# Fixed-Coefficient Iterative Bilateral Filters for Graph-Based Image Processing

Chang Jian, Kohei Inoue, Kenji Hara, and Kiichi Urahama

Kyushu University, Fukuoka 815-8540, Japan
{jian,k-inoue,hara,urahama}@design.kyushu-u.ac.jp

**Abstract.** We present a graph-based image processing algorithm using fast iterative bilateral filters. The computation of bilateral filters is accelerated with fixation of the coefficients during iterations and their approximate decomposition further speeds up the computation. We show that this fixed-coefficient iterative bilateral filter is an alternative solver for optimization problems in graph-based data analyses and apply its fast algorithm to graph-based image processing tasks. Performance of the present algorithm is demonstrated with experiments of contrast enhancement and smoothing of images using cross bilateral filters, in addition to semi-supervised image segmentation and colorization of monochromatic images.

## 1 Introduction

Graph-based learning algorithms and multivariate data analysis techniques[1] have become widely used in the fields of pattern recognition and computer vision[2]. Graph-based techniques are usually formulated by optimization problems of which iterative solution generally demands long computational time. Images and videos are typical examples of such large scale data.

In image processing, many iterative algorithms for solving optimization problems have also been used for various tasks such as segmentation and noise reduction by anisotropic diffusion[3]. The bilateral filter[4] has been developed for a one-step filtering without iterations for smoothing of images. In practice, however, it is applied to an image repeatedly if one-step smoothing is insufficient.

Though these graph-based algorithms and iterated filters have similar forms, they have been developed almost independently in different fields, therefore their relationship has not been noticed explicitly.

In this paper, we fix the weighting coefficients in the bilateral filter in order to accelerate its computational speed and show it is another iterative algorithm in addition to a graph-based algorithm for solving the common optimization problem. From this observation, we present a fast algorithm for graph-based image processing using an iterative filter with approximately decomposed weighting coefficients.

T. Wada, F. Huang, and S. Lin (Eds.): PSIVT 2009, LNCS 5414, pp. 473–484, 2009.

## 2 Graph-Based Data Analysis

A graph is composed of nodes linked with edges. We deal with only undirected graphs with symmetric edges in this paper. In this section, we review a principal graph-based technique for analyzing graphs with weighted edges and extend it to the graphs in which nodes are also weighted.

### 2.1 Laplacian Eigenmaps

Let there be given a set of $n$ data between which the similarity is denoted by $s_{ij}$ which is usually expressed by $s_{ij} = e^{-\alpha\|d_i - d_j\|^2}$ where $d_i$ is the feature vector of datum $i$. Such a dataset is called the similarity data and represented by an undirected graph in which the edge weight is $s_{ij}$. A fundamental procedure in graph-based approaches is embedding of data into a low dimensional space. In order to preserve the topology of data, mutually similar data are mapped to mutually close places of which coordinate $x_i$ is given by

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij}(x_i - x_j)^2 \tag{1}$$

of which optimal solution is computed with an iterative algorithm:

$$x_i^{(\xi+1)} = \sum_{j=1,\neq i}^{n} s_{ij}x_j^{(\xi)} / \sum_{j=1,\neq i}^{n} s_{ij} \tag{2}$$

where $\xi$ is an iteration counter. Notice that $i$ is excluded from $\sum_j$. This iteartive algorithm is the Jacobi method for solving the system of linear equations $(D' - S')x = 0$ where $S' = [s'_{ij}]$; $s'_{ii} = 0, s'_{ij} = s_{ij}$ $(i \neq j)$, $D' = \mathrm{diag}(d'_1, ..., d'_n)$; $d'_i = \sum_j s'_{ij}$. This algorithm is also the power method for computing the eigenvector of the stochastic matrix $(D')^{-1}S'$ where $x^{(\xi)}$ converges to a constant vector $[c, ..., c]^T$ which is the eigenvector of $(D')^{-1}S'$ with the maximal eigenvalue 1. This vector coincides with the eigenvector of the Laplacian matrix $L' = D' - S'$ with the minimal eigenvalue 0.

Whereas this principal eigenvector is discarded in the Laplacian eigenmaps[5] because the constant vector contains no information for discriminating data, this vector gives us useful information in the midway of the iteration before its convergence. As will be described below in this paper, the principal eigenvector on the midway of the iteration is outputted as a smoothed result in the iterated filtering of images.

### 2.2 Semi-supervised Clustering

The spectral clustering method[1] is an unsupervised learning algorithm where data are partitioned into several clusters in the low dimensional space mapped with the Laplacian eigenmap described above.

Let us next consider semi-supervised clustering methods where the membership is known for some data in advance of the learning. We deal here with the simplest case of bi-partitioning, i.e. partitioning data into two clusters. Its extension to multiple clusters is straightforward. Let $(x_i + 1)/2$ represent the membership in the first cluster, that is, if $x_i = 1$ datum $i$ belongs to the first cluster, conversely if $x_i = -1$ it belongs to the second cluster. Some data $i \in T_1$ are known to belong to the first cluster and data $i \in T_2$ belong to the second cluster. Based on this knowledge, we estimate $x_i$ of the remaining data $i \notin \{T_1, T_2\}$. A popular algorithm for this task is the semi-supervised learning by label propagation[6] of which iterative algorithm is eq.(2). The value of $x_i$ for $i \in \{T_1, T_2\}$) is fixed throughout the iteration.

### 2.3 Graph with Weighted Nodes

In the graph treated above, only the edges are weighted, while nodes are weightless, i.e. every node has weight 1. If nodes have weights $w_i$ in addition to edges, eq.(1) becomes

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j s_{ij}(x_i - x_j)^2 \tag{3}$$

of which solution can be computed with the iterative algorithm similar to eq.(2):

$$x_i^{(\xi+1)} = \sum_{j=1,\neq i}^{n} w_j s_{ij} x_j^{(\xi)} / \sum_{j=1,\neq i}^{n} w_j s_{ij} \tag{4}$$

which is also used for the label propagation in the graphs with weighted nodes and edges.

## 3 Bilateral Filter

The bilateral filter (hereinafter abbreviated as BF) has been widely used for image smoothing such as abstract stylization and noise reduction. We deal with grayscale images for simplicity. Let the graylevel of pixel $(i, j)$ be $d_{ij}$. The output of BF with the window $[-p, p] \times [-p, p]$ is given by

$$f_{ij} = \sum_{k=-p}^{p}\sum_{l=-p}^{p} s_{ijkl} d_{i+k,j+l} / \sum_{k=-p}^{p}\sum_{l=-p}^{p} s_{ijkl} \tag{5}$$

where $s_{ijkl} = e^{-\alpha(k^2+l^2)-\beta(d_{ij}-d_{i+k,j+l})^2}$ which has the same expression as the similarity in the above graph-based data analyses. In practice, however, only one-step filtering of an image with BF often results in insufficient smoothing. For such cases, BF is applied repeatedly as

$$f_{ij}^{(\xi+1)} = \sum_{k=-p}^{p}\sum_{l=-p}^{p} s_{ijkl}^{(\xi)} f_{i+k,j+l}^{(\xi)} / \sum_{k=-p}^{p}\sum_{l=-p}^{p} s_{ijkl}^{(\xi)} \tag{6}$$

where $s_{ijkl}^{(\xi)} = e^{-\alpha(k^2+l^2)-\beta(f_{ij}^{(\xi)}-f_{i+k,j+l}^{(\xi)})^2}$. The initial value of $f_{ij}$ is set to the graylevel in the input image as $f_{ij}^{(0)} = d_{ij}$. The following proposition holds for this iterated BF:

[Proposition 1] Eq.(6) is an iterative solution algorithm for the optimization problem

$$\max \quad \sum_{i,j} \sum_{k=-p}^{p} \sum_{l=-p}^{p} e^{-\alpha(k^2+l^2)-\beta(f_{ij}-f_{i+k,j+l})^2} \tag{7}$$

with a decelerated Jacobi method.

Since we do not experiment this iterated BF in this paper, the proof of this proposition is omitted. Also the proof of the monotonic increase in the objective function in eq.(7) through the iteration is omitted.

### 3.1 Fixed-Coefficient Iterative BF

This iterated BF takes long computational time because $s_{ijkl}^{(\xi)}$ must be updated at every iteration step. To alleviate this computational cost, we fix $s_{ijkl}^{(\xi)}$ to its initial value $s_{ijkl} = e^{-\alpha(k^2+l^2)-\beta(d_{ij}-d_{i+k,j+l})^2}$, then eq.(6) becomes

$$f_{ij}^{(\xi+1)} = \sum_{k=-p}^{p} \sum_{l=-p}^{p} s_{ijkl} f_{i+k,j+l}^{(\xi)} / \sum_{k=-p}^{p} \sum_{l=-p}^{p} s_{ijkl} \tag{8}$$

We call this algorithm the fixed-coefficient iterative BF (FCIBF). This algorithm can be implemented faster than the iterated BF by computing and saving $s_{ijkl}$ before starting the iteration and using it during the iteration. For instance, this FCIBF is about 5-times faster than the iterated BF in their 20 iterations for a $500 \times 500$ image.

This FCIBF resembles eq.(2), that is, we denote every pixel by nodes and link the nodes $(i, j)$ and $(i + k, j + l)$ in the window with an edge of the weight $s_{ijkl}$, then we can manipulate the FCIBF graph-theoretically. In the Markov random field (MRF), only 4 or 8 nearest-neighbor pixels are linked together, whereas edges are drawn between every pair of pixels in the window $\{(k+i, j+l);\ -p \le k \le p, -p \le l \le p\}$ in the FCIBF. Slight difference between eq.(2) and eq.(8) is that the righthand-side in eq.(8) includes $f_{ij}^{(\xi)}$ but eq.(2) does not. This difference is stated as eq.(2) is the Jacobi method while eq.(8) is a decelerated Jacobi method:

[Proposition 2] Eq.(8) is an iterative solution algorithm for eq.(8) for the optimization problem

$$\min \quad \sum_{i,j} \sum_{k=-p}^{p} \sum_{l=-p}^{p} s_{ijkl}(f_{ij} - f_{i+k,j+l})^2 \tag{9}$$

with a decelerated Jacobi method.

(Proof) The iterant in the Jacobi method for eq.(9) is

$$\tilde{f}_{ij}^{(\xi+1)} = \sum_{k,l}{}' s_{ijkl} f_{i+k,j+l}^{(\xi)} / \sum_{k,l}{}' s_{ijkl} \tag{10}$$

where $\sum_{k,l}{}'$ denotes the summation excluding $\{k=0, l=0\}$ from $\sum_{k=-p}^{p}\sum_{l=-p}^{p}$. Eq.(10) coincides with eq.(2). By denoting $\mu_{ij} = \sum_{k,l}{}' s_{ijkl} / \sum_{k=-p}^{p}\sum_{l=-p}^{p} s_{ijkl}$, we can express the relationship between $f_{ij}^{(\xi+1)}$ in eq.(8) and $\tilde{f}_{ij}^{(\xi+1)}$ in eq.(10) as

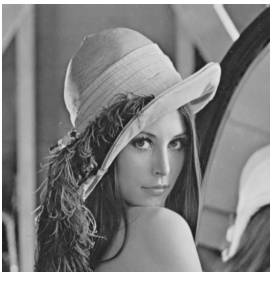
$$f_{ij}^{(\xi+1)} = \mu_{ij}\tilde{f}_{ij}^{(\xi+1)} + (1-\mu_{ij}) f_{ij}^{(\xi)} \tag{11}$$

which states that $f_{ij}^{(\xi+1)}$ is an interior division between $f_{ij}^{(\xi)}$ and $\tilde{f}_{ij}^{(\xi+1)}$ and is a point pulled back from the Jacobi iterant $\tilde{f}_{ij}^{(\xi+1)}$ directed to the previous iterant $f_{ij}^{(\xi)}$. Thus eq.(8) is a decelerated Jacobi method. (Q.E.D.)

The convergence of eq.(8) is therefore slower than the Jacobi method of eq.(10). The distance of pulling back of $\tilde{f}_{ij}^{(\xi+1)}$ is however short, hence $\mu_{ij} \cong 1$ and the difference between their convergence rates is small. We verify this observation with a $500 \times 500$ image in Fig.1 where the original image is shown on the left and it added with Gaussian noises of standard deviation 40 is shown on the right.

Firstly the variation in the value of eq.(9) for Fig.1(a) is shown in Fig.2 where the solid line denotes the FCIBF of eq.(8) and the dotted line is the Jacobi method (JM) of eq.(10). We set $\alpha = 0.001, \beta = 0.01$ and $p = 2$. The value of $E$ in the FCIBF is slightly larger than that of JM, that is, the convergence of FCIBF is slightly slower than JM. As is expected, however, this difference can be observed only when the window is narrow and becomes negligible for the window of $p$ larger than 5. The convergence is theoretically ensured for eq.(8) and eq.(10) since both the matrices $D^{-1}S$ and $(D')^{-1}S'$ are irreducively diagonal dominant.

Nextly, the PSNR(Peak Signal-to-Noise Ratio) between the iterated outputs of iterative filtering of Fig.1(b) and the noise-free Fig.1(a) is shown in Fig.3. The PSNR of FCIBF is slightly greater than JM, that is, the noise reduction



(a) original image

(b) image with noise

**Fig. 1.** Input image

capability of FCIBF is slightly higher than JM. This superiority of FCIBF may be attributed to that FCIBF is the average of more pixels than JM because FCIBF includes the central pixel of the window while JM does not.

The concave variation in the PSNR in Fig.3 is commonly observed. In the early steps in the iteration, the PSNR increases because added noises are reduced by smoothing and the PSNR reaches its maximum after which it turns to decrease by over-smoothing of the original image components in addition to noises. It is the best to stop the iteration at the peak of the PSNR, but the automatic determination of the stopping time is difficult and beyond the scope of this paper.

Notice furthermore that eq.(8) is also an adaptive steepest descent method besides the decelerated Jacobi method. Since the derivative of the function $E$ in eq.(9) is $\partial E/\partial f_{ij} = \sum_{k,l}{}' s_{ijkl}(f_{ij} - f_{i+k,j+l})$, the steepset descent iteration $f_{ij}^{(\xi+1)} = f_{ij}^{(\xi)} - h_{ij}\partial E/\partial f_{ij}^{(\xi)}$ reduces to eq.(8) by setting the step-length as $h_{ij} = 1/\sum_{k=-p}^{p}\sum_{l=-p}^{p} s_{ijkl}$.

Consequently, though there is slight difference between eq.(8) and eq.(10), they are alternative algorithms for solving the same optimization problem. Therefore the iterated filter of eq.(8) can be applied to the graph-based data analysis in section 2. This exploitation of eq.(8) for graph-based image processing is beneficial since eq.(8) can be executed fast using an approximated algorithm as will be described below.

### 3.2 Fixed-Coefficient Iterative Cross BF

In the above BF, the input is a single image and the coefficient $s_{ij}$ is calculated from the input image itself. On the other hand in the graph-based method in section 2, the edge weight $s_{ij}$ is calculated from the feature vector $d_i$ different from the variable $x_i$ which is the target of processing of the algorithms.

The same situation also appears in BF where the coefficient $s_{ij}$ is calculated from an augmented image different from an input image which is the target of processing. This BF utilizing an augmented image is called the cross BF[7]. The

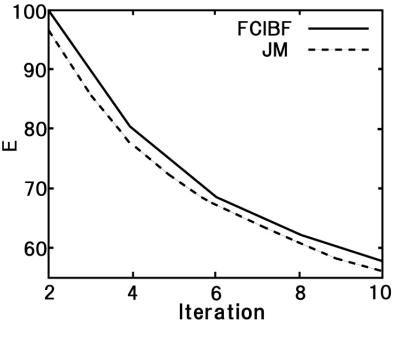



**Fig. 2.** Variantion in the objective function $E$ in eq.(9)

above trick of fixing the filter coefficients during the iteration is also available in the cross BF leading to the fixed-coefficient iterative cross BF (FCICBF) where the coefficient in eq.(8) is calculated from the pixel value $e_{ij}$ in an augmented image as $s_{ij} = e^{-\alpha(k^2+l^2)-\beta(e_{ij}-e_{i+k,j+l})^2}$.

### 3.3 Fast Algorithm with Approximated Decomposition of Coefficients

By approximately decomposing the coefficient $s_{ijkl}$ in eq.(8) into the product of the component along the $k$ direction and that along the $l$ direction as $e^{-\alpha k^2-\beta(d_{ij}-d_{i+k,j})^2} e^{-\alpha l^2-\beta(d_{i+k,j}-d_{i+k,j+l})^2}$, we can implement a fast procedure for the FCIBF in eq.(8) (its detailed derivation is omitted) as:

[Construction of arrays]
Step 1) We calculate $u_{ijk} = e^{-\alpha k^2-\beta(d_{ij}-d_{i+k,j})^2}$ for every $i, j, k$, and also $v_{ijl} = e^{-\alpha l^2-\beta(d_{ij}-d_{i,j+l})^2}$ for all $i, j, l$, and save them in 3-dimensional arrays.

Step 2) We calculate $b_{ij} = \sum_{l=-p}^{p} v_{ijl}$ for all $i, j$.

Step 3) We calculate $t_{ij} = \sum_{k=-p}^{p} u_{ijk} b_{i+k,j}$ for all $i, j$ and save it in a 2-dimensional array.

[Iteration of Filter]
Step 4) For all $i, j$, we calculate

$$a_{ij} = \sum_{l=-p}^{p} v_{ijl} f_{i,j+l}^{(\xi)}, \quad s_{ij} = \sum_{k=-p}^{p} u_{ijk} a_{i+k,j} \tag{12}$$

from which we compute $f_{ij}^{(\xi+1)} = s_{ij}/t_{ij}$ and repeat this computation.

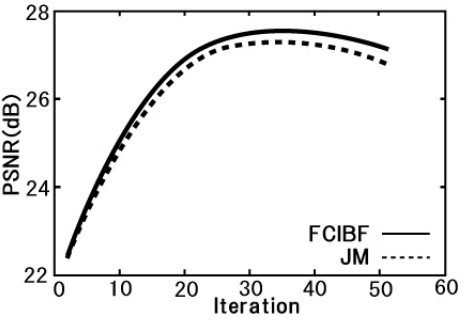
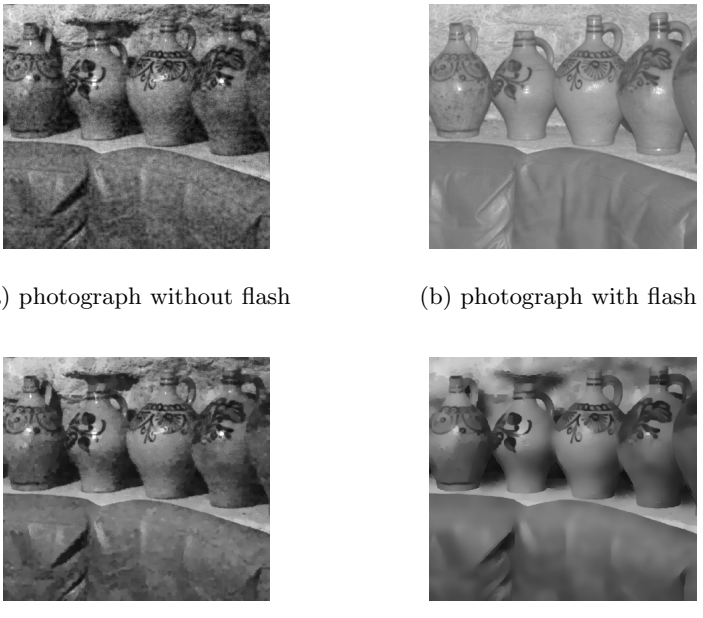



**Fig. 3.** Variation in PSNR

(a) input image (b) output image

**Fig. 4.** Contrast enhancement of under-exposed image



(a) photograph without flash (b) photograph with flash

(c) result of FCIBF (d) result of FCICBF

**Fig. 5.** Example of cross BF

For instance, this algorithm is about 6-times faster than the FCIBF in eq.(8) in its 20 iterations for a $500 \times 500$ image. We use this fast algorithm in the following experiments.

The first experiment is an application of the FCIBF to the enhancement of the contrast of an under-exposed image in Fig.4(a) of the size $319 \times 284$ of which pixel value is denoted as $d_{ij}$. We set $p = 5, \alpha = 0.01, \beta = 0.025$ and extract the pixel value $e_{ij}$ of the second output of the FCIBF, 30-th output $f_{ij}$,

and 100-th output $g_{ij}$. By combining these outputs as $h_{ij} = w_1 g_{ij} + w_2(f_{ij} - g_{ij}) + w_3(e_{ij} - f_{ij}) + w_4(d_{ij} - e_{ij})$, we obtain a contrast-enhanced image by normalizing $h_{ij}$ to [0,255]. The resultant image is shown in Fig.4(b) where we set $w_1 = 0.2, w_2 = 0.2, w_3 = 0.2, w_4 = 0.4$. The dark areas in Fig.4(a) where detail textures cannot be discriminated become visible more clearly in Fig.4(b). The computational time is 29.5 seconds for the FCIBF in eq.(8) which is reduced to 7.0 seconds using the fast algorithm described above.

Next experiment is an application of the FCICBF in section 3.2. We smooth the image in Fig.5(a) using an augmented image in Fig.5(b) for computing the filter coefficients. The size of both images is $400 \times 365$. Fig.5(a) is an ordinary photograph without flash lighting where the color of objects is faithfully captured but is dark and noisy. Contrastively object edges are clear and noise level is low in the photograph with flash in Fig.5(b) where, however, the color of every object is whitened. We set $p = 5, \alpha = 0.1, \beta = 0.02$ and iterate the filters 10 times. Fig.5(c) is a result of the fast FCIBF applied to Fig.5(a) solely. Noises are remained in Fig.5(c). Next Fig.5(d) is a result of the fast FCICBF where major edges are preserved and noises are satisfactorily reduced. Their computational time is 8.8 seconds for the naive FCIBF and is 4.3 seconds with the fast FCIBF.

## 4 Semi-supervised Image Processing

All of the above experiments are the examples of unsupervised image processing. We turn into the semi-supervised image processing in this section.

### 4.1 Semi-supervised Segmentation

We apply the semi-supervised clustering method in section 2.2 to image segmentation in this section. We compute the edge weight $s_{ijkl}$ from the pixel values in an input image and attach the label $x_{ij} = 1$ and $x_{ij} = -1$ to some pixels. We then propagate these labels to the remaining pixels in the image.

This semi-supervised label propagation can be equivalently reformulated by an unsupervised iterative filter for a graph with weighted nodes as follows.

When each pixel $(i, j)$ has a weight $w_{ij}$, eq.(8) becomes

$$f_{ij}^{(\xi+1)} = \sum_{k=-p}^{p} \sum_{l=-p}^{p} s_{ijkl} w_{i+k,j+l} f_{i+k,j+l}^{(\xi)} / \sum_{k=-p}^{p} \sum_{l=-p}^{p} s_{ijkl} w_{i+k,j+l} \tag{13}$$

which is an unsupervised iterative filter. In this weighted filter, if we set $w_{ij} = 1$ at labeled pixels and set $w_{ij}$ sufficiently small at the remaining unlabeled pixels, $x_{ij}$ of the labeled pixels does not vary from its initial value throughout the iteration and they propagate to their surrounding unlabeled pixels. Thus we can perform the semi-supervised clustering in section 2.2 using this weighted unsupervised filtering scheme.

The above fast algorithm for eq.(8) becomes for eq.(13):

[Construction of arrays]
Step 1) We calculate $u_{ijk} = e^{-\alpha k^2 - \beta(d_{ij} - d_{i+k,j})^2}$ for all $i, j, k$, and calculate $v_{ijl} = e^{-\alpha l^2 - \beta(d_{ij} - d_{i,j+l})^2}$ for all $i, j, l$, and we save them into 3-dimensional arrays.

Step 2) We calculate $b_{ij} = \sum_{l=-p}^{p} v_{ijl} w_{i,j+l}$ for all $i, j$.

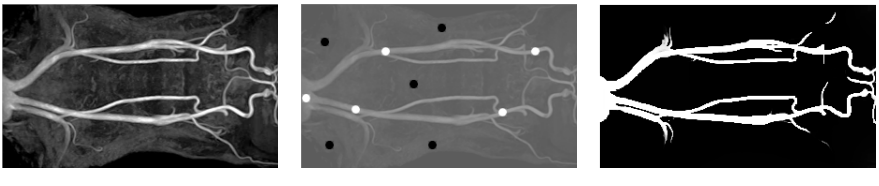
Step 3) We calculate $t_{ij} = \sum_{k=-p}^{p} u_{ijk} b_{i+k,j}$ for all $i, j$ and save it in a 2-dimensional array.

[Iteration of Filter]
Step 4) For all $i, j$, we calculate

$$a_{ij} = \sum_{l=-p}^{p} v_{ijl} w_{i,j+l} f_{i,j+l}^{(\xi)}, \quad s_{ij} = \sum_{k=-p}^{p} u_{ijk} a_{i+k,j} \tag{14}$$

from which we calculate $f_{ij}^{(\xi+1)} = s_{ij}/t_{ij}$ and repeat this computation.

We experiment this algorithm for an image of blood vessels in Fig.6(a) of size $295 \times 175$. We attach labels to 10 pixels shown in Fig.6(b) where labeled pixels are shown with disks. We set $x_{ij} = 1$ at the pixels of the center of 5 white disks and $x_{ij} = -1$ at the center pixels of 5 black disks for their initial values in the iterations. We set $p = 5, \alpha = 0.001, \beta = 0.01$ and iterate the filter 2000 times starting with the initial value $x_{ij} = 0$ for all the remaining unlabeled pixels. We set the weight of pixels as $w_{ij} = 1$ at 10 labeled pixels and $w_{ij} = 0.001$ at the unlabeled pixels. The value of $(x_{ij} + 1)/2$ after the 2000 iterations is shown in Fig.6(c) where major streams of blood vessels are extracted. Computational times are 286.6 seconds for FCIBF and 74.9 seconds with the fast FCIBF algorithm. This long computational time is due to many iterations of the filter needed for propagation of labels attached to a few pixels to the whole image. Full convergence is the reason for many iterations of 2000 times extremely longer than the optimal 30 times in Fig.3 for noise reduction which does not need the convergence of iterations. The convergence time can be decreased if labels are attached to more pixels.



(a) input image (b) labeled pixels (c) converged $x_{ij}$

**Fig. 6.** Semi-supervised image segmentation

(a) input image (b) HSL components (left:L, center:H, right:S)

(c) labeled pixels (d) converged H (left) and S (right) (e) output image

**Fig. 7.** Semi-supervised colorization of image

Object extraction scheme in the image matting task[8] is the same procedure as this semi-supervised image segmentation and the present algorithm is also useful for fast extraction of objects from photographs or videos with the aid of scribbles drawn by users.

### 4.2 Semi-supervised Colorization of Image

As a final experiment, we apply the above algorithm for semi-supervised label propagation to the colorization of monochromatic images. We compute the edge weight $s_{ijkl}$ between pixels from the pixel values in an input monochromatic image and teach the correct color at some pixels.

The color of the remaining monochromatic pixels is estimated by propagating the colors attached to these labeled pixels. We decompose the color values into HSL components in which the luminance is given as the input monochromatic image and we propagate the hue and saturate components from the labeled pixels to their surroundings.

We experiment with a color image in Fig.7(a) whose HSL components are shown in Fig.7(b) where the left is the luminance which is the input image, the center is the hue and the right is the saturation.

We give the correct color values at 28 pixels shown in Fig.7(c). We set $p = 10, \alpha = 0.01, \beta = 0.1$ and iterate the filter 2000 times. We set the node weights as $w_{ij} = 1$ at labeled pixels and $w_{ij} = 0.001$ for the remaining unlabeled pixels. The initial values of the hue and the saturation are set 127.5 at every unlabeled pixels. The hue after 2000 iterations of the filter is shown on the left in Fig.7(d) and the saturation on the right. Recombination of these hue and saturation with the luminance on the left in Fig.7(b) yields the color image shown in Fig.7(e) which reproduces Fig.7(a) well. The computational times are 228.3 seconds for

the FCIBF and 60.6 seconds with the fast FCIBF. This experiment also demands many iterations due to the color inputs at only a few pixels. The convergence becomes faster if the color is given at more pixels.

## 5 Conclusion

We have shown that the fixed-coefficient iterative bilateral filter is an equivalent solver for an optimization problem for a graph-based data analysis technique. Based on this equivalence, we have proposed a fast algorithm for the graph-based image processing using an iterated filter accelerated with decomposition of its coefficients. We have applied this algorithm to semi-supervised image segmentation and semi-supervised colorization of monochromatic images. Owing to its iterative nature of solution, our algorithm is suitable for interactive image processing with incremental labeling of pixels.

## References

1. von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. 17(4), 395–416 (2007)
2. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Patt. Anal. Mach. Intell. 22(8), 888–905 (2000)
3. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. Patt. Anal. Mach. Intell. 12(7), 629–639 (1990)
4. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proc. ICCV, pp. 839–846 (1998)
5. Belkin, P.M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Proc. NIPS, pp. 585–591 (2001)
6. Wang, F., Zhang, C.: Label propagation through linear neighborhoods. In: Proc. ICML, pp. 985–992 (2006)
7. Eisemann, E., Durand, F.: Flash photography enhancement via intrinsic relighting. ACM Trans. Graph. 23(3), 673–678 (2004)
8. Levin, A., Rav-Acha, A., Lischinski, D.: Spectral matting. In: Proc. CVPR, pp. 1–8 (2007)