All-in-focus with directional-max-gradient flow and labeled iterative depth propagation

Guijin Wang*, Wentao Li, Xuanwu Yin, Huazhong Yang

Department of Electronic Engineering, Tsinghua University, Beijing 100084, China

Abstract

Focus stacking is a computational technique to extend the Depth of Field (DOF) through combining multiple images taken at various focus distances. However, existing focus stacking methods could not cope with false edges produced by propagation of blur kernels, and are affected by colored texture in the stack. In this work, we propose a novel all-in-focus method based on directional-max-gradient flow (DMGF) and labeled iterative depth propagation. Firstly, we present a novel directional-max-gradient flow to describe gradient propagation along different directions in the stack to remove false edges and preserve accurate depth values of both strong and weak edges (also called source points). Then the source points are further labeled as in-plane edges and off-plane edges by unsupervised classification technique. Finally in our proposed labeled iterative Laplacian optimization, these edges are utilized to remove artifacts produced by colored texture in the stack and refine the all-in-focus image. Extensive experiments on both synthesized data and real data show that our method has achieved superior performance to state-of-the-art methods.

1. Introduction

Optical lenses usually focus on a specific plane, leaving other regions of the scene subject to various scales of defocus. Therefore, not all regions of a single image are in focus [1]. One way to extend the depth of field is to decrease the aperture size, but this could lead to lower signal-to-noise ratio. With the development of digital imaging technology, focus-stacking has become more popular [2,3], which captures a sequence of images focused at various planes and fuses them into a single all-in-focus image.

Focus-stacking has attracted lots of attention in the last decade, which could be divided into 2 categories: transform-domain-based approaches and depth-estimation-based approaches. For transform-domain-based approaches, source RGB images are converted in certain transform domain, and the final all-in-focus image is reconstructed by the inverse transform of the fused corresponding transform coefficients. Sroubek, Redondo et al. [4,5] fused the decomposed discrete wavelet transform (DWT) coefficients to get the all-in-focus image. Forster et al. [6] proposed a complex wavelet method to extend DOF of microscopy images. Dense scale invariant feature transform (DSIFT) [7] was utilized for the activity level measurement to fuse multi-focus images. Haghighat et al. [8] presented an approach for fusion of multi-focus images based on variance calculated in discrete cosine transform (DCT) domain. Kunthirummal et al. [9] has presented another technique called as Focal Sweep Imaging (FSI) to extend the DOF, where the sensor moved along the optical axis during one exposure. These transform-domain-based methods are usually complicated, unstable and sensitive to perturbation of transform coefficients.

For depth-estimation-based methods, researchers extract depth values of sparse pixels with the sharpest image gradient across the focal stack [10–12], then propagate the depth values to the dense depthmap. All-in-focus image is then reconstructed by extracting the pixels from focal stack according to the depthmap. Aguet et al. [13] estimated the all-in-focus image with a model based 2.5D deconvolution method. Moeller et al. [14] chose well-known modified Laplacian (MLAP) function as a measure of contrast, and propagated the resulting depth estimates in a single variational approach. Given a sharpness measurement, Suwajanakorn et al. [15] regarded the depthmap fusing problem as a multi-label MRF optimization problem on a regular 4-connect grid, where the pairwise energy was defined as total variation of gradients of neighboring pixels.

However, all the algorithms above have two main problems. Firstly, they did not consider large blur kernels case, where strong image edges would propagate the gradients in the scale of blur kernels to produce false edges and even shade neighboring weak edges. This problem could be partly overcome by Alonso et al. [16,17], who reconstructed each frequency component by

* Corresponding author.

E-mail addresses: wangguijin@tsinghua.edu.cn (G. Wang), lw14@mails.tsinghua.edu.cn (W. Li).

https://doi.org/10.1016/j.patcog.2017.10.040
0031-3203/© 2017 Published by Elsevier Ltd.
solving a linear equation based on a plane-wise approximation of the 3D scene. But these equations would become highly ill-posed and unstable when the number of images increases. Our previous MGF-ARF method [18] calculated max-gradient flow (MGF) to extract source points (whose depth values are confident) and relieved the problem of false edges. However, like most of the depth-estimation-based methods, its arched rolling filter (ARF) in depth propagation would suffer the second main problems: the depthmap might be affected by the colored texture of guided images. Specifically, in previous traditional depth-estimation-based methods, depth propagation process was based on color and structure differences of source images. Therefore, regions with different colored textures at the same depth plane might be wrongly estimated as different depth values.

In this paper, a novel depth-estimation-based method is proposed for focus-stacking to address the problems mentioned above. There are two main contributions: Firstly, we define a novel directional-max-gradient flow to improve gradient propagation model to remove false edges and preserve true depth values of edges as many as possible. Secondly, we classify source points into two different labels by an unsupervised classification technique, and propose a novel labeled iterative depth propagation method to remove the noisy effects of colored texture of guided images on the final depthmap. Extensive experiments on both synthesized and real-captured data show that our method could achieve superior performance to state-of-the-art methods.

The rest of our paper is organized as follows. In Section 2 the proposed hierarchical framework is described. Then we introduce the production of false edges and directional-max-gradient flow (DMGF) in Section 3. In Section 4, we describe our labeled iterative depth propagation method in detail. In Section 5, extensive experiments on both synthesized data and real data certify the effectiveness and robustness of our proposed method.

2. Hierarchical framework

As shown in Fig. 1, our hierarchical framework consists of two main modules: directional-max-gradient flows and labeled iterative depth propagation. Firstly, we define the directional-max-gradient flows DMGF$L$ and DMGF$R$ using 2 respectively orthogonal directional gradients. They describe the distribution and propagation of image gradients of strong edges and weak edges respectively. Given the directional-max-gradient flows, we could extract source points with valid depth values and construct sparse depthmap $\hat{d}$. The second module is labeled iterative depth propagation. In each iteration, source points are classified by K-means clustering into two different labels: off-plane edges and in-plane edges. Here in-plane edges refer to image edges on the same depth plane, while off-plane edges are image edges at boundaries of different depth planes. Given the labels, source points are utilized to construct labeled Laplacian Matrix $\hat{L}$. Then we propose labeled Laplacian optimization approach to estimate dense depthmap $d$ with Matrix $\hat{L}$ and sparse depthmap $\hat{d}$. Only color differences of off-plane edges in the guided image are utilized as correlation between neighboring pixels in the process of depth propagation. With the dense depthmap, we extract depth-edge features to update labels for source points in turn. Finally, our all-in-focus image could be reconstructed with the dense depthmap $d$ and focal stack $l$ as below:

$$Fl(x, y) = L_{d(x,y)}(x, y).$$

where $Fl(x, y)$ refers to the final all-in-focus image.

3. Directional-max-gradient flow

In this section, we firstly explain the production of false edges in large blur kernel case. Then we propose the directional-max-gradient flow to improve the model of propagation of gradients from different directions.

3.1. False edges

Given focal stacks $l_1, \ldots, l_n$, an all-in-focus image can be generated by stitching together the sharpest in-focus pixels across the focal stack. Different measures of pixel sharpness have been defined in some shape-from-focus literature [10–12]. Without loss of generality, we take the magnitude of gradient as sharpness measure to explain the production of false edge. The magnitude of gradient is defined as:

$$G_i = |\nabla I_i(x, y)| = \sqrt{(\frac{\partial I_i}{\partial x})^2 + (\frac{\partial I_i}{\partial y})^2},$$

where $G_i$ is the absolute gradient of $I_i$, the $i$-th image in the focal stack. Then, once the gradients of all images are calculated, the depthmap can be calculated as

$$d(x, y) = \arg \max_i G_i(x, y).$$

where $d(x, y)$ stores the depth value with maximum gradient of each pixel. However, in large blur kernel case, strong edges would propagate and result in false edges with larger gradients at neighboring pixels. These false edges would lead to wrong depth values following Eq. 3.

To describe the production of false edge clearly, we construct a simulated scene consisting of two planars with different depth values. The depthmap and groundtruth all-in-focus image are shown in Fig. 2(a). The stack consists of 17 images and is constructed by convoluting the all-in-focus image with different disk blur kernels according to the depthmap. The depth values of the left and right

---

**Fig. 1.** The framework of our proposed method.
Fig. 2. (a) Groundtruth of depthmap and all-in-focus of our simulated data. (b) Close-up view of 9th, 11th and 14th image in the stack. (c) The gradients of three sample points across the stack.

Fig. 3. Three images captured from real scene focused at 3 focusing plane respectively.

half are 9 and 11 respectively. Fig. 2(b) shows close-up views of patches from 9th, 11th and 14th images in the stack, which contain both strong edges and weak edges. The strong edges are sharp in Image 9 and blurred in the others, while the weak edges are sharp only in Image 11. There are three different points in Fig. 2(b). The blue point focused in Image 9 is located just at the strong edge. The red point focused in Image 11 is located at the weak edge within the blur range of the strong edge, while the green point focused in Image 11 is located at the weak edge out of the blur range of strong edges in Image 14. The gradients of three typical points are displayed in Fig. 2(c), where the horizontal axis is the index of image in the stack and the vertical axis is the gradient value of each point. From this figure, false edge occurs at the red point in Image 14, whose gradient is even bigger than that of the true edge in Image 11. Therefore, according to Eq. 3, the depth value would be estimated as 14 mistakenly because the strong edges would propagate its gradient along its normal direction and shade the original gradient of weak edges at the red point.

Similar scenario can also be observed in real scene. Here we capture a sample stack with a mono camera equipped with a SIGMA 50mm /F1.4 lens. This captured stack is utilized to describe our method here and also to evaluate the performance of our method in the experiment section. Fig. 3 shows three different images focused at respective depth planes in the stack. Fig. 4(a) shows the sample patches extracted from Image 6 and Image 11 respectively and three chosen points marked in different colors, which are all focused on Image 11. Gradients of these marked points across the depth index are given in Fig. 4(b). Observing Fig. 4, we could find that a false edge appears at the red point in Image 6, whose gradient is even larger than that of the true edge in Image 11 due to the propagation of gradients from the green point.

3.2. Directional-max-gradient flow

In this section, we define directional-max-gradient flow to model the propagation of gradients to remove false edges and extract points with true depth values. In our previous work [18], max-gradient flow is defined as

\[ \mathbf{MGF}(x, y) = (f_x(x, y), f_y(x, y))^T, \]

with the elements of its

\[ \begin{bmatrix} f_x(x, y) \\ f_y(x, y) \end{bmatrix} = \begin{bmatrix} \max_i G_i(x + \Delta x, y) - \max_i G_i(x, y) \\ \max_k G_k(x, y + \Delta y) - \max_i G_i(x, y) \end{bmatrix} \]

where \( G_i \) is calculated according to Eq. (2) in the last section. The direction along with the max-gradient flow can be derived as:

\[ \theta(x, y) = \arctan \frac{f_y(x, y)}{f_x(x, y)} \]

With the max-gradient flow, two kinds of points are defined: source points and trivial points. Source points are points whose depth values calculated by Eq. (3) are true and valid, while trivial points are points with false depth values from Eq. (3). The point \((x, y)\) is defined as a source point if:

\[ \nabla \cdot \mathbf{MGF}(x, y) > 0 \quad \text{and} \quad \max_i G_i(x, y) > G_{TH}, \]

which means that the direction of the flow changes oppositely at \((x, y)\). The threshold \(G_{TH} \) is introduced to remove the wrong edge detections caused by noises. Otherwise, point \((x, y)\) would be chosen as a trivial point, formulated as:

\[ \nabla \cdot \mathbf{MGF}(x, y) \leq 0 \quad \text{or} \quad \max_i G_i(x, y) \leq G_{TH}, \]
which means that the direction of the flow does not change oppositely at \((x, y)\).

However, since max-gradient flow is calculated by magnitude of points shown in Eq. (2), the max gradients of weak edges are easily affected by the spreading and crossing of neighboring strong edges from different directions in large blur kernel case. Therefore, although false edges are detected and true depth values of source points are determined with max-gradient flow, weak edges from other directions are ignored due to propagation of gradients of neighboring strong edges.

In our method, we extend our previous work [18] and decompose the gradient of each point along 2 respectively orthogonal directions. Since max gradient flow is calculated by gradient magnitude, it could depict the main propagation of gradients in the stack. Therefore we decompose the gradient of each point along and vertical to the direction of max-gradient flow respectively.

\[
[G_P(x, y), G_\perp(x, y)]^T = \begin{bmatrix} \frac{\partial h(x, y)}{\partial u} & \frac{\partial h(x, y)}{\partial v} \end{bmatrix}.
\]

Here \(u\) and \(v\) are the 2 orthogonal directions along and vertical to the direction of max-gradient flow. Therefore \(G_P\) depicts the gradients of strong edges, while \(G_\perp\) removes the effects of strong edges and extracts gradients of weak edges because of orthogonality. Then we define directional-max-gradient flows with 2 directional gradients as follows:

\[
DMGF_p(x, y) = \frac{\max_{j} G_{\perp, j}(x, y) + \Delta \mathbf{u}}{\| \Delta \mathbf{u} \|} - \frac{\max_{j} G_{\perp, j}(x, y)}{\| \Delta \mathbf{u} \|}
\]

\[
DMGF_w(x, y) = \frac{\max_{j} G_{\perp, j}(x, y) + \Delta \mathbf{u}}{\| \Delta \mathbf{u} \|} - \frac{\max_{j} G_{\perp, j}(x, y)}{\| \Delta \mathbf{u} \|}
\]

Considering these two directional-max-gradient flows, we could find that \(DMGF_p\) describes the distribution and propagation of the directional-max-gradient along the strong edges’ gradient direction, while \(DMGF_w\) calculates the directional-max-gradient flow with gradients orthogonal to the strong edges and describes the propagation of gradients of weak edges. Both strong edges and weak edges could be preserved with our directional-max-gradient flow.

With the analysis of directional-max-gradient flow, edges could be divided into two categories similarly: trivial points and source points. Point \((x, y)\) is defined as a source point if:

\[
(V \cdot DMGF_p > 0 \text{ and } \max_{j} G_{\perp, j}(x, y) > G_{\perp, T}) \quad \text{or} \quad (V \cdot DMGF_w > 0 \text{ and } \max_{j} G_{\perp, j}(x, y) > G_{\perp, T}),
\]

which means that either directional-max-gradient flow changes its direction oppositely at\((x, y)\). Otherwise, the point \((x, y)\) is defined as a trivial point if:

\[
(V \cdot DMGF_p \leq 0 \text{ or } \max_{j} G_{\perp, j}(x, y) < G_{\perp, T}) \quad \text{and} \quad (V \cdot DMGF_w \leq 0 \text{ or } \max_{j} G_{\perp, j}(x, y) < G_{\perp, T}),
\]

which means that neither directional max-gradient flow changes its direction oppositely at\((x, y)\).

If \(V \cdot DMGF_p > 0\), it means directional-max-gradient flow of strong edges at \((x, y)\) changes its direction oppositely and these strong edges are regarded as source points. Otherwise, if \(V \cdot DMGF_w > 0\), these weak edges are also regarded as source points. Therefore, our method not only removes false edges, but also preserves both strong edges and weak edges.

Fig. 5 shows the true source point extracted with directional-max-gradient flows. Fig. 5(a) displays the 11th and 14th image in the simulated focal stack in Fig. 2. The red point is sharpest in 11th image and is located exactly at the weak edge. Max-gradient flow is displayed at right column in Fig. 5(b) and (c). This flow only depicts gradient propagation of strong edges along the y-axis and classify the red point as a trivial point. In our proposed method instead, we decompose the gradient of the red point into two orthogonal directions\((u\) and \(v\), along and vertical to directional of max-gradient flow. In this way, in Fig. 5(c), \(G_p\) depicts the gradients of the strong edge along y-axis across the stack, while \(G_\perp\), whose peak value is at the 11th stack, describes the gradients of the weak edges towards upper right. Fig. 5(c) also shows the \(DMGF_p\) and \(DMGF_w\) calculated from Eq. (10) and Eq. (11), whose directions are respectively parallel to the gradients of strong edges and weak edges. These two flows could depict propagation of gradients of strong edges and weak edges. Although \(DMGF_p\) does not change its direction oppositely, \(DMGF_w\) changes its direction oppositely indeed, which classifies the red point into source points.

The extracted source points of the sample patch from our simulated focal stack and captured real scene are shown in Fig. 6. Compared with source points extracted with max-gradient flow (MGF) [18], we find that in our method, false edges are effectively suppressed and more true edges are preserved, especially those weak edges within the propagation range of strong edges’ blur kernels (the red ellipse). Therefore our method can estimate true depth.
values of source points as many as possible in large blur kernel case.

4. Depth propagation

From the previous step, we extract and calculate depth values of source points and generate a sparse depthmap. In this section, we assign source points two types of labels: off-plane edges or in-plane edges, and present an iterative method to propagate the depth value from source points to the dense depthmap and refine the all-in-focus image. Initially, all source points are set as in-plane edges. In each iteration, given the labels, dense depthmap is generated by labeled-Laplacian optimization approach. Then, depth-edge features for all source points are extracted from dense depthmap and their labels are updated by unsupervised classification technique. These labels are then utilized to regenerate the depthmap in turn. In this way, better depthmap is generated to refine the all-in-focus image.

4.1. Labeled Laplacian optimization

In our method, like the traditional Laplacian optimization [19], the depth propagation problem in each iteration could be formulated as minimizing the following cost energy:

$$E(d) = d^T L d + \lambda (d - \bar{d})^T D (d - \bar{d}),$$

(14)

where $L$ is the Laplacian matrix, $D$ is a diagonal matrix whose element $D(i, i)$ is equal to 1 if the pixel $i$ has valid depth value, $\bar{d}$ is vector form of valid sparse depthmap generated from last section and $d$ is the dense depthmap. The scalar $\lambda$ controls the balance between smoothness of depth propagation and the fidelity of source points. Here we denote $l_i$ and $l_j$ as colors of input all-in-focus image reconstructed from depthmap in each iteration at pixel $i$ and $j$ respectively. The $(i, j)$ element of traditional Laplacian matrix $L$ is defined as follows:

$$L(i, j) = \sum_{k \neq i, j \in \omega_k} \left( \delta_{ij} - \frac{1}{|\omega_k|} \left( 1 + (l_i - \mu_k)^T \Sigma_k + \frac{\epsilon}{|\omega_k|} U_j \right)^{-1} \right) \times (l_j - \mu_k),$$

(15)

where $\delta_{ij}$ is the Kronecker delta, $U_j$ is identity matrix, $\omega_k$ is a small window around pixel $k$, and $\Sigma_k$ is the covariance matrix of the colors in $\omega_k$, $\epsilon$ is a regularization parameter and $|\omega_k|$ is the size of window $\omega_k$. For the detailed derivation of the formation of Laplacian Matrix, readers can refer to [19].

In our method, different from traditional Laplacian optimization, the $(i, j)$ element of our new labeled-Laplacian Matrix $\hat{L}$ is defined as

$$\hat{L}(i, j) = \sum_{k \neq i, j \in \omega_k} \left( \delta_{ij} - \frac{1}{|\omega_k|} \left( 1 + (l_i - \mu_k)^T \left( \Sigma_k + \frac{\epsilon}{|\omega_k|} U_j \right)^{-1} \right) \right) \times (l_j - \mu_k).$$

(16)

where

$$\chi(i, k) = (1 - \Pi_1) l_i + \Pi_1 \mu_k.$$  

(17)

Here source points are divided into two categories: in-plane edges ($\Pi_1=0$) and off-plane edges ($\Pi_1=1$) and this labeled Laplacian Matrix could propagate depth values from these two type of edges in different manners. If the point $i$ or $j$ belongs to off-plane edges, the depth discontinuities should be aligned with image edges, the similarity value $\hat{L}(i, j)$ equals to traditional Laplacian matrix value calculated from mean and covariance matrix of colors in window $\omega_k$. If the point $i$ and $j$ belongs to in-plane edges, the depth discontinuities of it should be vanished, therefore we force $\chi(i, k) = l_i$ to constrain all pixels in the patch $\omega_k$ having the same color value. In this way we remove noises of depthmap at in-plane edges and reserve the sharp difference of depth values according to the difference of RGB colors at off-plane edges. In the next section, we would explain how to extract in-plane edges and off-plane edges from depthmap in each iteration.

4.2. Labels for source points with depthmap

Assuming dense depthmap is known, now explain how to classify source points into in-plane edges and off-plane edges. For simplification, we take 1-D ray integration process as the example to introduce our principle for classification. Fig. 7 shows the 1-D ray integration and depth-edge features extraction process. Four points (A,B,C,D) are located on two neighboring paralleled straight line segments(different colors on the line indicate colored texture) and are all source points because of their sharp gradients. In fact, point A and point D are off-plane edges, at the boundary of lines with different depth values, while the other two points B and C are indeed in-plane edges at the same planar. Only three blue points are visible whose depth values could be calculated with our DMGF method in Section 3.

We then utilize traditional Laplacian optimization to propagate depth values from visible source points (point A,B and C) to the entire 1D scene(the two straight line segments), as shown for the depthmap in the middle of Fig. 7. Spreading both N pixels
along and against rising direction of gradient in the depthmap, we extract depth values of surrounding $(2N+1)^2$ pixels of point A, B and C as corresponding depth-edge features for classification. Each depth-edge feature describes changes and distribution of depthmap surrounding each source point. Without loss of generality, we assume that smaller depth value means nearer depth plane. Firstly, point C, whose neighboring depth values are nearly the same, should be classified as in-plane edges because its depth-edge feature behaves as a horizontal straight line. Secondly, point B should also be classified as in-plane edges because its depth-edge feature rises in the first half and remains large depth value in the second half, as shown in Fig. 7. Since farther edges would be sheltered by nearer edges, we could verify that off-plane edges, which denote boundaries of objects with different depth values, must have relatively smaller depth values among neighboring points. Therefore point A should be classified as off-plane edges because its depth-edge feature keeps constant small depth value in the first half and rise in the second half. The discussion above indicates that depth-edge features are representative for classifying source points into off-plane edges and in-plane edges.

Extending our analysis above into 2-D scene, we extract $(2N+1)^2$ sized depth-edge operator $\alpha$, similarly as 1D situation, as input classification feature for each source point. According to the explanation of ray integration above, we could divide source points into three main categories according to their depth-edge features. Fig. 8 shows the results of features extracted from our captured 2D scene. Fig. 8(a) presents the sparse depth of source points extracted from Section 3. Three different example points are also displayed in Fig. 8(a). Only the red one (Point A) is the off-plane edge, while the blue point (Point B) and the green point (Point C) are both in-plane edges. Fig. 8(b) gives the initial dense depthmap in our first iteration, from which we extract depth-edge features (the close-up view). These example points’ depth-edge features are shown in Fig. 8(c), where the vertical axis is the depth value of each pixel and the horizontal axis is the coordinate of the pixel. From Fig. 8(c), we observe the similar depth-edge features as 1D case in Fig. 7. Therefore only point A is classified as off-plane edges while point B and C are classified as in-plane edges.

Therefore we use k-means algorithm with Euler-Distance as the cost function of depth-edge features of all source points to give them three labels ($l$), centered as three $(2N+1)^2$ sized vectors $C_1^i$, $C_2^i$ and $C_3^i$ ($i=1,2,3$).

Then the following equation is applied to choose the label $l^*$ of off-plane edges

$$l^* = \arg \max_c \frac{\mathbf{C}^i[2N+1] - \mathbf{C}^i[N+1]}{\mathbf{C}^i[N+1] - \mathbf{C}^i[1]}$$

where $C^i[n]$ refers to the n-th element of vector $C^i$. We choose source points whose cluster center has the largest depth value variation in the second half as off-plane edges. And we utilize $\pi_i$ to denote whether the source point $i$ belongs to off-plane edges by

$$\pi_i = \begin{cases} 1 & i \in l^* \\ 0 & i \notin l^* \end{cases}$$

4.3. Iteration between depthmap and off-plane edges

From the two sections above, the depthmap could be propagated by labeled Laplacian optimization according to the distribution of off-plane edges and in-plane edges. We smooth depthmap at in-plane edges while strengthening the differences of depthmap at off-plane edges. On the other hand, suppose that an estimated depthmap is available, the depth-edge features would be extracted for classification. Thus we take these two processes to form a ping-pong approach that extracts the true off-plane edges and refine the depthmap in turn. In this way, the depthmap and distribution of true off-plane edges could be updated iteratively. Initially, we set all source points in-plane edges. The iterative process is shown as below:

\[
\text{Initial: } \Pi^0 = 0, \\
\text{and the n-th iteration of our method goes as:} \\
\text{Step1: update the depthmap with Eq. (16) and Eq. (14), get } d^n \\
\text{Step2: update the added off-plane edges with Eq. (19) and get } \Delta \Pi^n \\
\text{Step3: update the distribution of off-plane edges with: } \Pi^{n+1} = \Pi^n + \Delta \Pi^n.
\]

Our iteration continues until the off-plane edges $\Pi^n$ shows no significant change between two iterations. After the depthmap is achieved, we estimate the all-in-focus image according to Eq. (1). Performance of the depthmap in our iteration is shown in Section 5.3.2.

5. Experiments

In this section, we will present the experimental performance of our proposed method. In Section 5.1, we introduce the datasets and the parameters in our experiments. Section 5.2 evaluates the overall performance of our proposed method with state-of-the-art methods on both real data and synthesized data. In Section 5.3, we analyze the advantage of two proposed modules: directional-max-gradient flow and labeled iterative Laplacian optimization, and show the performance of convergencies of iterations. At last, Section 5.4 discusses the impacts of parameters on the final all-in-focus image in detail.

5.1. Setup

To evaluate performance of our proposed method, we utilize two groups of synthesized datasets. The first group of synthesized focal stacks is reconstructed by light field data taken from the New Stanford Light Field Archive [20] which includes

- Card, a scene with a crystal ball located on tarot cards. The ball acts as a lens. The cards act as diffuse textured objects at many orientations and depths (512x512, 49 images)
- Truck, a Lego Technic truck with very complex geometry (640x480, 49 images)
- Chess, a chess board with pieces, which is great for demonstrating refocusing (700x400, 49 images)

| Table 1 | Experiment parameter. |
|--------|-----------------------|
| Parameter | Meaning | Value |
| $G_{th}$ | Threshold of gradient | 0.15 |
| $\lambda$ | balance between smoothness and fidelity | 4 |
| $\sigma$ | regularization parameter in Laplacian optimization | 0.0001 |
| $N$ | length of feature operator | 10 |
Table 2  
SSIM of different methods on synthesized data.

|                | ours   | MGF-ARF | DCT   | DSIFT | DWT   | CWB   | 2.5D  | FSI   |
|----------------|--------|---------|-------|-------|-------|-------|-------|-------|
| card           | 0.9585 | 0.9506  | 0.9501| 0.9532| 0.9440| 0.9177| 0.9274| 0.7864|
| truck          | 0.9555 | 0.9523  | 0.9419| 0.9480| 0.9551| 0.9328| 0.9368| 0.9067|
| chess          | 0.9333 | 0.9156  | 0.9019| 0.9051| 0.9272| 0.8598| 0.8847| 0.8563|
| knights        | 0.8654 | 0.8099  | 0.7598| 0.8462| 0.8005| 0.8321| 0.7818| 0.6973|
| treasure       | 0.9649 | 0.9109  | 0.8837| 0.8612| 0.8355| 0.9312| -     | 0.7308|
| buldozer       | 0.9493 | 0.9134  | 0.8804| 0.8959| 0.8623| 0.9210| 0.8919| 0.7792|
| boxes          | 0.9768 | 0.9666  | 0.9684| 0.9725| 0.9698| 0.9626| -     | 0.9633|
| cotton         | 0.9919 | 0.8984  | 0.9907| 0.9948| 0.9953| 0.9912| 0.9843| 0.9759|
| dino           | 0.9935 | 0.8984  | 0.9904| 0.9652| 0.9915| 0.9795| 0.9820| 0.9601|
| sideboard      | 0.9691 | 0.9519  | 0.9358| 0.9647| 0.9610| 0.9539| 0.9414| 0.8442|

Table 3  
PSNR of different methods on synthesized data.

|                | ours   | MGF-ARF | DCT   | DSIFT | DWT   | CWB   | 2.5D  | FSI   |
|----------------|--------|---------|-------|-------|-------|-------|-------|-------|
| card           | 31.232 | 29.822  | 29.638| 30.770| 29.862| 26.831| 28.390| 20.476|
| truck          | 37.234 | 35.902  | 34.724| 36.002| 36.876| 32.903| 34.031| 29.515|
| chess          | 38.314 | 34.998  | 33.229| 35.809| 36.382| 27.823| 31.268| 26.680|
| knights        | 27.309 | 25.325  | 23.635| 26.936| 24.760| 26.961| 25.040| 20.977|
| treasure       | 33.477 | 29.103  | 28.443| 27.652| 27.716| 31.608| -     | 21.929|
| buldozer       | 32.505 | 28.728  | 25.646| 29.601| 26.784| 29.535| 27.980| 20.691|
| boxes          | 35.301 | 32.834  | 33.281| 34.188| 33.753| 33.025| -     | 33.751|
| cotton         | 47.738 | 44.588  | 42.626| 48.046| 49.120| 45.200| 41.098| 37.933|
| dino           | 41.160 | 38.813  | 39.293| 31.011| 30.663| 35.867| 35.947| 31.914|
| sideboard      | 30.729 | 29.296  | 27.030| 30.198| 30.620| 29.544| 29.380| 23.585|

- Knights, a lego scene which contains relatively simple geometric objects at various depths (512×512, 49 images)
- Treasure, a chest with gemstones, coins, and jewelry spilling out which contains highly complex specular geometry at a fairly high resolution (768×640, 70 images)
- Buldozer, a Lego Technic bulldozer with very complex geometry (768×576, 70 images)

We also reconstruct focal stacks from light field data taken from Training set of 4D Light Field Benchmark [21]. This dataset supplies depthmap groundtruth and contains

- Boxes, boxes with complex geometry (512×512, 49 images)
- Cotton, a statue with simple colored texture (512×512, 49 images)
- Dino, scene which contains relatively simple geometric objects at various depths (512×512, 49 images)
- Sideboard, a sideboard which contains relatively simple geometric objects at various depths (512×512, 49 images)

We also test the performance of our proposed method on real-captured focal stack, which contains:

- Caps, provided by Levin et al. [22] (1350×700 41 images)
- Fedex, provided by Levin et al. [22] (1350×700 41 images)
- Slice, captured by us with an Imperx B4020 mono camera equipped with a SIGMA 50mm/F1.4 lens [18](504×336, 14 images)

When capturing the real-captured data, the movement of the focusing plane will cause the change of Field of View (FOV). This FOV change is corrected with the image registration technique [23].

In our experiments, parameters are set as Table 1. These parameters are commonly used in all synthesized data and real data.

5.2. Overall performance

In this section, we compare our method on datasets with the DCT-based [8], DWT-based [4,5], DSIFT-based methods [7], 2.5D deconvolution method[2,5D] [13], the Complex Wavelet-based Method(CWB) [6], MGF-ARF method [18] (codes provided by the author) and FSI-based method [9] (implemented by ourselves).

We run all experiments on the same hardware setup: Intel Core i7-4790 CPU @3.6GHz, 32GB RAM. Take the dataset Truck for example, our CPU-only Matlab implementation without parallel optimization costs less than two minutes to reconstruct one entire all-in-focus image. In our algorithm, the labeled iterative depth propagation process takes most of the time, mainly depending on the spatial resolution of images in the stack rather than numbers of focal stack we utilize. We also test time complexity for other state-of-the-art methods with provided source codes (all implemented by CPU-only Matlab except for 2.5D deconvolution and CWB-based method, provided as plugin of Imagej, a general purpose image-processing package). In all these methods, FSI-based method take tens of seconds to implement, but with poor performance of reconstructed all-in-focus image. CWB-based method and DWT-based method could be implemented within one minute, and MGF-ARF based method also costs one to two minutes while DSIFT-based method and DCT-based method take more than five minutes and even longer along with the increase of numbers of focal stack.

We firstly compare our all-in-focus method on synthesized data with state-of-the-art methods by peak signal-to-noise ratio (PSNR) and Structural Similarity (SSIM) [24] compared with the groundtruth. These two indexes are representative to evaluate the similarity of two images. The results are presented in Table 2 and 3. In the tables below, - means that we could not generate all-in-focus image with the provided codes of 2.5D deconvolution on dataset Treasure and Boxes. From the comparison, our method
achieves the highest SSIM values and PSNR values in nine of ten datasets. For the Cotton dataset, weak textures in the background of the statue is smoothed in our proposed method due to tiny gradients, and our proposed method still behaves comparable scores with other state-of-the-art methods.

To show the advantage of our method visually, we show all-in-focus results of the synthesized dataset Chess in Fig. 9. In Fig. 9(a), we preserve all sharp edges on the checkerboard, free of artifacts and false edges. Therefore our method achieves the highest SSIM value. For other methods, DWT-based, CBW-based, FSI-based and DSIFT-based method all produce false edges near strong edges of checkerboards. DCT-based method produces colored blocking noises. MGF-ARF method and 2.5D deconvolution method both show blurry edges due to wrong depthmap affected by black-and-white colored checkerboard in the guided image.

In Fig. 10, Fig. 11 and Fig. 12, we display performance of all-in-focus image calculated from our proposed method and other state-of-the-art methods on the real-captured datasets.

We firstly compare the performance of our methods with others on the dataset Caps. Fig. 10 presents the whole all-in-focus result and 4 different patches of the image. From the comparison, we can see that our method removes false edges and behaves best all-in-focus effects visually in all four patches. The CBW-based and DCT-based method produce blocking or particle noises in the whole image. The DWT-based, FSI-based and DSIFT-based methods produce blurry edges between different depth planes, such as the edge between the cup and the pen. MGF-ARF method behaves blurry in the patches because of noises of the dense depthmap. 2.5D deconvolution method even could not generate all-in-focus image because of large spatial resolution.

Fig. 11 shows another real-captured dataset Fedora. In our method, the all-in-focus image behaves the best visually because false edges are removed and sharp edges of objects at different depth planes are all reserved. Whereas, DCT-based, DWT-based, FSI-based and CBW-based methods all produce false edges in the two patches on the left. DSIFT-based and DWT-based appear blocking noises in patches on the right, while MGF-ARF produce blurry edges because of wrongly estimated dense depthmap.

In Fig. 12, we show the performances on the dataset Slice captured in real scene of large blur kernels. From the comparison, our method, Fig. 12(a) achieves the best all-in-focus image visually. In the bigger patch, weak edges are preserved as well as strong edges, and the result is free of artifacts and false edges. In the smaller patch, we preserve more source points on the top of the box with directional-max-gradient flow and make the textures much sharper and clearer. Whereas, the FSI-based method produces artifacts near strong edges. 2.5D deconvolution method, CBW-based method and DWT-based method all lose weak edges in the large blur kernel case, especially near strong edges in the larger patch. The DSIFT-based method and DCT-based method could not handle the large blur kernel case, and produce blurry results. MGF-ARF method displays blurry on top of the box in the smaller patch due to fewer source points.

5.3. Module analysis

In this section, we mainly analysis the advantage of our main two modules. Firstly, we show the advantage of applying directional-max-gradient flow to the all-in-focus problems. Secondly, we discuss our proposed labeled iterative Laplacian optimization for depth propagation and compare depthmap with other state-of-the-art depthmap-estimation method to certify the validity of our proposed method. At last, we show the performance of convergency of our iterative depth propagation method.

5.3.1. Directional-max-gradient flow

Fig. 13 displays the entire and close-up views of 3 different patches of depth value of sparse points. Fig. 13(a) shows the depth value result for points with gradient values larger than \( G_{max} \) of captured image without applying our proposed directional-max-gradient flow method, while (b) and (c) give depth values of source points extracted with max-gradient flow and our directional-max-gradient flow respectively. In the figure, Fig. 13(a) appears false depth values caused by false edges in large blur kernel case, which has been removed in both Fig. 13(b) and (c), but our directional-max-gradient flow reserves more depth values of weak edges than max-gradient flow, especially within the propagation range of strong edges’ blur kernel. Therefore our DMGF not only removes false edges appeared in large blur kernel case, but also preserves both strong edges and weak edges.

5.3.2. Depthmap propagation

In this section, we compare our proposed labeled iterative Laplacian optimization method with 3 other classical methods for depth propagation: traditional Laplacian optimization (implemented by ourselves), Variational Depth From Focus(VDFF)\(^5\) [14] and MGF-ARF to explain the validity of our propagation method.

The comparison on the synthesized data Chess is shown in Fig. 14. Our method achieves the best dense depthmap shown in Fig. 14(a). Whereas, in all other methods, the depthmaps on the chessboard are affected by the black and white texture and appear some depth discontinuities at these in-depth edges.

To evaluate the performance of our depth propagation method numerically, we also compare the depthmap of our proposed method with other depth-estimation methods on the synthesized dataset Dinos, which supplies depthmap groundtruth. Fig. 15 shows depthmap generated by our proposed method, VDFF, traditional Laplacian method and MGF-ARF evaluated by SSIM values, together with groundtruth of depthmap and all-in-focus image. From two extracted patches in Fig. 15, our method preserves most of the sharp edges of different depth planes while smoothing depthmap at in-plane edges and behaves highest SSIM value. In MGF-ARF and traditional Laplacian methods, however, the depthmaps in the second patch are affected by the texture on the wall of the guided image and appear depth discontinuities at these in-depth edges. For VDFF method, the edges of the toy bear is unclear in the first patch, and some edges of different depth planes are lost.

5.3.3. Iteration performance

In this section, we show the convergence performance of our iterative depth propagation process between depthmap and off-plane edges distribution. Our method is applied to our own captured Sliced data. The depthmap in 3 different iterations and off-plane edges in the final iteration are shown in Fig. 16.

From the figure, we find that the depthmap in the iteration behaves smooth at textures in the same depth plane, and becomes sharper at locations where depth value changes (off-plane edges) with the iterations. We could also find from Fig. 16(e) that our iterative depth propagation method has fast convergency speed; the distribution of off-plane edges keeps constant after only 5 iterations.

5.4. Parameter discussion

In this section, we discuss the influence of parameters shown in Table 1 on all-in-focus image in detail. To verify the performance of

---

5 https://github.com/adrelino/variational-depth-from-focus.
Fig. 9. Comparison of our method with state-of-the-art methods on synthesized data. (a) upper left: result of the all-in-focus of our method. down left: error-map of all-in-focus compared with groundtruth right. close-up view of 2 different patches (b)-(g) the result of our method and other state-of-the-art methods.
Fig. 10. Comparison of our method with state-of-the-art methods on real images Caps. (a) result of the all-in-focus of our method and close-up views of four different patches (b)-(g) the result of other state-of-the-art methods.

our method versus different parameter values (here parameters include $G_{TH}$, $\lambda$, $\epsilon$ and $N$), we carry out experiments on dataset Treasure taken from the New Stanford Light Field.

5.4.1. Impact of $G_{TH}$

$G_{TH}$ affects the source points with DMGF method. Fig. 17(a) shows how the performances of our method varies with different values of the $G_{TH}$ by SSIM. We can find that when $G_{TH}$ is small, the performance dramatically increases along with $G_{TH}$ because at this
time, some tiny noises of source points might be produced but not excluded by our DMGF method. When $G_{TH}$ increases, the SSIM values keep stable. And when $G_{TH}$ is large enough (over 0.5), the performance slightly decreases because some true source points might be excluded and removed. Considering the similar performance of changes of $G_{TH}$ on other synthesized datasets, we choose 0.15 as the value of $G_{TH}$ of all synthesized datasets.

5.4.2. Impact of $\lambda$, $\epsilon$ and $N$

In this section, we discuss the influence of values of $\lambda$, $\epsilon$ and $N$ on the performance of our method by SSIM as shown in

Fig. 11. Comparison of our method with state-of-the-art methods on real images of FedEx. (a) result of the all-in-focus of our method and close-up view of four different patches (b)-(g) the result of other state-of-the-art methods.
Fig. 12. Comparison of our method with state-of-the-art methods on real images of large blur kernel. (a) left: result of the all-in-focus of our method. right: close-up view of 2 different patches (b)-(h) the results of other state-of-the-art methods.
Fig. 13. (a): the method without applying gradient flows. (b): depth value for source points extracted with max-gradient flow, (c) depth value for source points extracted with directional-max-gradient flow.

Fig. 14. Comparison of depthmap of our method with state-of-the-art methods on synthesized data Chess, including MGF-ARF method, traditional Laplacian optimization and VDFF.

Fig. 15. Comparison of depthmap of our method with state-of-the-art methods on synthesized data Dinos, including MGF-ARF method, traditional Laplacian optimization and VDFF by SSIM.

Fig. 17(b)(c)(d). These three parameters are utilized in labeled iterative depth propagation: \( \lambda \) controls the balance between smoothness and fidelity, while \( \epsilon \) is the regularization parameter in Laplacian optimization and controls the extent of sharpness at edges of depthmap. \( N \) decides the length of the feature we extract. When \( \lambda \) is small, the SSIM value of our proposed method increases dramatically with \( \lambda \) and the fidelity of depth values of source points are ignored. When \( \lambda \) is large enough, the performance of our method keeps quite stable, for the reason that fidelity and smoothness of...
Given the large number of experiments, we have derived several conclusions. First, our methods have achieved superior performance to state-of-the-art methods in the denoising process. Second, the proposed methods have been found to be effective in preserving structural information in the images.

**6. Conclusion**

In this paper, we propose a novel all-in-focus method with a directional-max-gradient flow and labeled iterative depth propagation. Firstly, we define a directional-max-gradient flow to describe gradient propagation across the stack along different directions to divide points into source points and trivial points. Source points are extracted as true valid edges and utilized to remove false edges and preserve both strong and weak edges. Then we propose labeled iterative Laplacian optimization for depth propagation to refine the depthmap and all-in-focus image. In each iteration, unsupervised classification technique is utilized to divide source points into off-plane edges and in-plane edges. Given the labels, the labeled-Laplacian optimization is utilized to smooth depth value of regions at the same depth plane, while strengthening boundaries of different depth planes. Extensive experiments on both synthesized data and real data show that our method has achieved superior performance to state-of-the-art methods.

**Acknowledgements**

This work is partially supported by the 863 Program 2015AA016304.

**References**

[1] E. Allen, S. Triantaphillidou, The Manual of Photography and Digital Imaging, CRC Press, 2012.
[2] N.T. Goldsmith, Deep focus; a digital image processing technique to produce improved focal depth in light microscopy, Image Analysis & Stereology 19 (3) (2011) 163–167.
[3] W. Huang, Z. Jing, Evaluation of focus measures in multi-focus image fusion, Pattern Recognit. Lett. 28 (4) (2007) 493–500.
[4] R. Redondo, F. Sroubek, S. Fischer, G. Cristóbal, Multifocus fusion with multisize windows, in: Proc. SPIE, 5905, 2005, p. 59091B.
[5] R. Redondo, F. Sroubek, S. Fischer, G. Cristóbal, Multifocus image fusion using the log-gabor transform and a multisize windows technique, Inf. Fus. 10 (2) (2009) 163–171.
[6] B. Forster, D. Van De Ville, J. Berent, D. Sage, M. Unser, Complex wavelets for extended depth of field: a new method for the fusion of multichannel microscopy images, Microsc. Res. Tech. 65 (1–2) (2004) 33–42.
[7] Y. Liu, S. Liu, Z. Wang, Multi-focus image fusion with dense sifl, Inf. Fus. 23 (2015) 139–155.
[8] M.B.A. Haghighat, A. Aghagolzadeh, H. Seyedarabi, Real-time fusion of multi-focus images for visual sensor networks, in: Machine Vision and Image Processing (MVIP), 2010 6th Iranian, 2010, pp. 1–6.
[9] S. Kothirummal, H. Nagahara, C. Zhou, S.K. Nayar, Flexible field of depth photography, IEEE Trans. Pattern Anal. Mach. Intell. 33 (1) (2011) 58–71.
[10] V. Aslanas, A depth estimation algorithm with a single image, Opt. Express 15 (8) (2007) 5024–5029.
[11] M.T. Mahmood, T.-S. Choi, Focus measure based on the energy of high-frequency components in the s transform, Opt. Lett. 35 (8) (2010) 1272–1274.
[12] W. Huang, Z. Jing, Multi-focus image fusion using pulse coupled neural network, Pattern Recognit. Lett. 28 (9) (2007) 1123–1132.
[13] F. Aguet, D. Van De Ville, M. Unser, Model-based 2-d deconvolution for extended depth of field in brightfield microscopy, IEEE Trans. Image Process. 17 (7) (2008) 1144–1153.
[14] M. Moeller, M. Benning, C. Schönlieb, D. Cremers, Variational depth from focus reconstruction, IEEE Trans. Image Process. 24 (12) (2015) 5369–5378.
[15] S. Suwajanakorn, C. Hernandez, S.M. Seitz, Depth from focus with your mobile phone, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3497–3506.
[16] J.R. Alonso, A. Fernández, J.A. Ferrari, All-in-focus image reconstruction under severe defocus, Opt. Lett. 40 (8) (2015) 1671–1674.
[17] J.R. Alonso, A. Fernández, J.A. Ferrari, Reconstruction of perspective shifts and refocusing of a three-dimensional scene from a multi-focus image stack, Appl. Opt. 55 (9) (2016) 2380–2386.
[18] X. Yin, G. Wang, W. Li, Q. Liao, Large aperture focus stacking with max-gradient flow by anchored rolling filtering, Appl Opt 55 (20) (2016) 5304–5309.
[19] A. Levin, D. Lischinski, Y. Weiss, A closed-form solution to natural image matting, IEEE Trans. Pattern Anal. Mach. Intell. 30 (2) (2008) 228–242.
[20] V. Vaish, A. Adams, The (new) stanford light field archive, Computer Graphics Laboratory, Stanford University (2008).
[21] Ad light field dataset, http://ific.unican.org/.
[22] A. Levin, F. Durand, Linear view synthesis using a dimensionality gap light field prior, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010, pp. 1831–1838.
[23] B. He, G. Wang, X. Lin, C. Shi, C. Liu, High-accuracy sub-pixel registration for noisy images based on phase correlation, IEEE Trans. Inf. Syst. 94 (12) (2011) 2541–2544.
[24] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
Guijin Wang received his BS and PhD degrees (with honor) from Tsinghua University, China, in 1998 and 2003 respectively, all in electronic engineering. From 2003 to 2006, he was a researcher at Sony Information Technologies Laboratories. Since October 2006, he has been with the Department of Electronic Engineering, Tsinghua University, China, as an associate professor. His research interests focus on wireless multimedia, depth sensing, pose recognition, intelligent human-machine UI, intelligent surveillance, industry inspection, and online learning.

Wentao Li received his BS degree from Tsinghua University, Beijing, China, in 2014. He is currently working toward his PhD degree in electronic engineering with Tsinghua University, Beijing, China. His research interests include depth sensing technique, 3D imaging, and computational imaging.

Xuanwu Yin received his BS degree from Tsinghua University, Beijing, China, in 2011. He is currently working toward his PhD degree in electronic engineering with Tsinghua University, Beijing, China. His research interests include passive/active depth sensing technique, 3D reconstruction, and computational imaging.

Huazhong Yang received his BS degree from Tsinghua University, China in 1989 in Semiconductor Devices and Physics, and received his MS and PhD degrees from Tsinghua University, China in 1993 and 1998 respectively, all in Circuits and Systems. From 1998, he was a Professor at Dept. of Electronic Engineering, Tsinghua University, China. His research interests focus on wireless sensor networks, green circuits and systems, CMOS RF, Analog, and Mixed-Signal Integrated Circuits.