x^R - 2x^G + x^B)^2}, \tag{8}$$

and $w_{m,n}$ are graph weights selected according to the estimated contour orientations.

The first term of the objective function regularizes the luminance to suppress zipper artifacts while the second term regularizes the chrominance to suppress color artifacts. The parameter $\alpha$ balances between the two regularizers. If $\alpha$ is close to 1, then the result has no noticeable zipper artifacts but has color artifacts. Increasing $\alpha$ reduces color artifacts but increases zipper artifacts.

The weights $w_{m,n}$ are created in an ad-hoc manner through several steps. The intent is that the weights should encourage the result to have contours with the same orientations as what the contour stencils detected. First, the best-fitting mosaicked contour stencils are determined to estimate the contour orientation at each pixel. The orientation estimates are used to assign initial weights $w^{\text{init}}$ between a pixel and its eight adjacent neighbors.




*Weights $w^{init}$ assigned according to the estimated orientation.*

To impose spatial regularity, it is important that the graph is connected (i.e., pixels $m$ and $n$ are connected if $w_{m,n} > 0$ and the graph is connected if there exists a path between any two pixels). We ensure that the graph is connected by adding a positive value $\epsilon$ to the weights of all edges between adjacent pixels,

$$w^{\text{reg}}_{m,n} := \begin{cases} w^{\text{init}}_{m,n} + \epsilon & \text{if } \|m-n\| = 1 \text{ or } \sqrt{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Finally, the graph weights are symmetrized and spatially smoothed with Gaussian filtering,

$$w_{m,n} := \sum_{m'} (w^{\text{reg}}_{m',n} + w^{\text{reg}}_{n,m'}) \exp\!\big(-\tfrac{\|m-m'\|^2}{2\sigma^2}\big). \tag{10}$$

This spatial smoothing reduces the effect of incorrectly detected orientations. Pixels near the image boundaries are handled by extrapolating the weights by whole-sample symmetric extension. The resulting weights are nonzero only for edges between adjacent pixels (no longer edges are created).

# 4 Split Bregman

The minimization is solved efficiently by the split Bregman algorithm [10]. The problem is split by introducing auxiliary variable $d$,

$$\begin{cases} \arg\min\limits_{d,u} \sum\limits_m \Big(\sum\limits_n \big(w_{m,n} d^L_{m,n}\big)^2\Big)^{1/2} + \alpha \sum\limits_m \Big(\sum\limits_n \Big(w_{m,n}\sqrt{(d^{C1}_{m,n})^2 + (d^{C2}_{m,n})^2}\Big)^2\Big)^{1/2} \\ \text{subject to } d_{m,n} = C(u_m - u_n),\ m,n \in \mathbb{Z}^2, \\ \qquad\qquad u^k_m = f_m,\ m \in \Omega^k, k \in \{R,G,B\}, \end{cases} \tag{11}$$

where $C$ denotes the color transformation

$$\begin{pmatrix} x_L \\ x_{C1} \\ x_{C2} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{pmatrix} \begin{pmatrix} x_R \\ x_G \\ x_B \end{pmatrix}.$$

Note that since the weights are nonzero only between adjacent pixels, $d_{m,n}$ is only needed where $m$ and $n$ are adjacent.

The constrained problem is solved by Bregman iteration. In each iteration of the Bregman algorithm, the following unconstrained problem is solved:

$$\begin{aligned} \arg\min_{d,u} \sum_m \Big( \sum_n \big(w_{m,n} d^L_{m,n}\big)^2 \Big)^{1/2} + \alpha \sum_m \Big( \sum_n \Big( w_{m,n} \sqrt{(d^{C1}_{m,n})^2 + (d^{C2}_{m,n})^2} \Big)^2 \Big)^{1/2} \\ + \frac{\gamma_1}{2} \sum_{m,n} \|d_{m,n} - C(u_m - u_n) - b^1_{m,n}\|_2^2 + \frac{\gamma_2}{2} \sum_{k \in \{R,G,B\}} \sum_{m \in \Omega^k} (f_m - u^k_m - b^2_m)^2, \end{aligned} \tag{12}$$

where $\gamma_1$ and $\gamma_2$ are positive parameters and $b^1$ and $b^2$ are variables associated with the Bregman algorithm. This problem is solved by alternatingly solving for $d$ with $u$ fixed and $u$ with $d$ fixed.

**$d$ subproblem** With $u$ fixed, the $d$ variable subproblem is

$$\begin{aligned} \arg\min_{d} \sum_m \Big( \sum_n \big(w_{m,n} d^L_{m,n}\big)^2 \Big)^{1/2} + \alpha \sum_m \Big( \sum_n \Big( w_{m,n} \sqrt{(d^{C1}_{m,n})^2 + (d^{C2}_{m,n})^2} \Big)^2 \Big)^{1/2} \\ + \frac{\gamma_1}{2} \sum_{m,n} \|d_{m,n} - C(u_m - u_n) - b^1_{m,n}\|_2^2. \end{aligned} \tag{13}$$

The problem decouples over $m$ and also decouples between the luminosity channel $L$ and the chromatic channels $C1$ and $C2$. This leads for each $m$ to two subproblems of the form

$$\arg\min_{x \in \mathbb{R}^N} \Big( \sum_{n=1}^N (w_n x_n)^2 \Big)^{1/2} + \frac{\gamma}{2} \sum_{n=1}^N (x_n - y_n)^2. \tag{14}$$

The minimizer of this problem satisfies

$$w_m^2 x_m = \gamma (y_m - x_m) \|x\|_w, \quad \|x\|_w := \Big( \sum_{n=1}^N (w_n x_n)^2 \Big)^{1/2}. \tag{15}$$

We can approximate the solution by fixed point iteration,

$$x_m^{\text{next}} = y_m \frac{\gamma \|x\|_w}{w_m^2 + \gamma \|x\|_w}, \tag{16}$$

where $\|x\|_w$ is computed using the solution from the previous Bregman iteration or, if it is the first iteration or the previous solution was 0, as $\|y\|_w$. Only one iteration of the fixed point iteration needs to be performed per Bregman iteration. The combined effect over multiple Bregman iterations solves the subproblem accurately.

$u$ **subproblem** With $d$ fixed, the $u$ variable subproblem is

$$\arg\min_u \frac{\gamma_1}{2}\sum_{m,n}\|C(u_m-u_n)-d_{m,n}+b^1_{m,n}\|_2^2+\frac{\gamma_2}{2}\sum_{k\in\{R,G,B\}}\sum_{m\in\Omega^k}(u^k_m-f_m+b^2_m)^2. \tag{17}$$

The minimizer of this problem satisfies

$$\gamma_1\sum_n C^*\big(2C(u_m-u_n)-(d_{m,n}-b^1_{m,n})+(d_{n,m}-b^1_{n,m})\big)+\gamma_2 e_m(e_m\cdot u_m-f_m+b^2_m)=0, \tag{18}$$

where $C^*$ denotes the adjoint of $C$ and $e_m$ is $(1,0,0)^T$, $(0,1,0)^T$, $(0,0,1)^T$ respectively at red, green, and blue locations. We solve this equation by one sweep of block Gauss–Seidel per Bregman iteration,

$$\begin{aligned}&(2\cdot 8\gamma_1 C^*C+\gamma_2 e_m e_m^T)u_m^{\text{next}}=\\&\quad\gamma_1\sum_n C^*\big(2Cu_n+(d_{m,n}-b^1_{m,n})-(d_{n,m}-b^1_{n,m})\big)+\gamma_2 e_m(f_m-b^2_m).\end{aligned} \tag{19}$$

On the right hand side, the factor 8 is a consequence of each pixel having 8 adjacent neighbors. The matrices

$$(2\cdot 8\gamma_1 C^*C+\gamma_2 e_m e_m^T) \tag{20}$$

have size $3\times 3$ and their inverses can be precomputed. There are three such matrices, and which matrix is used depends on whether $m$ is a red, green, or blue pixel location.

The overall minimization algorithm is

Initialize $d=b^1=0$, $b^2=0$
**while** $\|u^{cur}-u^{prev}\|_2 > tol\|f\|_2$ **do**
  Solve the $u$ subproblem
  Solve the $d$ subproblem
  $b^1_{m,n}=b^1_{m,n}+C(u_m-u_n)-d_{m,n}$
  $b^2_n=b^2_n+u^k_n-f_n,\quad n\in\Omega^k$

**Algorithm 1**

Bilinear demosaicking is used as the initial solution for $u$. The stopping condition is according to the sum squared difference between the current and previous iteration's solution of $u$. The algorithm converges to the same solution for any positive $\gamma_1$ and $\gamma_2$, though the choice of these parameters does affect the convergence speed. In the examples, we use $\gamma_1=4$ and $\gamma_2=256$. The computational complexity per Bregman iteration is linear in the number of pixels.

# 5 Implementation

In the implementation, $b^1$ is replaced by $\tilde{d}=d-b^1$, and the minimization algorithm is

Initialize $d=\tilde{d}=0$, $b=0$
**while** $\|u^{cur}-u^{prev}\|_2 > tol\|f\|_2$ **do**
  Solve the $u$ subproblem
  Solve the $d$ subproblem
  $\tilde{d}_{m,n}=\tilde{d}_{m,n}-C(u_m-u_n)+d_{m,n}+\Delta d_{m,n}$
  $b_n=b_n+u^k_n-f_n,\quad n\in\Omega^k$

**Algorithm 2**

where $\Delta d := d^{\text{cur}}-d^{\text{prev}}$ is the change in $d$ since the previous iteration. This version is algebraically equivalent to the algorithm with $b^1$ but eliminates some computations in solving the subproblems.

# 6 Examples

In the examples, the parameters used are $\alpha = 1.8$, $\epsilon = 0.15$, $\sigma = 0.6$ and $tol = 0.001$. With these parameters, around 40 iterations are needed for convergence on most images. To protect against the possibility of an infinite loop, the implementation additionally sets a limit of 250 Bregman iterations, but this limit should never be reached with these parameters.

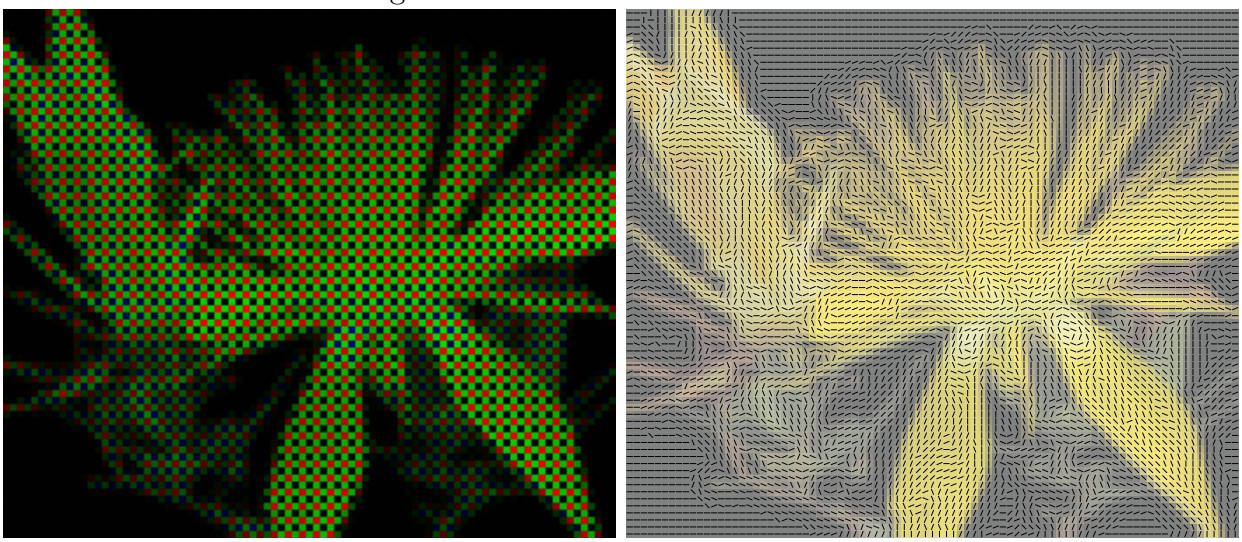
The proposed demosaicking is computationally expensive compared to existing methods. In a serial implementation on a recent laptop (2.40GHz Intel® T770 with 2GB RAM), the computation time on a $512 \times 512$ image is 0.1 s per iteration or about 4 s total. The majority of the time is spent solving the graph-regularized optimization problem. The orientation estimation with mosaicked contour stencils itself is very fast, with a run time of 0.025 s on a $512 \times 512$ image.

To avoid boundary effects, each border of the input image is padded by 16 pixels using whole-sample symmetric extension. This extension can be computed directly on the mosaicked data because the Bayer pattern is whole-sample symmetric about any row or column. This padding is then trimmed from the demosaicking result. Most examples below show the mean squared error (MSE) between the demosaicking result and the original image (the padding is not included in the MSE computation). This methodology is also applied in the online demo.

## 6.1 Contour Estimation

In the figure below, contour stencils are applied to a mosaicked image to estimate the contour orientations. For each pixel, the orientation modeled by the best-fitting stencil $\mathcal{S}^\star$ is displayed superimposed on the original image.

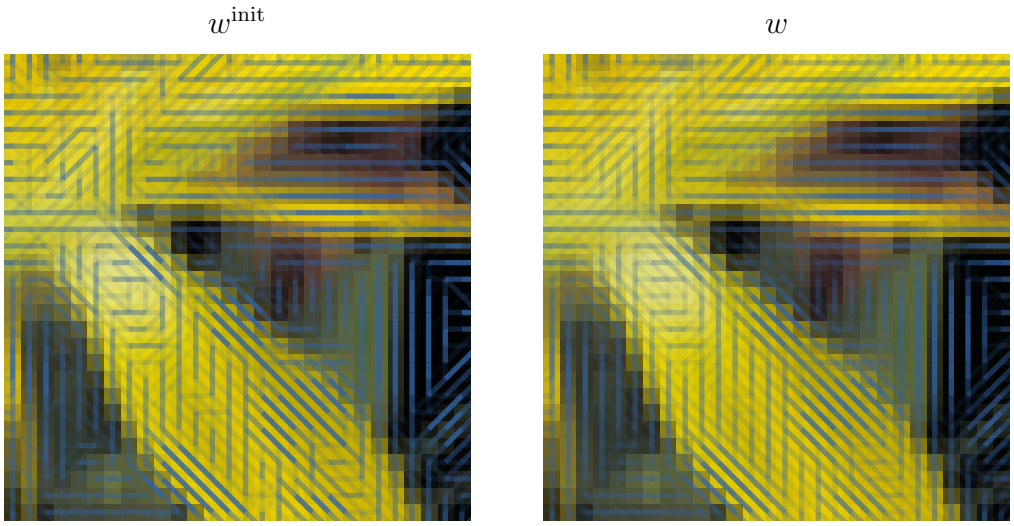
Observed Image | Estimated Contours



*Contours estimated from the mosaicked image.*

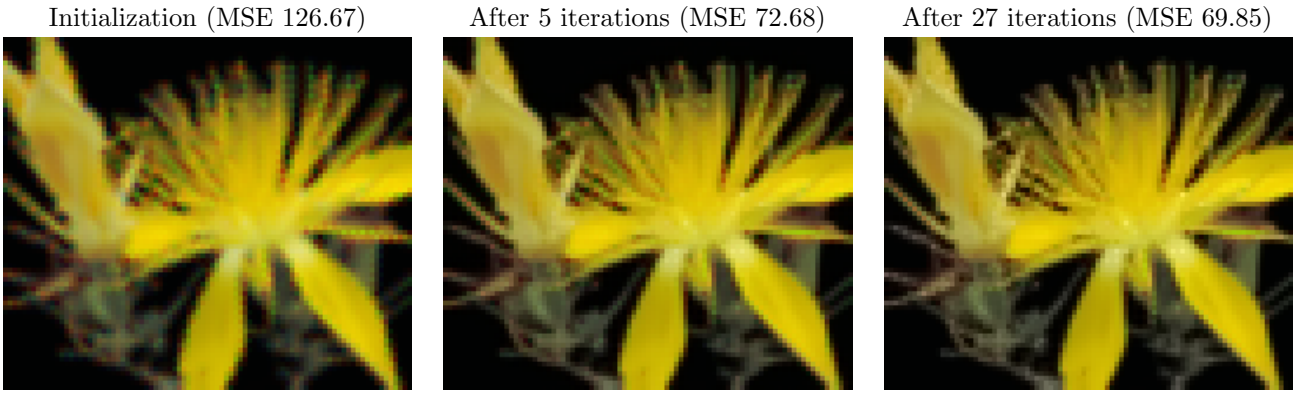
## 6.2 Weighted Graph Construction

The contours estimated by the contour stencils are used to construct a weighted graph. The initial weights $w^{\text{init}}$ and the final weights $w$ after adding $\epsilon$ and Gaussian filtering are shown over a small region of the image below.



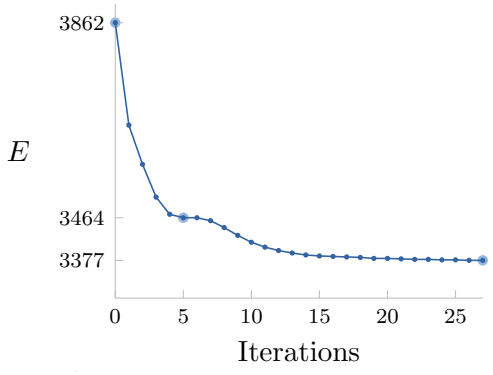

## 6.3 Demosaicking Minimization

We now show the contour stencil demosaicking, which solves the graph-regularized minimization problem using the graph weights $w$. The following figure shows the initialization (bilinear demosaicking) and the minimization result after 5 and 27 Bregman iterations. After 27 iterations, the minimization converges with tolerance $tol = 0.001$.




*Evolution of the demosaicking minimization.*

The energy is decreased with each Bregman iteration. The energy is initially 3862, reducing to 3464 after 5 iterations, and converging to 3377 after 27 iterations.

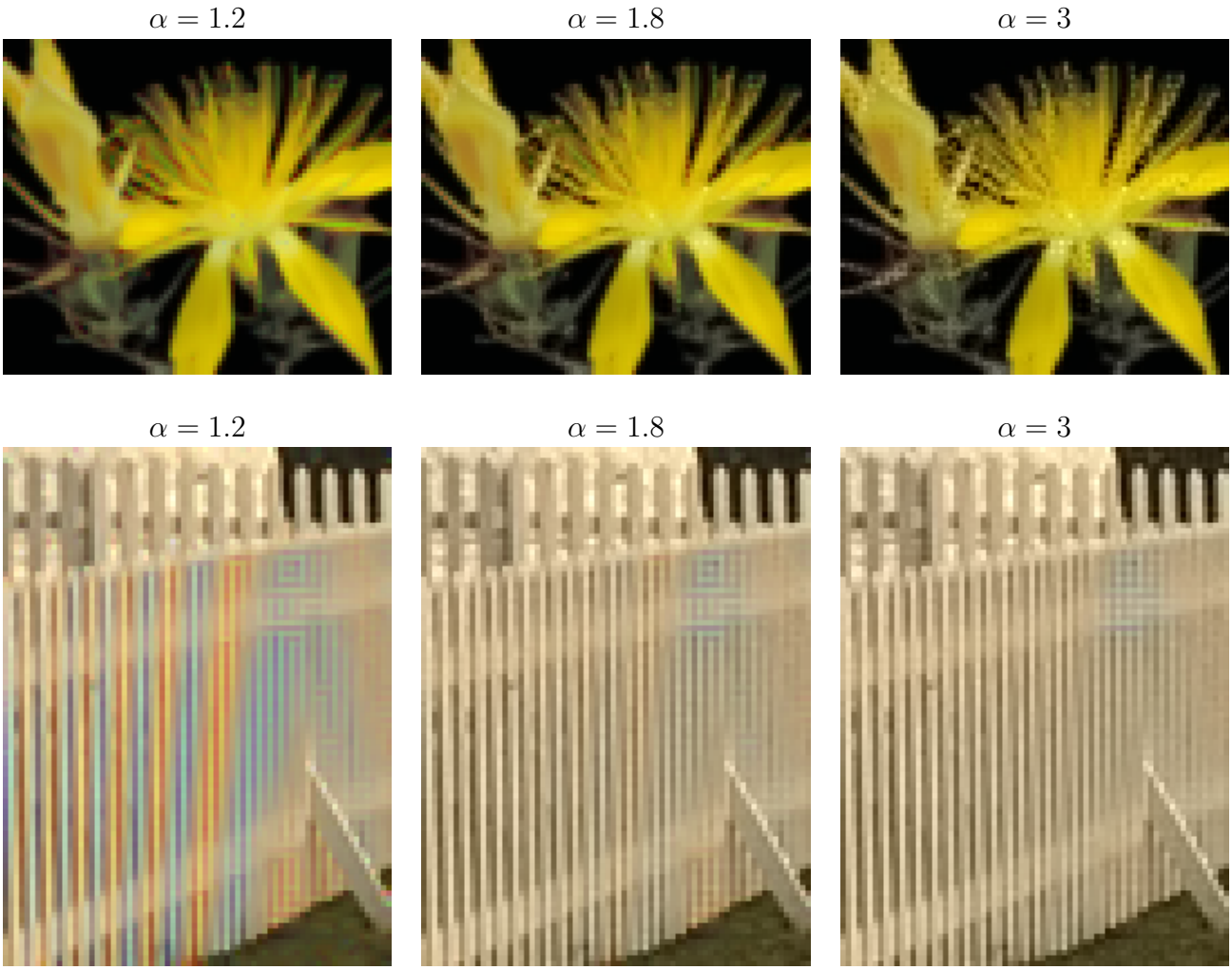



*Energy value vs. Bregman iterations.*

## 6.4 Effect of $\alpha$

Demosaicking involves a balance of avoiding two different undesirable effects, color artifacts and zipper artifacts. *Color artifacts* are the appearance of artificial colors. *Zipper artifacts* are unnatural patterns of alternating bright and dark pixels. These effects usually manifest on thin structures and near edges, where interpolation is most difficult. Demosaicking methods that completely avoid one artifact usually suffer heavily from the other. For example, the demosaicking methods of Gunturk et al. [3] and Li [4] explicitly remove color artifacts but suffer significant zipper artifacts, and on the other extreme, bilinear demosaicking has no zipper artifacts but significant color artifacts. So instead, many demosaicking methods manage a balance of these two artifacts.

In contour stencil demosaicking, the parameter $\alpha$ explicitly controls the balance between color artifacts and zipper artifacts. This figure shows the contour stencil demosaicking result for three different values of $\alpha$.

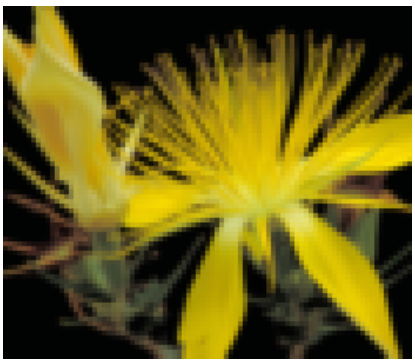



Smaller $\alpha$ leads to stronger color artifacts while larger $\alpha$ leads to more zipper artifacts. The optimal $\alpha$ depends on the image. For the images above, a smaller $\alpha$ is better on the flower while a larger $\alpha$ is better on the fence. We fix $\alpha = 1.8$ in the subsequent examples, which is a good setting for most images.

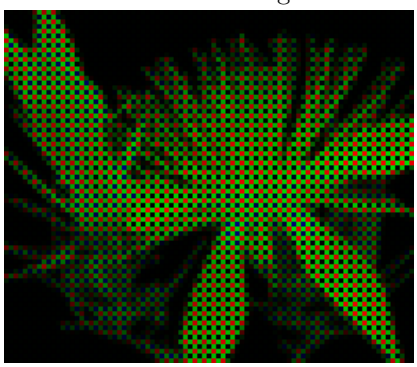
## 6.5 Comparison with Existing Methods

The figure hereafter compares the method with several existing demosaicking methods.

Exact



Observed Image

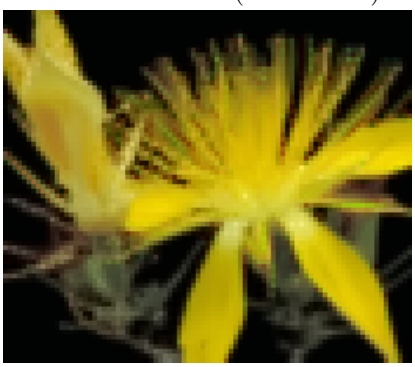


Zhang–Wu [5] (MSE 117.93)

Chung–Chan [6] (MSE 105.36)

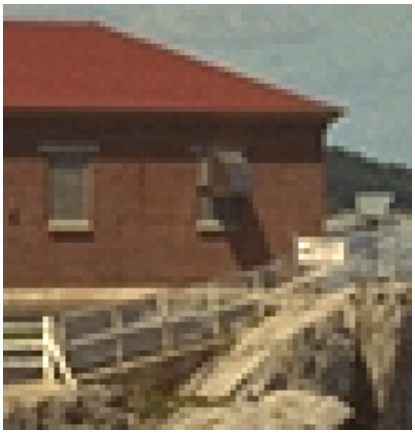
SSD [11] (MSE 88.89)

Contour Stencils (MSE 69.85)



The next figure compares the methods on a crop of image 21 of the Kodak Image Suite. The fence is a difficult feature in this image, where artifacts are visible with most of the methods.

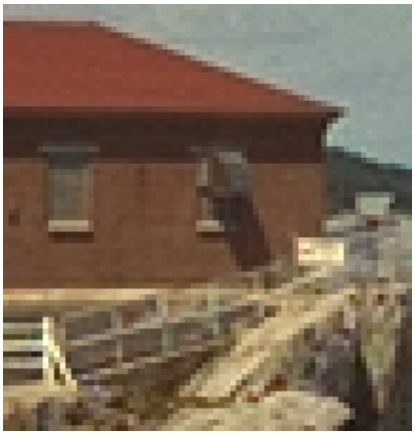
Exact



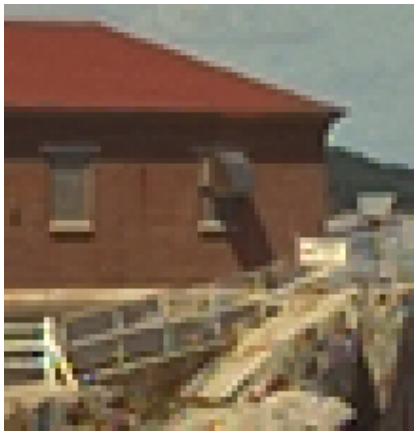
Observed Image

Zhang–Wu [5] (MSE 14.22)

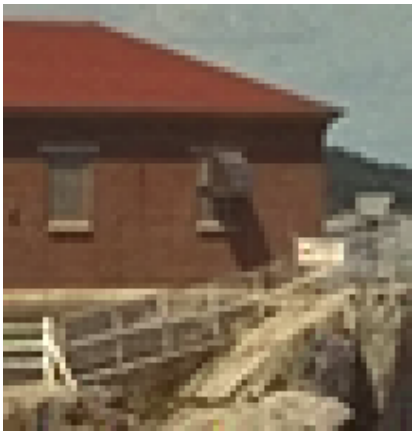
Chung–Chan [6] (MSE 14.22)



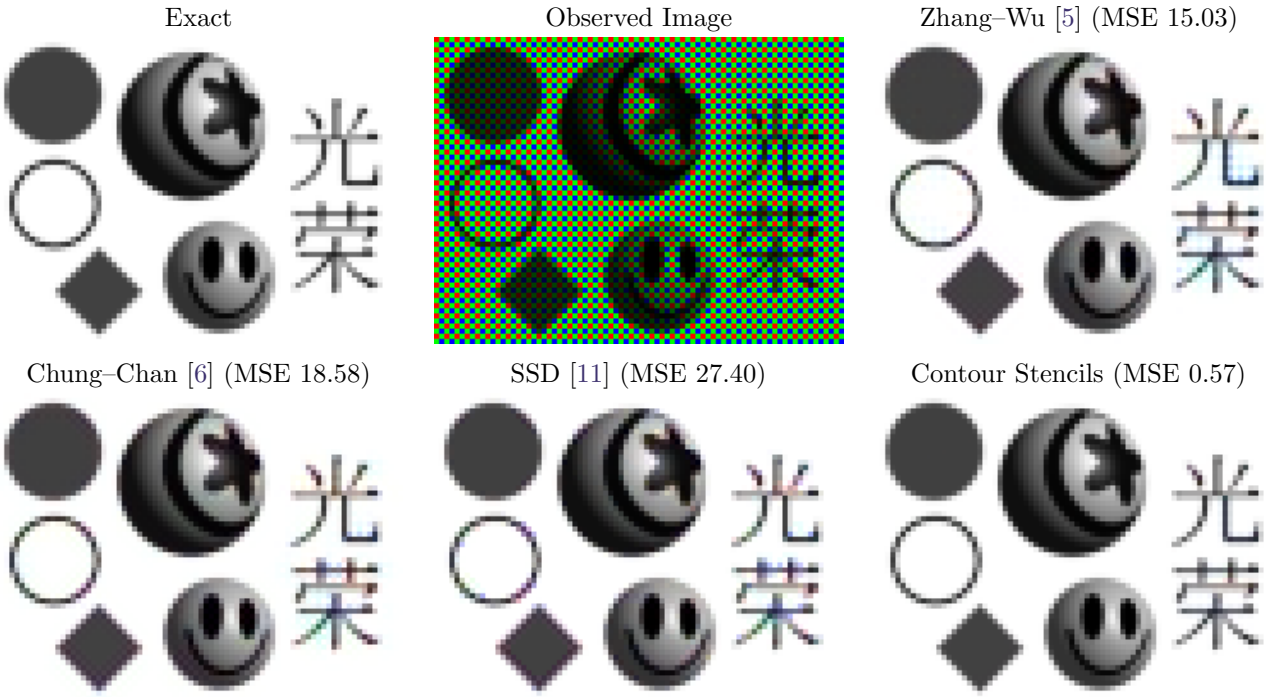
SSD [11] (MSE 23.35)



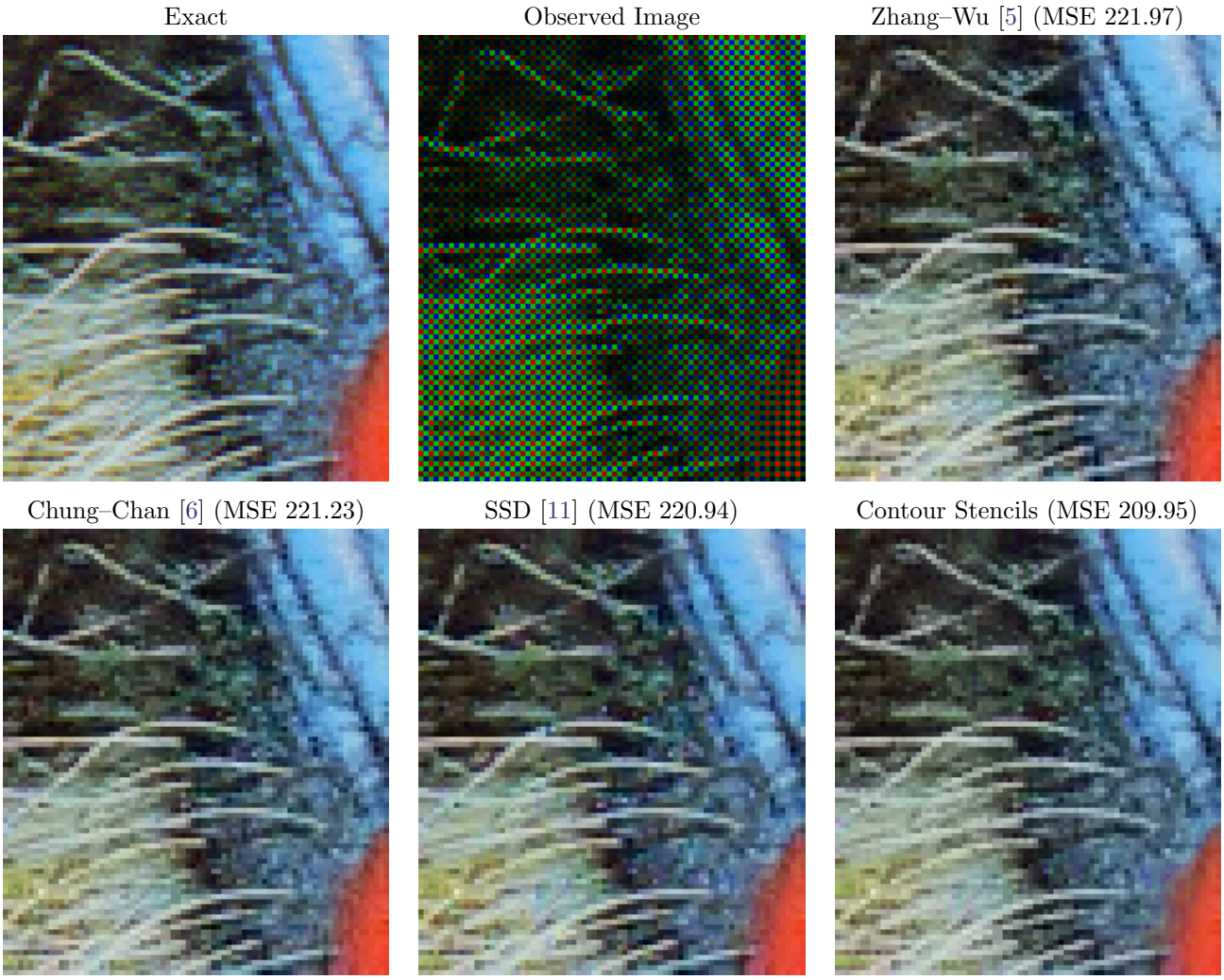
Contour Stencils (MSE 12.89)



The next example tests demosaicking on a gray synthetic image containing different geometric structures. Contour stencils demosaicking is very successful on this image.




The last example compares the demosaicking methods on a crop from the standard mandrill image. This image is extremely challenging due to the fine structure of the whiskers and strong color contrast.




## Acknowledgments

This material is based upon work supported by the National Science Foundation under Award No. DMS-1004694. Work partially supported by the Office of Naval Research under grant N00014-97-1-0839 and by the European Research Council, advanced grant "Twelve labours."

## Image Credits

 Adonis Blazingstar, Thomas Barnes, U.S. Fish and Wildlife Service, Public Domain (http://digitalmedia.fws.gov/cdm4/item_viewer.php?CISOROOT=/natdiglib&CISOPTR=8158)

 Kodak Image Suite, image 19 (http://r0k.us/graphics/kodak/)

 Kodak Image Suite, image 21 (http://r0k.us/graphics/kodak/)

 Pascal Getreuer

 Mandrill standard test images

# References


[1] B. E. Bayer, "Color imaging array," U.S. Patent 3971065, 1976.

[2] J. F. Hamilton, Jr. and J. E. Adams, Jr., "Adaptive color plan interpolation in single sensor color electronic camera," U.S. Patent 5629734, 1997.

[3] B. K. Gunturk, Y. Altunbasak, and R. M. Mersereau, "Color plane interpolation using alternating projections," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 997–1013, 2002. http://dx.doi.org/10.1109/TIP.2002.801121

[4] X. Li, "Demosaicing by successive approximation," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 370–379, 2005. http://dx.doi.org/10.1109/TIP.2004.840683

[5] L. Zhang and X. Wu, "Color demosaicking via directional linear minimum mean square-error estimation," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2167–2178, 2005. http://dx.doi.org/10.1109/TIP.2005.857260

[6] K.-H. Chung and Y.-H. Chan, "Color demosaicing using variance of color differences," *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 2944–2955, 2006. http://dx.doi.org/10.1109/TIP.2006.877521

[7] P. Getreuer, "Contour stencils for edge-adaptive image interpolation," *Proceedings of SPIE*, vol. 7257, 2009. http://dx.doi.org/10.1117/12.806014

[8] P. Getreuer, "Image zooming with contour stencils," *Proceedings of SPIE*, vol. 7246, 2009. http://dx.doi.org/10.1117/12.805934

[9] L. Condat, "A new color filter array with optimal sensing properties," *Proceedings of the IEEE International Conference on Image Processing*, 2009. http://dx.doi.org/10.1109/ICIP.2009.5414383

[10] T. Goldstein and S. Osher, "The Split Bregman Method for L1 Regularized Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 323–343, 2009. http://dx.doi.org/10.1137/080725891

[11] A. Buades, B. Coll, J.-M. Morel, C. Sbert, "Self-Similarity Driven Demosaicking," *Image Processing On Line*, 2011. DOI: http://dx.doi.org/10.5201/ipol.2011.bcms-ssdd.

[12] P. Getreuer, "Image Interpolation with Contour Stencils," *Image Processing On Line*, 2011. DOI: http://dx.doi.org/10.5201/ipol.2011.g_iics.

[13] P. Getreuer, "Color Demosaicing with Contour Stencils," *Proceedings of the 17th International Conference on Digital Signal Processing*, 2011. http://dx.doi.org/10.1109/ICDSP.2011.6004885