A Generalized Laplacian of Gaussian Filter for Blob Detection and Its Applications

Hui Kong, Hatice Cinar Akakin, and Sanjay E. Sarma

***Abstract*—In this paper, we propose a generalized Laplacian of Gaussian (LoG) (gLoG) filter for detecting general elliptical blob structures in images. The gLoG filter can not only accurately locate the blob centers but also estimate the scales, shapes, and orientations of the detected blobs. These functions can be realized by generalizing the common 3-D LoG scale-space blob detector to a 5-D gLoG scale-space one, where the five parameters are image-domain coordinates (x, y), scales (σ_x, σ_y), and orientation (θ), respectively. Instead of searching the local extrema of the image's 5-D gLoG scale space for locating blobs, a more feasible solution is given by locating the local maxima of an intermediate map, which is obtained by aggregating the log-scale-normalized convolution responses of each individual gLoG filter. The proposed gLoG-based blob detector is applied to both biomedical images and natural ones such as general road-scene images. For the biomedical applications on pathological and fluorescent microscopic images, the gLoG blob detector can accurately detect the centers and estimate the sizes and orientations of cell nuclei. These centers are utilized as markers for a watershed-based touching-cell splitting method to split touching nuclei and counting cells in segmentation-free images. For the application on road images, the proposed detector can produce promising estimation of texture orientations, achieving an accurate texture-based road vanishing point detection method. The implementation of our method is quite straightforward due to a very small number of tunable parameters.**

***Index Terms*—Blob detection, generalized Laplacian of Gaussian (LoG) (gLoG), nuclei (cell) splitting, scale space, texture orientation estimation, vanishing point detection.**

I. Introduction

In COMPUTER vision society, blob detection refers to visual modules that are aimed at detecting points and/or regions in the image that are either brighter or darker than the surrounding. There are two main classes of blob detectors: 1) differential methods based on derivative expressions called differential detectors [1]–[3] and 2) methods based on local extrema in the intensity landscape called watershed detection [4]. In this paper, we focus on differential detectors and will not consider watershed-based approaches. With more recent terminology used in the field, these detections can also be referred to as interest point operators [5], [6] or, alternatively, interest region operators.

One of the first and also most common differential blob detectors is based on the Laplacian of Gaussian (LoG) [1], i.e., given the Gaussian function

$$G(x, y; \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{1}$$

where σ is the standard derivation. The Gaussian scale-space representation $L(x, y; \sigma)$ of image $f(x, y)$ is

$$L(x, y; \sigma) = f(x, y) * G(x, y; \sigma) \tag{2}$$

where $*$ is the convolution operator. Then, the Laplacian operator ∇^2

$$\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3}$$

can be applied to the Gaussian scale-space representation of an image, or equivalently, the LoG operator (4) is computed first and then convolved with the image to create the LoG scale-space representation

$$\nabla^2 \mathrm{G}(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{\pi\sigma^4} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right). \tag{4}$$

The Gaussian step before the Laplacian one is to smooth an image and to attenuate noise. The Laplacian highlights regions of rapid intensity change and is therefore often used for edge detection [1]. As the scale σ of LoG increases, bloblike structures converge to local extrema at some scale. This intuition motivates using LoG filter in blob detection.

Equation (3) usually results in strong positive responses for dark blobs of extent σ and strong negative responses for bright blobs of similar size. The response of a LoG blob detector strongly depends on the relationship between the size of blob structures and the scale of the Gaussian kernel used for pres-moothing. Specifically, to detect blobs with a radius of s in the image domain, the standard deviation σ of LoG is usually computed as $\sigma = (s - 1)/3$. This is motivated by the "3σ" rule that 99% of the energy of a Gaussian is concentrated within three standard deviations of its mean. The interval that has 99% energy of the Gaussian is called the support of LoG. Therefore, when applying this operator at a single scale, it usually fails in detecting blobs of different sizes in the image domain.

Manuscript received April 9, 2012; revised August 24, 2012; accepted November 5, 2012. Date of publication January 9, 2013; date of current version November 18, 2013. This paper was recommended by Editor D. Tao.

H. Kong and S. E. Sarma are with the Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: huikong@mit.edu; sesarma@mit.edu).

H. C. Akakin is with the Department of Electrical and Electronics Engineering, Anadolu University, Eskişehir 26470, Turkey (e-mail: haticecinarakakin@anadolu.edu.tr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2012.2228639

In order to automatically detect blobs of different (unknown) sizes, a multiscale approach is necessary. According to the scale-space theory [7], the conventional multiple-scale LoG blob detector [3] can locate blobs of different scales by detecting local extrema of the LoG scale-space representation after the scale-normalized LoG operation, where the scale of the detected blob is determined by selecting the one at which the maximum filter response is assumed [2], [3]. However, a postprocessing of blob pruning is necessary due to the fact that a large amount of overlapping blobs are detected.

A couple of variants of LoG blob detector have also been proposed. For example, the difference of Gaussian (DoG) [1] is essentially similar to the LoG and can be seen as an approximation of the LoG operator. However, the DoG can be more efficiently computed than the LoG operation. In a similar fashion as for the Laplacian blob detector, blobs can be detected from the scale-space extrema of DoG. By computing the determinant of Hessian (DoH) [3], [7] of the Gaussian scale-space representation, the DoH method can also detect image blob structures by locating local extrema of the scale-normalized determinant of the Hessian.

The aforementioned differential operators have shown a promising performance in locating the centers of 2-D near-circular blobs in images. However, these operators are limited in detecting blobs with general elliptical shapes and are not able to estimate the orientation of the detected blobs, because the conventional LoG used for blob detection is only rotational symmetric, i.e., the σ is set to be equal for both x and y coordinates. To detect blobs subject to affine transformation, Garding and Lindeberg proposed an affine-adapted 2-D blob detector for 3-D surface inference based on a multiscale descriptor of image structure called the windowed second moment matrix, which is computed with adaptive selection of both scale levels and spatial positions [8], [9]. A promising affine-invariant interest point detector has been proposed by a hybrid Laplacian and DoH operator (Hessian–Laplace) [6] to find affine-invariant regions for image matching and object recognition. These points can be obtained by applying affine shape adaptation to a blob descriptor, where the shape of the smoothing kernel is iteratively warped to match the local image structure around the blob. A similar affine-invariant work in feature detection and extraction is proposed in [21]. A detailed comparison among affine region detectors is given in [22].

In this paper, we propose a set of generalized LoG (gLoG) filters to improve blob detector performance. Specifically, our contributions are summarized as follows: *First*, we develop a set of generalized (including both isotropic and anisotropic) LoG kernel operators, which can be implemented based on the second-order derivatives of the generalized 2-D Gaussian. *Second*, to accurately estimate blob size and shape, we show that scale normalization is also necessary for the gLoG filters by giving an analysis that the strength of gLoG filter response at blob regions decreases with both increasing bandwidths of the filter. Therefore, a log-scale-normalization step is proposed. *Third*, we propose to locate blob centers by detecting the local extrema of an intermediate map, which is obtained by aggregating the convolution responses from a set of automatically selected gLoG filter banks. *Finally*, we propose two types of applications of our gLoG blob detector. For biomedical application, it is used for detecting cell nuclei and for splitting touching cells (for counting nuclei furthermore). For natural-image application, we propose a texture-based road vanishing point detection method based on the estimated orientation of road texture by gLoG. This paper is best viewed in color. The high-resolution versions of the images used in this paper and supplemental results are available at http://web.mit.edu/huikong/www/Generalized_LoG.html.

II. gLoG Blob Detector

A. Generation of gLoG Kernels

Let $\mathrm{G}(x, y)$ be the 2-D Gaussian function (kernel) with the following general form:

$$\mathrm{G}(x, y) = \mathcal{A} \cdot e^{-\left(a(x-x_0)^2 + 2b(x-x_0)(y-y_0) + c(y-y_0)^2\right)} \tag{5}$$

where $\mathcal{A}$ is a normalization factor, and x_0 and y_0 are the coordinates of the kernel center. The coefficients a, b, and c explicitly control the shape and orientation of the kernel $\mathrm{G}(x, y)$ by means of θ, σ_x, and σ_y, i.e.,

$$a = \frac{\cos^2\theta}{2\sigma_x^2} + \frac{\sin^2\theta}{2\sigma_y^2} \tag{6}$$

$$b = -\frac{\sin 2\theta}{4\sigma_x^2} + \frac{\sin 2\theta}{4\sigma_y^2} \tag{7}$$

$$c = \frac{\sin^2\theta}{2\sigma_x^2} + \frac{\cos^2\theta}{2\sigma_y^2}. \tag{8}$$

Without losing generality, we let x_0 and y_0 be zero in the following sections. Therefore, the 2-D Gaussian function turns into

$$\mathrm{G}(x, y) = \mathcal{A} \cdot e^{-(ax^2 + 2bxy + cy^2)}. \tag{9}$$

The Laplacian ∇^2 of a 2-D signal $f : R^2 \to R$ is given by (3). As usual, the Laplacian is not directly applied to an image. Instead, it is desirable to smooth the image by convolving it with a Gaussian kernel beforehand. In gLoG, the Gaussian kernel used for convolution with the 2-D signal is generalized to the one given by (9), and the gLoG-filtered image can be represented as

$$\nabla^2 \left[\mathrm{G}(x, y) * \mathrm{f}(x, y)\right] = \nabla^2 \left[\mathrm{G}(x, y)\right] * \mathrm{f}(x, y) \tag{10}$$

where the equation is satisfied due to the following fact:

$$\begin{aligned} \frac{d}{dt}\left[\mathrm{h}(t) * \mathrm{f}(t)\right] &= \frac{d}{dt}\int \mathrm{f}(\tau)\mathrm{h}(t-\tau)d\tau \\ &= \int \mathrm{f}(\tau)\frac{d}{dt}\mathrm{h}(t-\tau)d\tau \\ &= \mathrm{f}(t) * \frac{d}{dt}\mathrm{h}(t). \end{aligned} \tag{11}$$

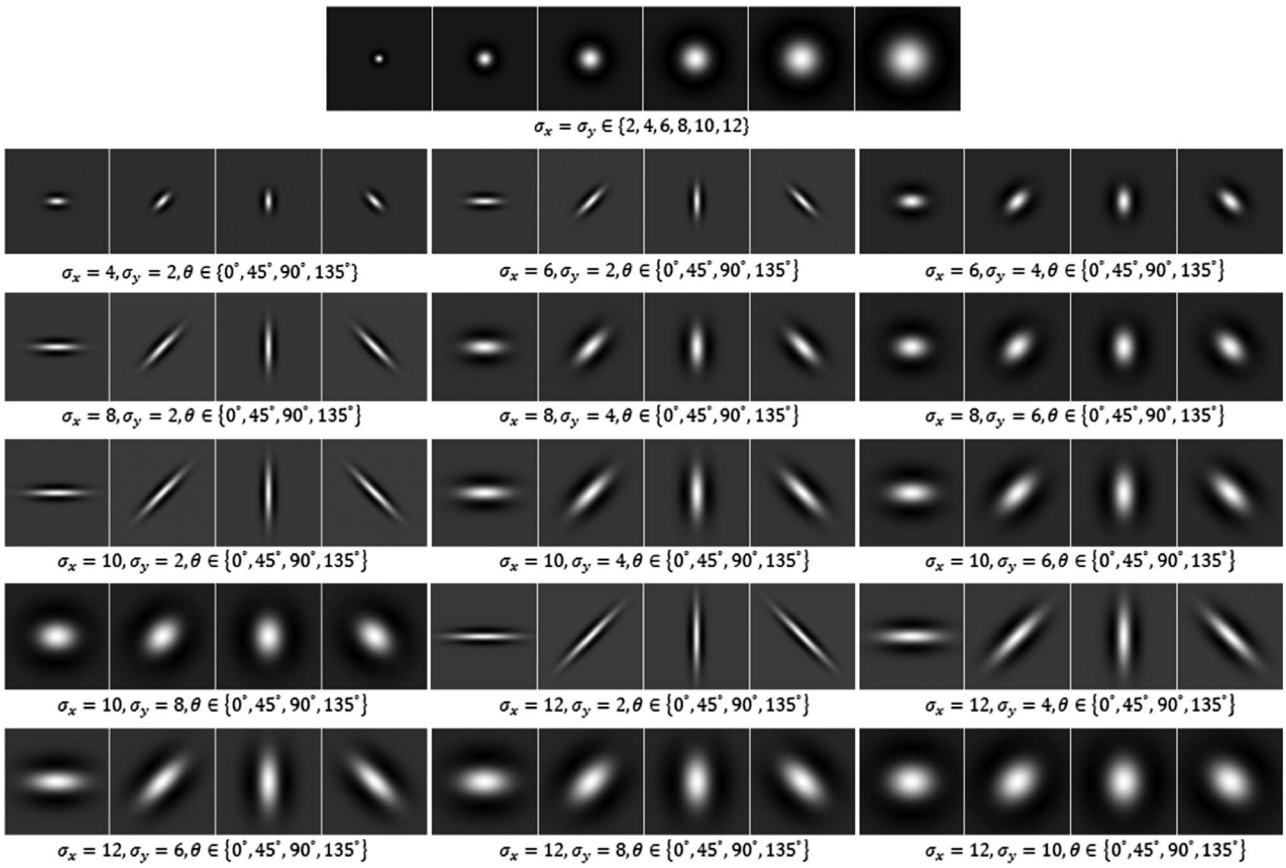


Fig. 1. gLoG kernels.

Therefore, we can obtain the gLoG $\nabla^2 \mathrm{G}(x, y)$ first and then convolve it with the input image. To do so, we need to compute

$$\nabla^2 \mathrm{G}(x, y) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}. \tag{12}$$

By inserting (9) into (12), we get $\partial^2 G/\partial x^2$ and $\partial^2 G/\partial y^2$, respectively, as follows:

$$\frac{\partial^2 G}{\partial x^2} = \mathcal{A}\left[(2ax + 2by)^2 - 2a\right] \cdot e^{-(ax^2 + 2bxy + cy^2)} \tag{13}$$

$$\frac{\partial^2 G}{\partial y^2} = \mathcal{A}\left[(2bx + 2cy)^2 - 2c\right] \cdot e^{-(ax^2 + 2bxy + cy^2)}. \tag{14}$$

Figs. 1 and 2 shows the gLoG and generalized Gaussian kernels with different scales and orientations. Compared with the conventional rotational-symmetric LoG kernels, intuitively, the proposed gLoG operators can be readily applicable to detect blobs with general elliptical shapes. The blob detection (with orientation estimation) can be modeled as a pattern matching process, which matches the proposed gLoG kernels of different scales and orientations to the given image patterns. This can be achieved by convolving the given image pattern with a set of predefined gLoG kernels and then determining the shape and orientation of the detected blob from the maximum convolution response over the different kernels. However, an additional normalization step is needed before the convolution responses can be used. This is due to the following observation: The signal amplitude of the gLoG convolution at blob regions decreases monotonically with the increasing of both scales.

Fig. 3 shows the necessity of scale normalization using a real-world image example, where the two blobs, blob1 and blob2, are highlighted by two red ellipses. The two blobs are convolved with a set of 66 gLoG kernels (those in Fig. 1) at their center locations. The convolution responses for blob1 and blob2 are shown in Fig. 4(a) and (b), respectively. The order of convolution with the 66 gLoG kernels is listed in Table I along with the scales and orientations of the kernels.

We set σ_y no larger than σ_x because σ_x and σ_y are interchangeable in kernel construction. When σ_x is equal to σ_y, the LoG kernel is rotational symmetric, and θ can be set to any value. For convenience, we set it to zero. Apparently, from Fig. 4(a) and (b), the gLoG-convolved signals at the two blob centers, in general, have a decreasing magnitude with the increasing of both scales. Note that the intermediate local perturbations (local extreme) are caused by either the orientation (θ) change of the kernels or by the abrupt increase/decrease of either σ_x or σ_y. Even when convolved with the gLoG kernels of the same θ value, we have noticed that the convolution response does not completely monotonically decrease or increase due to the abrupt change in σ_x or σ_y. However, at a certain orientation, we have observed a monotonic decrease in convolution response when both σ_x and σ_y are increasing. Therefore, the scheme, where the blob size (shape) is determined by selecting the σ_x and σ_y at which the maximum gLoG operator response is assumed, is not effective.

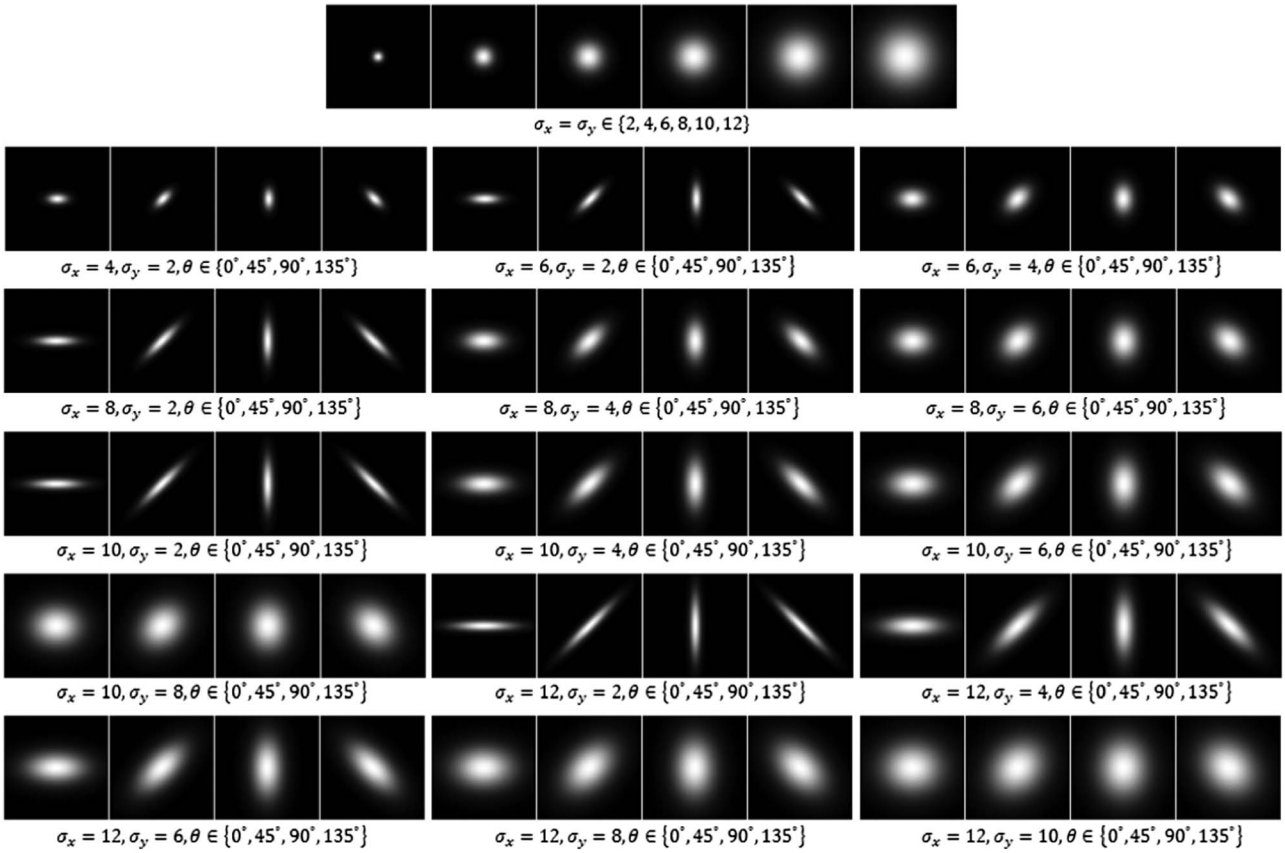


Fig. 2. Generalized Gaussian kernels.

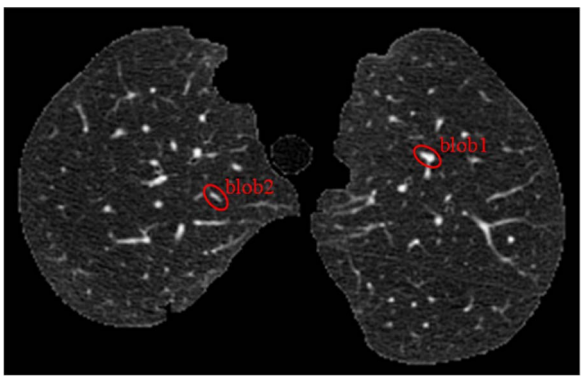


Fig. 3. Examples of blobs (blood vessels) in a computed tomography image, represented by blob1 and blob2, respectively. Note that the orientation of the two blobs is about 135° and the ratio of σ_x and σ_y is about two.

To deal with this problem, we normalize the convolution response by kernel scales. Specifically, let $\nabla^2 I(\cdot; \sigma_x, \sigma_y, \theta)$ be the convolution response of image pattern I with gLoG kernel $\nabla^2 G(\sigma_x, \sigma_y, \theta)$; the scale normalized response is

$$\mathcal{L}_n = (1 + \log(\sigma_x)^\alpha)\,(1 + \log(\sigma_y)^\alpha)\,\bigtriangledown^2 I(\cdot; \sigma_x, \sigma_y, \theta) \quad (15)$$

where α is a positive number, and it can be used to control the blob-center detection accuracy and the eccentricities of the detected blobs. Specifically, a larger α value tends to get a blob estimation with a larger eccentricity and vice versa. Note that we have also tried normalizing the convolution response by the product of σ_x and σ_y, i.e.,

$$\mathcal{L}_n = (\sigma_x)(\sigma_y)\,\bigtriangledown^2 I(\cdot; \sigma_x, \sigma_y, \theta). \quad (16)$$

However, the normalization based on (16) does not give a correct blob shape estimation (see our simple validation in the next paragraph). For convenience, the former normalization method is called log scale normalization, and the latter is called product scale normalization.

We have found that the one shown in (15) produces promising results. Fig. 4(c) and (d) shows the product-scale-normalized convolution responses of blob1 and blob2 by (16), respectively. In contrast, Fig. 4(e) and (f) shows the log-scale-normalized convolution responses of blob1 and blob2 by (15) ($\alpha = 1$), respectively. From Fig. 4(d), we find that the maximum product-scale-normalized convolution response is obtained when convolved with the kernel indexed 21 for blob 1, which is apparently not correct because the aspect ratio of the elliptical shape is around two instead of six. In contrast, we find that the estimated aspect ratio based on (15) is more accurate.

In Fig. 5, we show the examples of blob detection, orientation, and shape estimation based on the aforementioned two scale normalization methods and the complete algorithm given in Section II-B. Visually, the product-scale-normalization method is unable to estimate a blob's orientation and shape. In contrast, the log-scale-normalization method can produce promising results. We also observe that the eccentricities of the detected blobs generally increase when the α value becomes larger.

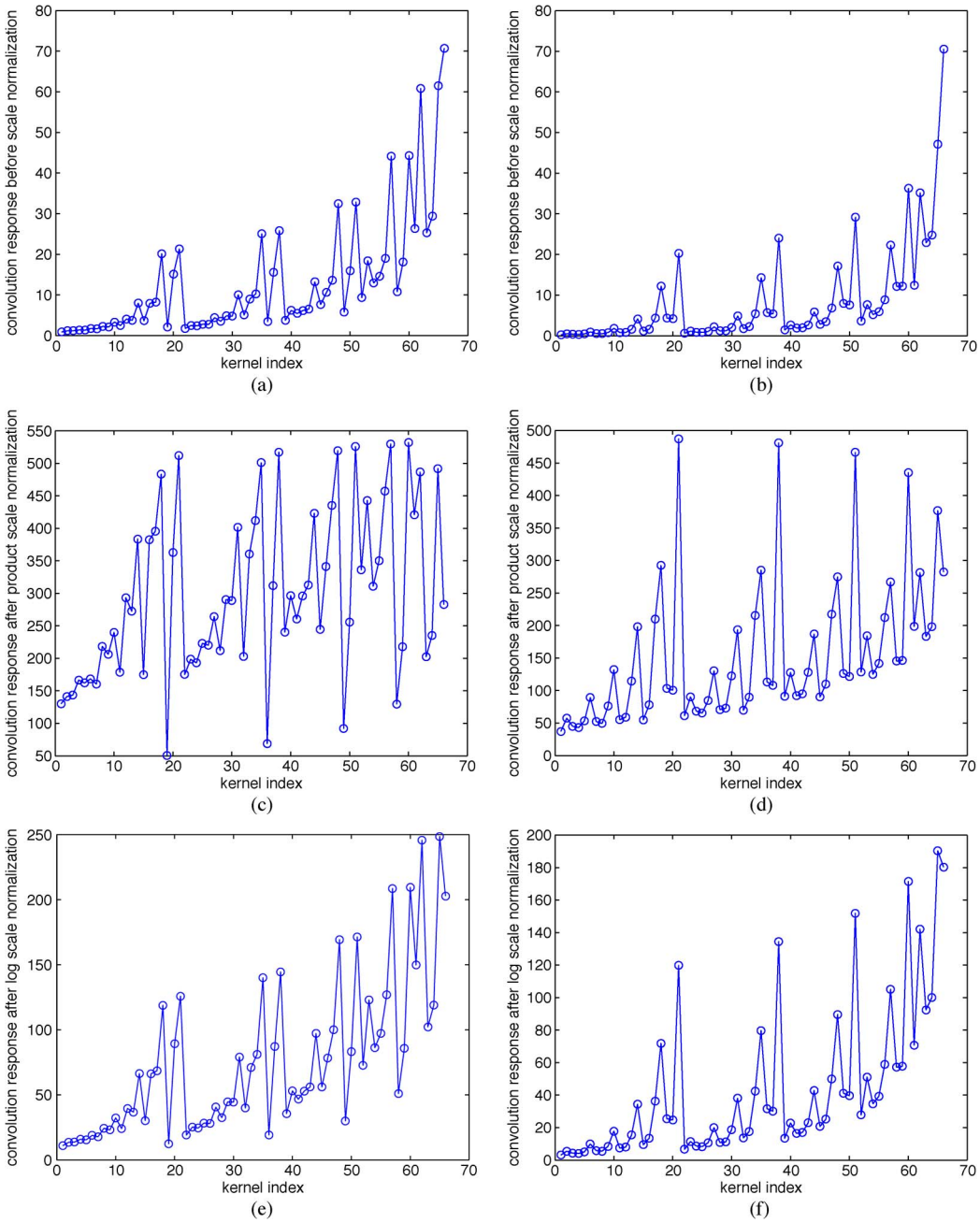


Fig. 4. (a) and (b) Convolution responses before scale normalization at blob1 and blob2, respectively. (c) and (d) Convolution responses after product scale normalization at blob1 and blob2, respectively. (e) and (f) Convolution responses after log scale normalization ($\alpha = 1$) at blob1 and blob2, respectively. Note that the horizontal coordinate represents the index of kernels and the vertical one refers to the convolution response.

TABLE I
INDEX OF gLoG KERNELS USED FOR CONVOLVING WITH BLOB1 AND BLOB2 SHOWN IN FIG. 4

Index	σ_x, σ_y	θ (degree)	Index	σ_x, σ_y	θ (degree)
1	12,12	0	2-5	12,10	0,45,90,135
6-9	12,8	0,45,90,135	10-13	12,6	0,45,90,135
14-17	12,4	0,45,90,135	18-21	12,2	0,45,90,135
22	10,10	0	23-26	10,8	0,45,90,135
27-30	10,6	0,45,90,135	31-34	10,4	0,45,90,135
35-38	10,2	0,45,90,135	39	8,8	0
40-43	8,6	0,45,90,135	44-47	8,4	0,45,90,135
48-51	8,2	0,45,90,135	52	6,6	0
53-56	6,4	0,45,90,135	57-60	6-2	0,45,90,135
61	4,4	0	62-65	4,2	0,45,90,135
66	2,2	0			

B. Blob Detection, Orientation, and Scale Estimation by gLoG

The entire procedure of blob localization, orientation, and scale estimation by gLoG is explained in Table II. The detection of blob centers is carried out in steps 1 through 6. At step 1, we initialize the range of σ_x and σ_y. Specifically, we set σ_x of the gLoG filter to be any integer value between $\sigma_x^{\min}$ and $\sigma_x^{\max}$, where $\sigma_x^{\min}$ and $\sigma_x^{\max}$ are decided based on σ_c (σ_c will be discussed in more detail later). We set σ_y to be the positive integer values that are no larger than σ_x. At step 2, we set the angles that the gLoG filter's orientation can take, e.g., $\theta = \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$. At step 3, we generate a set of n gLoG kernels, K_i, $i = 1, \ldots, n$, based on different combinations of $\{\sigma_x, \sigma_y, \theta\}$. At step 4, we have two options: to detect bright

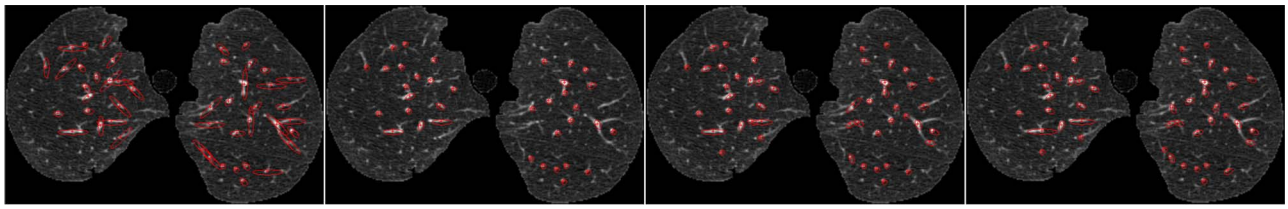


Fig. 5. Examples of blob detection, orientation and scale estimation by using product- and log-scale-normalization, respectively. The first image shows the result by the product-scale-normalization method, and the last three by the log-scale-normalization method ($\alpha = 1, 2, 4$, respectively). Note that the eccentricities of the detected blobs increase with increasing α value.

TABLE II
BLOB LOCALIZATION, ORIENTATION COMPUTING, AND SCALE ESTIMATION BY gLoG

1. Initialize the range of σ_x and σ_y
2. Set the orientation range.
3. Generate a set of n gLoG kernels, $K_i, i = 1, ..., n$.
4. Do log-scale-normalized convolution based on Eq.15 on the original image or negative one to get R_i.
5. Let $R_{sum} = \sum_{i=1}^{n} R_i$.
6. Detect blob centers as the local maxima of R_{sum}.
7. Estimate the orientation of the blob at B_j, $\hat{\theta}_j$
8. Estimate the scale of the blob at B_j, $\{\hat{\sigma}_x^j, \hat{\sigma}_y^j\}$

blobs with a dark surrounding or detect dark blobs with a bright surrounding. If bright blobs with a dark surrounding are to be detected, let $R_i = \mathcal{N}(I * (-K_i))$, where I is the test image, $*$ being the convolution operator, and $\mathcal{N}()$ is the scale normalization operator based on (15). Otherwise, convert the image by $I = 255 - I$ before convolution. At step 5, we compute an intermediate map R_{sum} by integrating the nR_i values. At step 6, the locations of local maximum values of R_{sum} are detected as blob centers, denoted as B_j, $j = 1, \ldots, m$, where m is the number of detected local maxima. At step 7, we estimate the orientation of the blob at each B_j as follows: We can obtain a set of convolution responses, which correspond to a specific θ_i, after we convolve the image with the set of kernels of the same orientation. We denote these sets of convolution responses by $\mathbf{r}_{jk}, k = 1, \ldots, n_\theta$, where n_θ is the number of orientations that θ can assume. The orientation of the detected blob at B_j is computed by $\hat{\theta}_j = (\arg_k \max(\bar{\mathbf{r}}_{jk}) - 1) \times (180^\circ / n_\theta)$, where $\bar{\mathbf{r}}_{jk}$ is the mean value of $\mathbf{r}_{jk}$. Let $\theta_I = \arg_k \max(\bar{\mathbf{r}}_{jk})$ represent the index for the estimated orientation, $\theta_I \in \{1, 2, \ldots, n_\theta\}$. At step 8, once $\hat{\theta}_j$ is obtained, the blob scale can be estimated by $\{\hat{\sigma}_x^j, \hat{\sigma}_y^j\} = \arg_{\sigma_x, \sigma_y} \max(\bar{\mathbf{r}}_{jk} | k = \theta_I)$.

Before Step 1, it is necessary to automatically decide a suitable pair of $\sigma_x^{\min}$ and $\sigma_x^{\max}$. We initialize them based on the circular LoG scale space [3], which is created using a set of σ. Since we are aimed at detecting blobs whose sizes could span a relatively large range, the σ is set to vary between relatively small and large values. Specifically, the σ is generated by $\sigma = \exp(t), t \in \{0.5, 0.7, 0.9, 1.1, \ldots, 3.5\}$. Thus, the smallest σ is $\exp(0.5) \approx 1.65$ (for detecting small blobs with a size of 11×11), and the largest σ is $\exp(3.5) \approx 33$ (for detecting large blobs with a size of 200×200). Next, we find the characteristic scale of the whole image σ_c. The $\sigma_x^{\min}$ is set to the scale which is three steps ahead of σ_c in order. The $\sigma_x^{\max}$ is set to the scale which is three steps after σ_c in order. In other words, if σ_c is computed as $\exp(t_0)$, $\sigma_x^{\min}$ is set to $\exp(t_0 - 0.6)$ and $\sigma_x^{\max}$ is set to $\exp(t_0 + 0.6)$. For example, when $t_0 = 2$ (σ_c is about 7.4), $\sigma_x^{\min}$ and $\sigma_x^{\max}$ are set to 4.1 and 13.5, respectively.

To find σ_c, the largest scale-space value is found as l_g. We also find the smallest value in each scale-space slice (the scale-normalized convolution map with a specific sigma value), denoted by s_l^i. Then, each scale-space slice SS_i is normalized by $SS_i = (SS_i - s_l^i) \times \mathcal{C} / (l_g - s_l^i)$, where $\mathcal{C}$ is a constant. Since the maximum value of the whole scale space is upper bounded by $\mathcal{C}$, we call this process as an upper bounded normalization. For each normalized SS_i, the sum of intensities of the pixels whose values are larger than $0.6 \times \mathcal{C}$ is calculated as ζ_i. The σ_c is selected as the σ (from $\sigma = \exp(t), t \in \{0.5, 0.7, 0.9, 1.1, \ldots, 3.5\}$) which can produce the largest ζ_i. Fig. 6 shows the LoG scale-space representation after upper bounded normalization, where the value of the corresponding σ used for creating the scale space is shown under each scale-space slice image. The σ_c of this example is computed to be approximately 8.17 by our method.

III. APPLICATIONS IN BIOMEDICAL IMAGE ANALYSIS

We apply our gLoG blob detector to pathological and fluorescent images. In quantitative analysis of pathological images, such as grading systems of lymphoma diseases, quantification of features is usually carried out on single cells before grading them by classification algorithms. In fluorescent image analysis, accurate nuclei counting helps to quantify the fluorescence images of certain proteins involved in a cell survival strategy and leads to the identification of an important mechanism of drug resistance in cancer cells. The success of both imaging-based diagnosis systems largely depends on the quality in splitting interweaved cells. To this end, the objective of this study is to use the gLoG method to detect nuclei blobs for an accurate touching-nuclei splitting and counting, both of which are based on accurate nuclei center detection.

A. Nuclei Detection in Pathological and Fluorescent Images

To test the nuclei detection performance of the gLoG detector, we have compared it with three other methods: the original LoG blob detector [3], a scale-selective circular LoG (ssLoG) blob detector [10], and a fast radial-symmetry interest point detector [11]. Note that we did compare our method with the Hessian-affine one proposed in [6]. However, it does not work on our pathological and fluorescent images; therefore, we do not present the results by [6] in our empirical result section. For example, we tested the Hessian-affine code of [6] (available from http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html) on our pathological and fluorescent images. For each image, we tune the parameter (threshold) value from 1 to 1000 with a step of 50 and find that the number of detected blobs decreases as the threshold value increases. Some

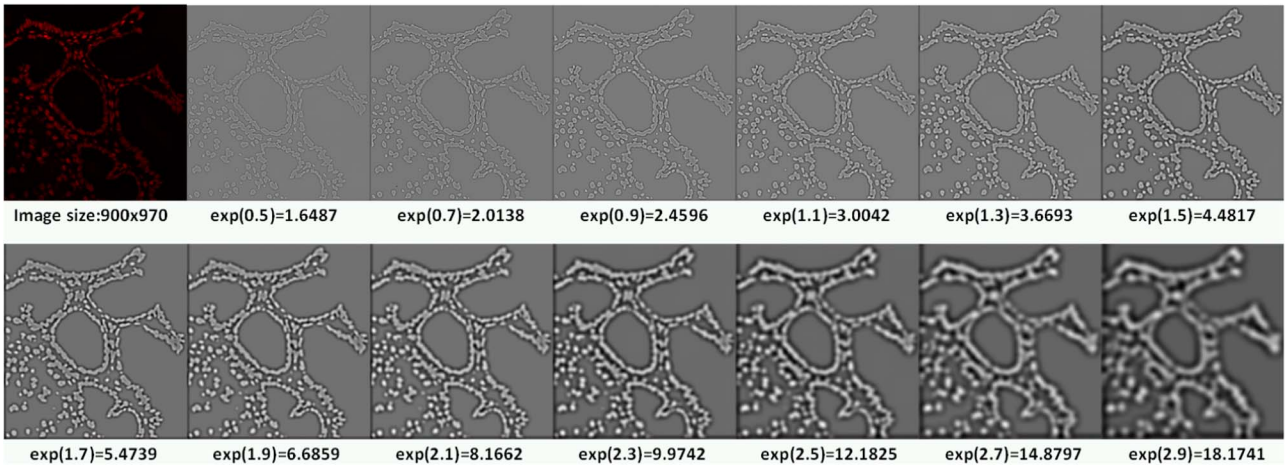


Fig. 6. Estimation of an image's characteristic scale by finding the dominant convolution response map in the upper bound normalized LoG scale space.

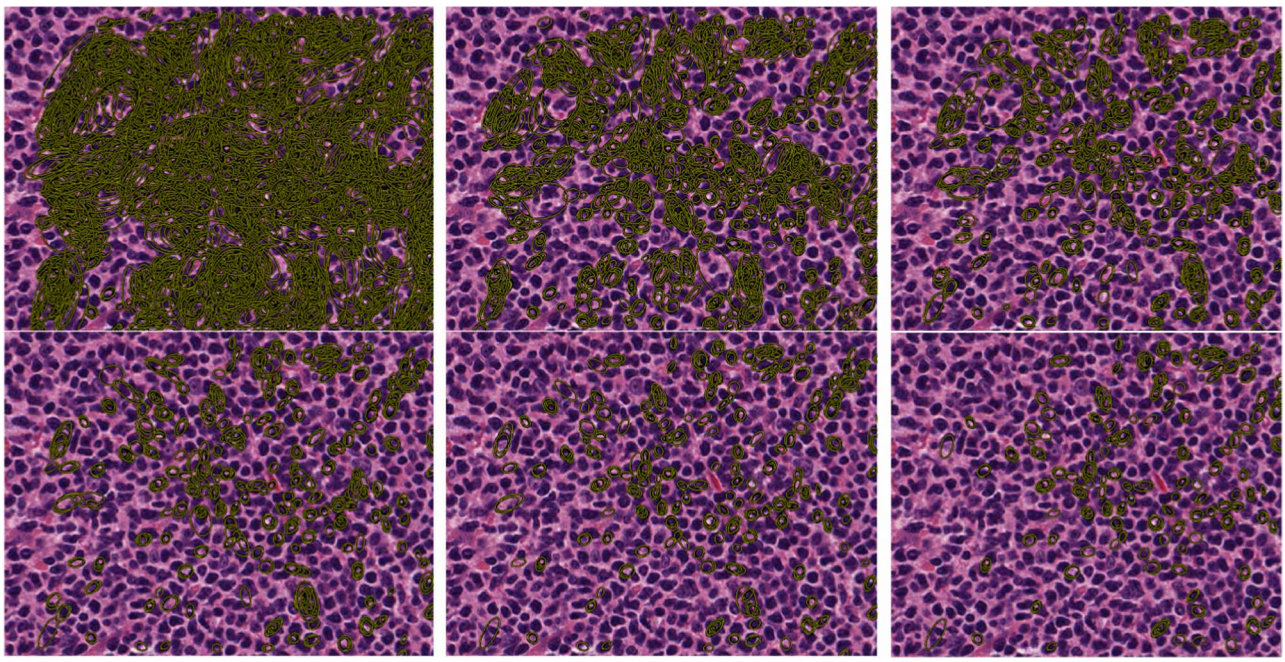


Fig. 7. Blob detection results by the Hessian-affine detector [6] on one image. The threshold values are 1, 50, and 100, respectively, for the top three images, and 150, 200, and 250, respectively, for the bottom three images.

results on an exemplar pathological image are shown in Fig. 7. From this example, we found that the Hessian-affine method in [6] is not applicable to our test images.

The circular LoG blob detector is implemented as follows: The value of t is set between 0.5 and 3 with an increment of 0.2, and σ is computed as $\sigma = \exp(t)$. The circular LoG filters are created based on σ and convolved with the test image. To determine the location and size of blobs, the local maxima in the spatial domain as well as in the scale dimension are found by locating in the LoG scale space those 3-D points which are larger than a preset threshold (set to 0.05 after tuning if we normalize the LoG filter response values to the range of zero to one) and must be larger than their eight immediate neighbors. An example is shown in Fig. 8(a). However, this leads to a lot of overlapping regions (false detection). Therefore, we managed to reduce the number of false detections by a pruning process. Specifically, we prune the blobs based on their pairwise overlapping area: If the ratio between the overlapping area of two blobs and the area of the smaller blob is larger than a threshold (set to 0.25 after tuning), we delete the blob which has a smaller value in the scale-space representation. Fig. 8(b) shows the pruned blobs. Note that our gLoG algorithm does not need the pruning procedure because the blobs are just detected as the local maxima in the obtained aggregated map.

The ssLoG blob detector [10] was proposed recently for cell nuclei detection. Optimal representative scale interval was selected according to the local maxima values of the LoG-filtered cell images. For each scale of the LoG filter, the sum of the top T local maxima values was used as a matching score between the scale of that filter and the cell sizes in the given image. Here, T is set to 10%. The filter scales corresponding to the top three matching scores were selected as the representative scale interval for the given cell image. The local maxima of the aggregate response by the filters with the selected scales are detected as nuclei centers. Like gLoG, the ssLoG method does not need pruning either.

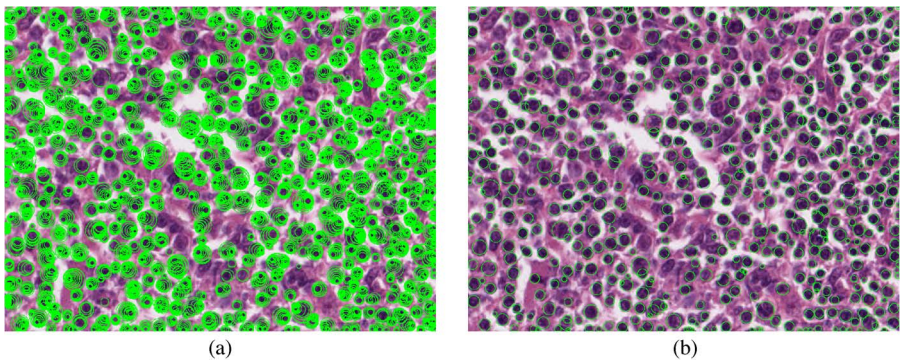


Fig. 8. (a) and (b) Detected blobs by circular LoG blob detector before and after pruning, respectively.

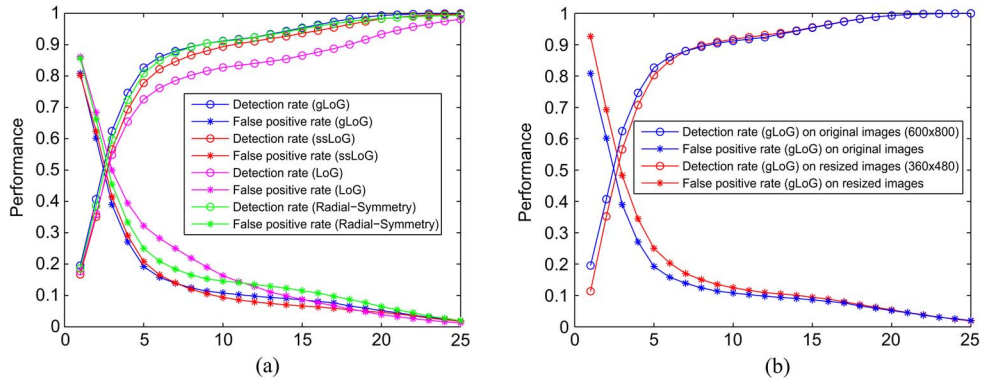


Fig. 9. (a) Average nuclei center detection performance on 15 pathological images: True-detection and false-positive rates. (b) Validation of the adaptiveness of our gLoG for detecting nuclei centers of pathological images at different image resolutions: a comparison between the detection performance on the original 600×800 images and that on the resized 360×480 images. We regard the detection correct if the distance between the detected blob center and ground-truth one is smaller than a threshold (see more text explanation hereinafter). The horizontal axis represents this threshold.

In [11], each pixel in the image votes for symmetry at a given radius based on the orientation of the pixel's gradient. The radial-symmetric center r_i can be found as the corresponding pixel with maximum value in the voting image. The source code of the fast radial-symmetry detection is available online. The problem of [11] is that we need to tune a suitable range of radius for specific types of images to get good performance. After tuning, the radius range for obtaining optimal radial-symmetry detection performance is set to [7] and [13] for the original pathological images and [9] and [15] for the original fluorescent images, respectively.

We have selected 15 pathological images of size 800×600 and seven fluorescent microscopic images of size 1280×1024. The number of nuclei ranges from about 350 to 750 in pathological images and ranges from about 300 to 1300 in fluorescent images. We asked three students to mark each nuclei center, and the centroid is used as the ground-truth nuclei center. For each compared method, the true- and false-detection rates are computed as follows: For each ground-truth center, we look for whether there exists a detection within a scope of a certain radius (number of pixels) of it. If there is one, we view it as a true detection. The true-detection rate is calculated as the number of true detections divided by the number of ground-truth nuclei centers. To compute the false-detection rate, we check whether there is a ground-truth nuclei center within a neighborhood of each detected center. If not, the detection is a false one. The false-detection rate is computed as the number of false detections divided by the number of detected centers.

The nuclei detection performance is calculated as the average result on the 15 pathological images and seven fluorescent images, respectively. Fig. 9(a) shows both true- and false-detection rates with different thresholds for the radius of circular neighborhood on pathological images. We observed that the gLoG filter can achieve the highest blob detection rate with any threshold setting while lowest false-detection rate when the threshold is smaller than nine. The fast radial-symmetry detector can achieve a comparable blob detection rate to our method, but it produces many more false detections than ours. The ssLoG method can achieve a similar false-detection rate, but its detection rate is lower than that of the gLoG and fast radial-symmetry detector. The circular LoG filter is the worst among the four compared methods in terms of true- and false-detection rates. The first row of Fig. 10 shows some examples of detected nuclei centers in pathological images.

To show the adaptiveness of our gLoG detector for detecting nuclei centers of pathological image at different resolutions, we resize the original 600×800 images to smaller versions of 360×480 and apply the gLoG method to the resized images. To make a comparison between the detection performance on the original images and that on the resized images, once we obtained the center coordinates of detected nuclei in the resized images, we resize them by a factor of 5/3. From Fig. 9(b), we observe that the detection performance on the resized (smaller) images is a little worse than that on the original images but still comparable.

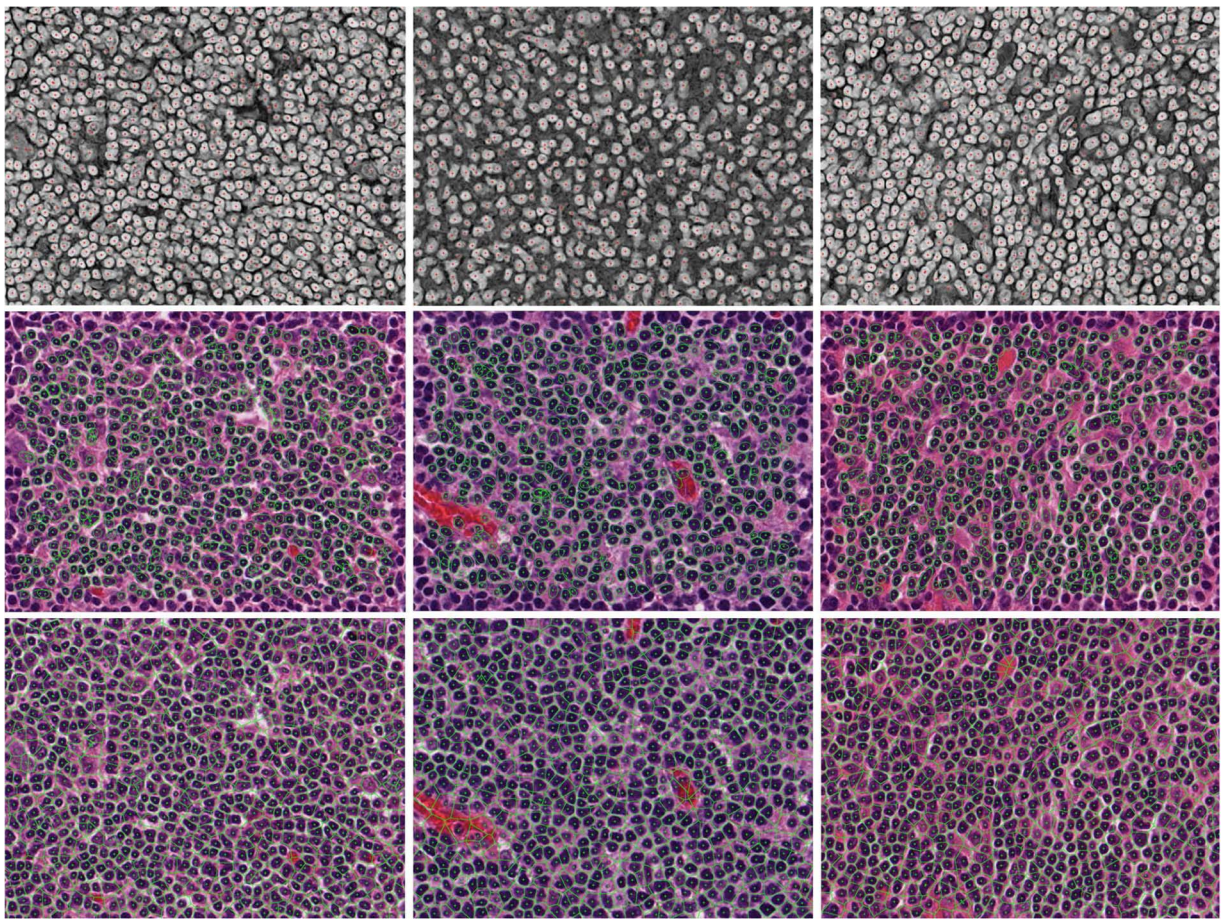


Fig. 10. Example of nuclei splitting and orientation/scale estimation in pathological images. *(First row)* Detected nuclei centers. *(Second row)* Detected blobs at nuclei centers. *(Third row)* Nuclei-splitting results based on a marker-controlled watershed scheme.

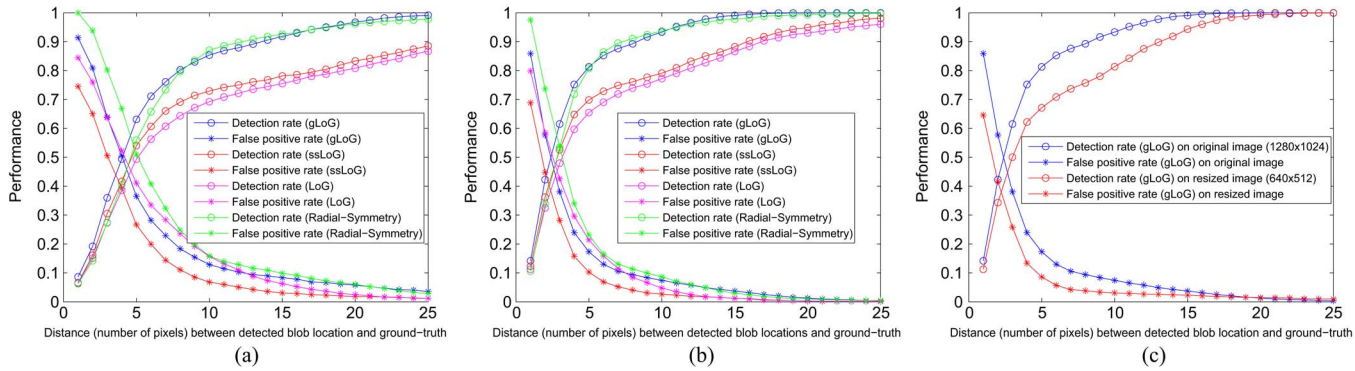


Fig. 11. Blob detection performance comparison between gLoG and the other three methods on seven fluorescent images. (a) and (b) Before and after coordinate normalization, respectively. (c) Validation of the adaptiveness of the gLoG detector for detecting nuclei centers of the fluorescent images at different image resolutions: a comparison between the detection performance on the original 1024 × 1280 images and that on the resized 512 × 640 images. We regard the detection correct if the distance between the detected blob center and the ground-truth one is smaller than a threshold (in pixels). The horizontal axis represents this threshold.

Likewise, we also obtained the nuclei detection performance on fluorescent images, which is shown in Fig. 11(a). Since the size of each test image is 1280 (width) by 1024 (height), both true- and false-detection rates appear to be much worse on the fluorescent images than on the pathological images. Therefore, we made a coordinate normalization, i.e., dividing the coordinates (x and y) of the detected nuclei centers by two scalars (1280/800 and 1024/600), respectively. This normalization is useful for the comparison between the gLoG's performance on the pathological images and that on the fluorescent images. We plot the normalized results in Fig. 11(b), where we observed a consistent promising performance of the gLoG filter on the more challenging fluorescent images. Note that the eccentricity of nuclei is much larger than that in the pathological images. The fast radial-symmetry detector produces a very high false-detection rate since it does not work well for noncircular nuclei. Due to the same reason, the ssLoG and LoG cannot achieve a better true-detection rate than ours, although the ssLoG method has a lower false-detection rate.

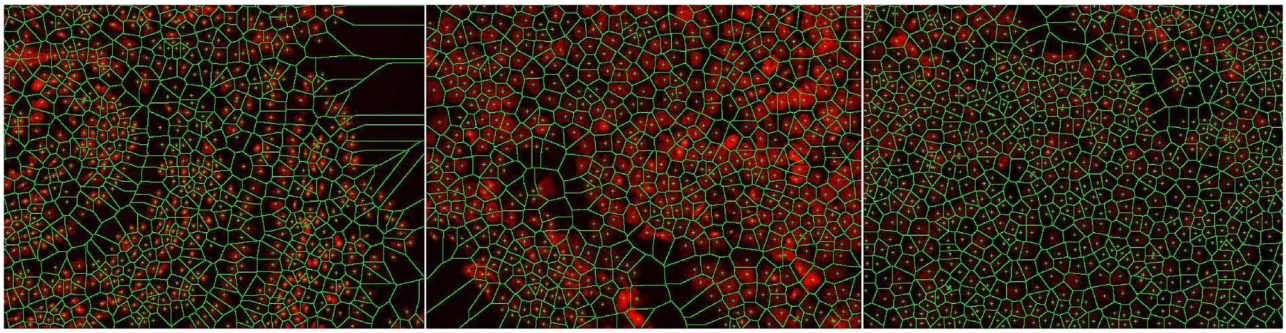


Fig. 12. Examples of cell counting and splitting in three fluorescent microscopic images.

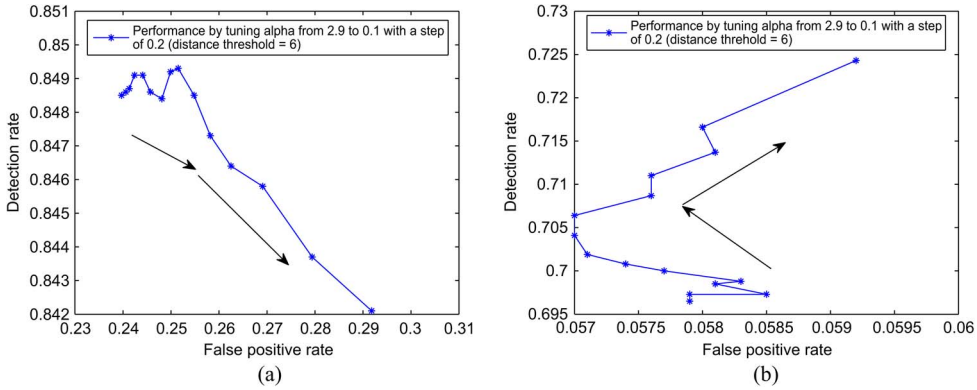


Fig. 13. (a) and (b) Effect of different α values on the nuclei center detection performance on the resized pathological and resized fluorescent images, respectively. The arrows show the direction from the starting point to the ending one.

Similarly, to show the adaptiveness of our blob detector on fluorescent images at different resolutions, we resize the original 1024 $\times$ 1280 images to 512 $\times$ 640 and apply the gLoG detector to the resized images. Likewise, to make a comparison between the detection performance on the original images and that on the resized images, once we obtained the coordinates of the nuclei centers of the resized images, we multiply the factor of two to these coordinates. Then, both the nuclei center coordinates detected on the original and the scaled ones detected on resized images are normalized by dividing two scalars (1280/800 and 1024/600), respectively. From Fig. 11(c), we find that the detection rate on the resized (smaller) images is worse than that on the original images; however, the false-positive rate on the resized images is lower. We show some detection results in fluorescent images in Fig. 12.

As we mentioned earlier, when we introduced (15), different α values could produce different blob detection performances. Fig. 13(a) and (b) shows this effect on the resized pathological and fluorescent images, respectively, where we change the value of α from 2.9 to 0.1 with a step of -0.2. On the resized pathological images, we can observe a significant increase on false-positive rate while the detection rate remains almost unchanged. Note that the coordinates of the detected nuclei centers have been scaled by a factor of 5/3, and the distance threshold is set to six, while on the resized fluorescent images, we can observe a very small change in false-positive rate while the detection rate increases a little. Also, note that the coordinates of the detected nuclei centers have been normalized by dividing 640/800 and 512/600, respectively, and the distance threshold is also set to six.

B. Splitting Touching Nuclei in Pathological Images

Nuclei splitting is a process where the pathological image is split into disjoint regions so that each region contains only one nucleus. We differentiate nuclei splitting from nuclei segmentation. By nuclei segmentation, we mean that the nuclei regions are segmented from the background and other extracellular regions. However, after nuclei segmentation, some nuclei might still be touching each other. Most of the current methods [12]–[17] first carry out nuclei segmentation before splitting touching nuclei and result in a binary foreground/background map based on some popular segmentation or supervised classification methods, e.g., graph cut [18], support vector machine [19], linear discriminant analysis [19], Bayesian classifier [19] or active contour [20] etc.

In contrast, our splitting method does not explicitly segment nuclei regions from the background before splitting. We utilize the detected nuclei centers as markers for a marker controlled watershed splitting [4]. Specifically, we initialize a binary image I_b with ones and let the regions corresponding to the detected blob centers be zeros. Then, a Euclidean distance transform is applied to I_b to get the distance map D_m. Finally, an h-minima controlled watershed (h set to two in our paper) is applied to the compliment of D_m. The third row of Fig. 10 shows some examples of split nuclei. To evaluate the nuclei-splitting accuracy, we use the following types of errors: the undersplitting error, oversplitting error, and encroachment error. The undersplitting error occurs when the algorithm does not place a boundary between a pair of touching nuclei. The oversplitting error occurs when the algorithm places a boundary within a single nontouching cell. The encroachment error

TABLE III
EVALUATION OF SPLITTING PERFORMANCE BASED ON OUR NUCLEI-SPLITTING STRATEGY AND THE METHOD PROPOSED IN [17] (THE LATTER)

Image ID	Number of cells	Correct split	Under-split	Over-split	Encroach. errors
1	593	567/525	16/33	5/15	5/20
2	495	470/444	12/20	7/17	6/14
3	362	332/297	4/14	18/36	8/15
4	719	690/653	15/36	8/18	6/12
5	565	536/502	8/16	13/32	8/15
6	733	709/673	11/30	7/17	6/13
7	464	448/417	8/19	4/15	4/13
8	533	517/485	5/13	8/16	3/19
9	507	493/460	6/14	3/15	5/18
10	718	685/656	15/29	10/19	8/14
11	558	538/507	6/17	6/18	8/16
12	422	409/378	5/17	3/11	5/16
13	518	506/477	6/16	4/15	2/10
14	632	617/581	5/22	3/16	4/13
15	595	574/540	8/21	7/12	6/12
Ave.	561	540/517	8.4/21	7.1/17.4	5.5/14.2

occurs when the splitting algorithm does not correctly place the boundary between a pair of touching nuclei. In other words, it is the error in delineating the true border between two nuclei. Table III gives a comparison between our splitting method and the one proposed in [17] on the 15 pathological images. Our method can produce much fewer splitting errors than [17]: twenty-three fewer on the average.

C. Evaluation on Blob Orientation and Scale Estimation

Altogether, fourteen images with associated ground truth are used to assess the accuracy of orientation and scale estimation. Based on Table II, we use eight-orientation gLoG kernels to estimate the orientation $(\hat{\theta})$ and scales $(\hat{\sigma}_x - \hat{\sigma}_y)$ of each detected blob. Then, $\hat{\theta}$ is further refined by linear interpolation using eight gLoG kernels (i.e., $\{0^\circ, 22.5^\circ, 45^\circ, 67.5^\circ, 90^\circ, 112.5^\circ, 135^\circ, 157.5^\circ\}$ of the same $\hat{\sigma}_x - \hat{\sigma}_y$ pair). The fourteen selected images include ten pathological (original size) and four fluorescent images (original size). In each image, for objectiveness, about 120 evenly sampled seed points are selected. We manually calibrate the orientations and shapes (scales) of the nuclei which coincide with these seed points. Altogether, 1144 valid ground-truth blobs are used for error assessment.

The error in orientation estimation is calculated based on the difference between actual orientation and estimated one in degrees. To show the effect of α on orientation estimation, we change α from 0.1 to 2.9 with a step of 0.3. The orientation estimation performance is listed in Table IV. We find that the orientation estimation performance is not sensitive to the α value. Note that if both the detected blob and the corresponding ground-truth one are circular $(\sigma_x = \sigma_y)$, the error is regarded as zero. However, it would be difficult to measure the error if only either one is circular. We find that most of the detected blobs are close to circular if α is smaller than 0.7. Therefore, we do not consider such cases.

To analyze the error in scale estimation, we denote the ground-truth σ_x by σ_x^g and the ground-truth σ_y by σ_y^g. We

TABLE IV
ORIENTATION ESTIMATION PERFORMANCE WITH RESPECT TO THE MEAN ERROR

α	0.1	0.4	0.7	1.1	1.4	1.7	2.1	2.4	2.7
mean	NA	NA	3.7	3.5	3.4	3.4	3.6	3.6	4.0

TABLE V
SCALE ESTIMATION PERFORMANCE

α	0.7	1.1	1.4	1.7	2.1	2.4	2.7
Percentage	88%	91%	90%	88%	85%	83%	80%

regard the estimated scales for a detected blob correct if both $\hat{\sigma}_x/\sigma_x^g$ and $\hat{\sigma}_y/\sigma_y^g$ are between 0.8 and 1.2 and if the absolute difference between the estimated orientation and the ground-truth one is smaller than 10°. The second condition excludes those situations where the estimated scales are similar while the orientation is totally different. According to the two conditions, we calculate the percentage of the correct scale estimation by changing α. Table V demonstrates the performance.

IV. APPLICATION TO TEXTURE ORIENTATION ESTIMATION FOR VANISHING POINT DETECTION

In this section, we apply the gLoG approach to estimate texture orientation in road images. We treat each image pixel as a blob center and estimate a texture orientation for each pixel. Based on the estimated orientations, we adopt the soft-voting scheme proposed by Kong *et al.* [23], [24] to detect road vanishing point and compare with the Gabor-filter-based method (36 orientations used in Gabor filters) [23], [24] for vanishing point detection.

Initially, we need to decide on an optimal set of gLoG filters, i.e., $\sigma_x^{\max}$, $\sigma_x^{\min}$, and the number of orientations. Empirically, we find that the vanishing point detection performance based on the automatically computed $\sigma_x^{\max}$ and $\sigma_x^{\min}$ is worse than that based on a heuristically selected $\sigma_x^{\max}$ and $\sigma_x^{\min}$ (see comparison in Fig. 16). Therefore, we heuristically set $\sigma_x^{\max}$ and $\sigma_x^{\min}$ to eight and two, respectively. Then, σ_x can be any even integer value from two to eight, and the σ_y can assume any even integer value that is no larger than σ_x and no smaller than two. We use 12 orientations, and the interval between two adjacent orientations is 15°. In the end, we obtained six filters for each specific orientation θ_i, i.e., $\{\sigma_x = 8, \sigma_y = 6, \theta = \theta_i\}$, $\{\sigma_x = 8, \sigma_y = 4, \theta = \theta_i\}$, $\{\sigma_x = 8, \sigma_y = 2, \theta = \theta_i\}$, $\{\sigma_x = 6, \sigma_y = 4, \theta = \theta_i\}$, $\{\sigma_x = 6, \sigma_y = 2, \theta = \theta_i\}$, and $\{\sigma_x = 4, \sigma_y = 2, \theta = \theta_i\}$. We denote the six filters of the same orientation by F_{θ_i}, where $i = 1, \ldots, 12$. Each image is convolved with F_{θ_i}, and six log-scale-normalized convolution response maps are obtained for θ_i. The sum of the six scale-normalized convolution response maps is called the integrated response map for θ_i. Finally, the texture orientation at each pixel is estimated as the one, which can produce the largest integrated response value at that pixel.

Since the gLoG filter can produce extreme response values at bright blob regions, it usually gives a correct estimation of texture orientation in these areas, yet it tends to fail in dark regions. The first row of Fig. 14 shows this situation, where the third image is the estimated orientation map. Since road texture orientation usually changes gradually from one

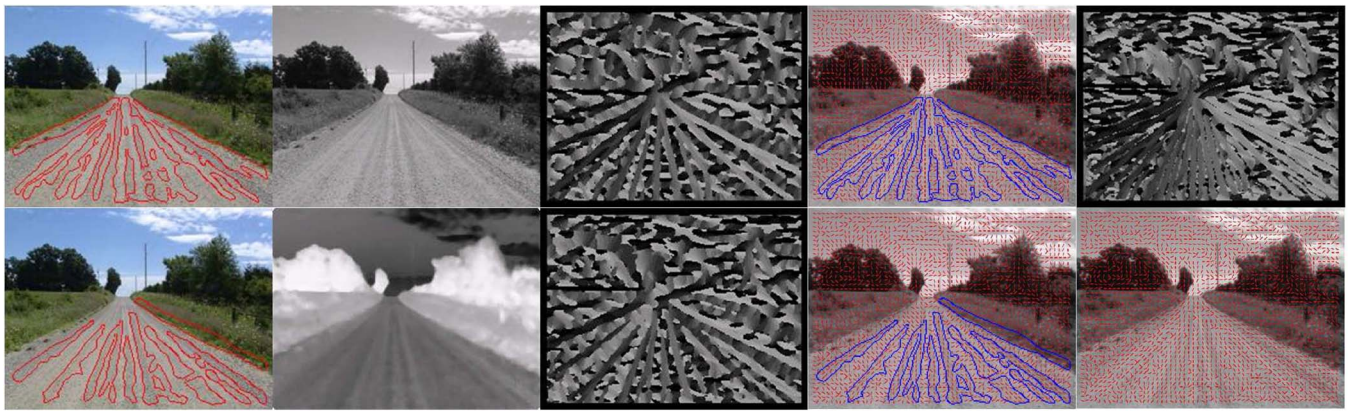


Fig. 14. Illustration of texture orientation estimation in the (top row) gray-scale image and (bottom row) its negative. *(Top row)* First image is the color image where the red areas correspond to bright regions (in contrast to their surroundings), the second image being the gray scale one, the third image being the estimated partial texture orientation map based on the second image, the fourth image being the visual plot of the estimated texture orientation at some evenly sampled locations, with highlight in bright regions, and the fifth image being the complete texture orientation map based on the two partial orientation maps. *(Bottom row)* First image is the color image with the highlighted red areas corresponding to dark regions (also in contrast to their surroundings), the second to fourth images of this row being similar to their counterparts in the top row, and the fifth image showing the displayed orientation bars of the complete texture orientation map at some even sample locations.

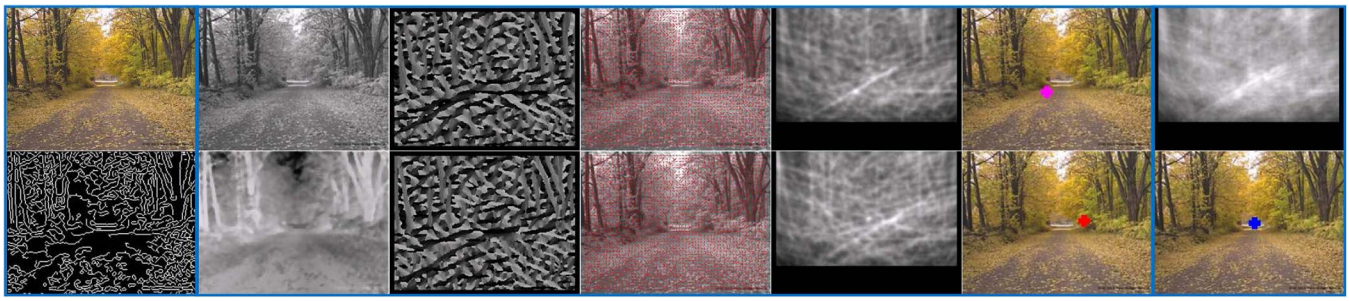


Fig. 15. Illustration of the proposed dual voting maps for detecting road vanishing point. *(First column)* Color and canny-edge images. *(Second column)* Gray image and its negative. *(Third column)* Two estimated partial texture orientation maps. *(Fourth column)* Images overlaid with texture orientation bars at evenly sampled locations. *(Fifth column)* Two voting maps based on the two partial texture orientation maps, respectively. *(Sixth column)* Detected vanishing points based on the two voting maps, respectively. *(Last column)* Fused voting map and detected vanishing point in it.

side to the other, we can also expect a corresponding pixel-intensity change in the true orientation map. However, from the estimated orientation map, we find that the estimation for some parts (dark regions) is obviously not correct. This can also be reflected by the fourth image, where the orientation bars in bright (blue) areas are highlighted. We can observe that the estimated orientations in bright areas are mostly correct and are erroneous in dark regions.

To fully utilize each pixel for road vanishing point detection, we must also correctly estimate texture orientation in dark road regions. Thus, we first get the negative image of the original gray one and apply the same set of gLoG filters to it. Likewise, we illustrate this process in the second row of Fig. 14, where, although the estimation in dark regions seems to be much more accurate than that based on the original gray image, the estimation at bright regions is not. Therefore, neither of the two orientation maps can be a complete estimation, and we thus call them partial texture orientation maps.

To get a correct estimation at both bright and dark regions, we fuse the aforementioned two partial ones. Specifically, we apply the same set of gLoG filters, F_{θ_i}, $i = 1, \ldots, 12$, to both the original gray and negative images. At each pixel, there should be two largest integrated response values (refer to the second paragraph of this section), which correspond to the

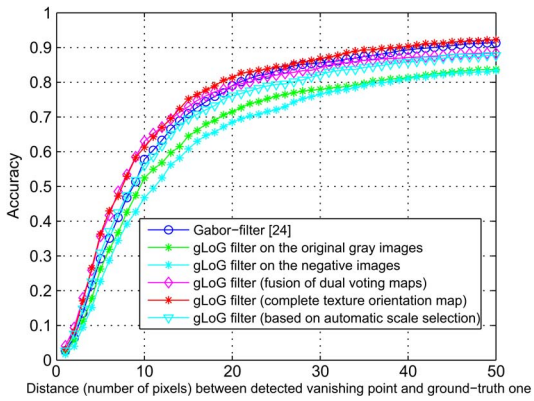


Fig. 16. Road vanishing point detection accuracy: a comparison with Gabor-filter-based method.

estimated texture orientations based on the gray image and the negative one, respectively. If the largest integrated response value obtained based on the original gray image is larger than the one based on the negative image, the texture orientation is estimated as the one which is obtained based on the original gray image. Otherwise, it is estimated as the one which is obtained based on the negative image. In Fig. 14, the last image of the first row is the fused orientation map based on the two

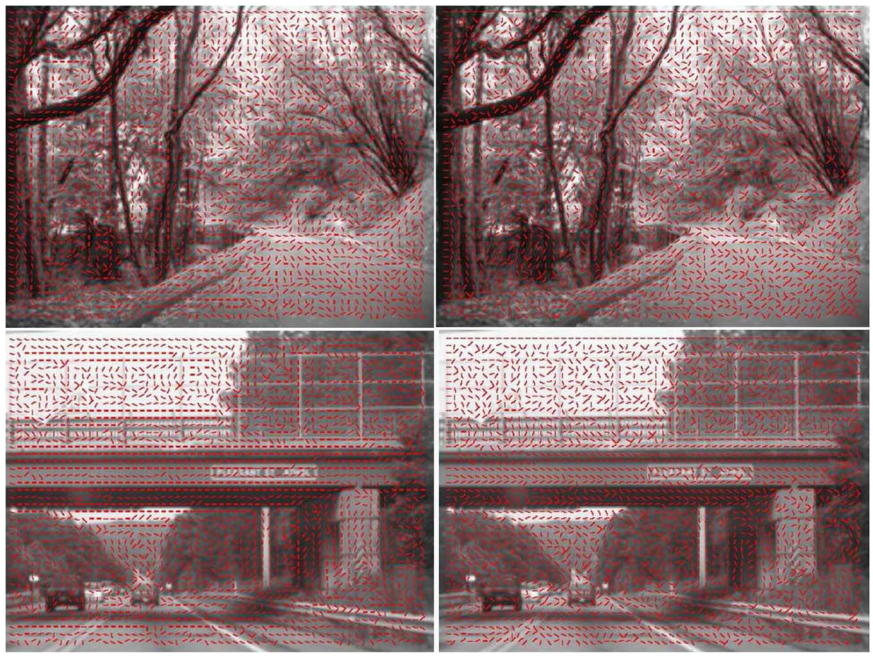


Fig. 17. Visual comparison of texture orientation estimation by Gabor and gLoG filters, respectively. *(First column)* Complete estimation by fusing the two partial maps. *(Second column)* Estimation based on Gabor filter. For a comparison, we can see the difference in the regions corresponding to the road surface, tree bodies, bridge shadow, and fence.

partial ones. We can observe that the intensity change in road region is gradual. In addition, the orientation bars displayed in the last image of the second row also show that the fused orientation map is more accurate. In comparison to the partial ones, it can be called a complete texture orientation map.

To detect road vanishing point, we adopted the soft-voting method proposed by Kong *et al.* [23], [24] once texture orientations are given. We adopt two different schemes in vanishing point detection: The first one relies on the estimated complete texture orientation map, and the other is based on dual voting maps. The first scheme is more straightforward in that the voting map for vanishing point is obtained based on the soft-voting algorithm according to the complete texture orientation map.

For the second scheme, we need to obtain two voting maps, i.e., one is based on the partial texture orientation map which has been obtained from the original gray image, and the other is based on the one obtained from the negative image. We illustrate this process in Fig. 15, where the partial texture orientation map based on the original gray image is shown as the third image of the top row. The first vanishing point voting map is obtained based on this partial orientation map according to [23], [24], shown as the fifth image of the top row. The vanishing point based on the first voting map is shown as the sixth image of the top row. Similarly, we can also get the second partial orientation map (the third image of the bottom row) based on the negative image (shown as the second image of the bottom row). The fifth image of the bottom row shows the corresponding voting map, and the detected vanishing point is shown as the sixth image of the bottom row. As explained, the estimated texture orientations based on the original image and its negative are complimentary, so are the two voting maps. Therefore, we expect to improve vanishing point detection performance by (adding) fusing the two voting maps. The last image of the top row shows the fused voting map. The last image of the bottom row shows the detected vanishing point based on the fused voting map, which is more accurate than either one obtained based on the corresponding partial texture orientation map. Our empirical results on 1003 general road images also validate the effectiveness of the fusion scheme (see Fig. 16).

To visually compare the accuracy of orientation estimation by Gabor and gLoG filters, we have overlaid the plotted texture orientation bars at some evenly sampled locations and displayed them in Fig. 17. We observed that the complete texture orientation map based on the gLoG filter is more accurate than Gabor filter, e.g., on the road surface, tree bodies, the bridge shadows, and fences. Quantitatively, we tested both the gLoG- and Gabor-filter-based methods on 1003 general road images (downloadable from our project page). All images are normalized to the same fixed size of 180×240, and the ground-truth vanishing point locations are provided.

For each image, we compute the distance between the detected vanishing point location and the ground-truth one. If the distance is smaller than a threshold, we regard the detected vanishing point as correct. In this way, we can get a detection rate for a specific threshold value. By changing the threshold from 1 to 50 pixels, we obtain the following statistics, as shown in Fig. 16: the one by the gLoG filter on the original gray image, the one by the gLoG filter on the negative image, the one based on dual voting maps when applying the gLoG filter on dual images, the one based on the complete texture orientation map by fusing dual partial ones, the one based on the automatically selected scales, and the one based on Gabor filter. The six statistics are obtained with the same parameter settings: the soft-voting scheme as proposed in [23] and [24]; however, instead of using a highly confident region as voting area, the voting region for our experiments is set to be the local

image area (whose height is set to 0.4 times the image's height) which is just below each vanishing point candidate pixel. From Fig. 16, we observe that the detection accuracy by gLoG on the original image is better than the one by gLoG on the negative image. The accuracy based on the dual voting maps is much better than either of the aforementioned two and is better than the Gabor-filter-based method when the threshold is smaller than 20. The one based on the complete texture orientation map can achieve the best result among all.

V. Time Cost

The time cost on detecting nuclei and estimating their shapes is larger than that spent on estimating texture orientation. Our algorithm is implemented in Matlab. We tested our code on a Windows PC with a 2.4-GHz CPU and 4G memory. For the pathological images (800×600), the average time cost (including blob detection, orientation, and shape estimation) is 3.5 min/image. The time cost spent on blob detection is only about 25 s/image, and the rest of the time cost is spent on orientation and shape estimation. The time cost of the LoG method in blob detection is approximately the same as our gLoG method because the LoG method needs the extra pruning process. Although the ssLoG radial-symmetrical detectors is faster than the gLoG ones, they cannot estimate blob orientation and shape. For road image application, the time cost of our gLoG method is only 6 s/image for pixelwise orientation estimation. The average time cost for vanishing point detection on an image is 55 s for our slow gLoG method. Note that our code is not optimized yet, e.g., we could replace the convolution operation by FFT and inverse FFT and use some lookup tables to replace some cost-expensive computations. Recently, we have developed an efficient gLoG-based vanishing point detection method [25], which is almost 60 times faster than the method proposed in this paper.

VI. Conclusion

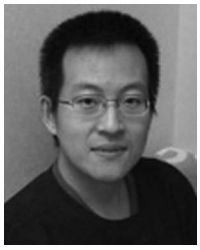
We have proposed a fully automatic framework for detecting blob structures in images by a new gLoG blob detector. The proposed framework is applied to both biomedical and natural images. For the biomedical applications, it is used for splitting touching nuclei in pathological images and counting the number of cells in fluorescent microscopic images. For the application on road images, it can produce a very promising estimation of dominant texture orientations and achieve accurate road vanishing point detection results on 1003 general road images via two different schemes. The detected vanishing points will be used for a constrained road region segmentation in the future.

Acknowledgment

The authors would thank Dr. F. Tang at Hewlett-Packard, CA, for running the Hessian-affine code on the pathological images, and A. Suhre and Y. G. Cinar for creating the ground-truth nuclei centers.

References

[1] R. Gonzales, R. Woods, and S. Eddins, *Digital Image Processing With Matlab*. Englewood Cliffs, NJ: Prentice-Hall, 2004.
[2] T. Lindeberg, "Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention," *Int. J. Comput. Vis.*, vol. 11, no. 3, pp. 283–318, Dec. 1993.
[3] T. Lindeberg, "Feature detection with automatic scale selection," *Int. J. Comput. Vis.*, vol. 30, no. 2, pp. 79–116, Nov. 1998.
[4] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 6, pp. 583–598, Jun. 1991.
[5] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremum regions," in *Proc. British Mach. Vis. Conf.*, 2002, pp. 384–393.
[6] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *Int. J. Comput. Vis.*, vol. 60, no. 1, pp. 63–86, Oct. 2004.
[7] T. Lindeberg, *Scale-Space Theory in Computer Vision*. Norwell, MA: Kluwer, 1994.
[8] T. Lindeberg and J. Garding, "Shape-adapted smoothing in estimation of 3-D shape cues from affine deformations of local 2-D brightness structure," *Image Vis. Comput.*, vol. 15, no. 6, pp. 415–434, Jun. 1997.
[9] J. Garding and T. Lindeberg, "Direct computation of shape cues using scale-adapted spatial derivative operators," *Int. J. Comput. Vis.*, vol. 17, no. 2, pp. 163–191, Feb. 1996.
[10] H. C. Akakin, H. Kong, and M. Gurcan, "A scale-selective log blob detector for nuclei counting," *IEEE Trans. Biomed. Eng.*, 2012, to be published.
[11] G. Loy and A. Zelinsky, "Fast radial symmetry for detecting points of interest," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 959–973, Aug. 2003.
[12] H. Kong, M. Gurcan, and K. Boussaid, "Partitioning histopathological images: An integrated framework for supervised color-texture segmentation and cell splitting," *IEEE Trans. Med. Imaging*, vol. 30, no. 9, pp. 1661–1677, Sep. 2011.
[13] L. Yang, P. Meer, and D. J. Foran, "Unsupervised segmentation based on robust estimation and color active contour models," *IEEE Trans. Inf. Technol. Biomed.*, vol. 9, no. 3, pp. 475–486, Sep. 2005.
[14] P. Yan, X. Zhou, and M. Shah, "Automatic segmentation of high-throughput RNAi fluorescent cellular images," *IEEE Trans. Inf. Technol. Biomed.*, vol. 12, no. 1, pp. 109–117, Jan. 2007.
[15] H. Masmoudi, S. M. Hewitt, N. Petrick, K. J. Myers, and M. A. Gavrielides, "Automated quantitative assessment of HER-2/neu immunohistochemical expression in breast cancer," *IEEE Trans. Med. Imaging*, vol. 28, no. 6, pp. 916–925, Jun. 2009.
[16] K. Mao, P. Zhao, and P.-H. Tan, "Supervised learning-based cell image segmentation for p53 immunohistochemistry," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 6, pp. 1153–1163, Jun. 2006.
[17] Y. Al-Kofahi, W. Lassoued, W. Lee, and B. Roysam, "Improved automatic detection and segmentation of cell nuclei in histopathology images," *IEEE Trans. Biomed. Eng.*, vol. 57, no. 4, pp. 841–852, Apr. 2009.
[18] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2001, pp. 1025–1112.
[19] C. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer-Verlag, 2006.
[20] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 266–277, Feb. 2001.
[21] J. Morel and G. Yu, "ASIFT: A new framework for fully affine invariant image comparison," *SIAM J. Imaging Sci.*, vol. 2, no. 2, pp. 438–469, Apr. 2009.
[22] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. J. Van Gool, "A comparison of affine region detectors," *Int. J. Comput. Vis.*, vol. 65, no. 1/2, pp. 43–72, Nov. 2005.
[23] H. Kong, J.-Y. Audibert, and J. Ponce, "Vanishing point detection for road detection," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 96–103.
[24] H. Kong, J.-Y. Audibert, and J. Ponce, "General road detection from a single image," *IEEE Trans. Image Process.*, vol. 19, no. 8, pp. 2211–2220, Aug. 2010.
[25] H. Kong, S. E. Sarma, and F. Tang, "Generalizing Laplacian of Gaussian filters for vanishing-point detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 1, pp. 408–418, Mar. 2013.



Hui Kong received the Ph.D. degree from Nanyang Technological University, Singapore, in 2007.

From 2008 to 2009, He did his postdoc with the Willow team of the Ecole Normale Superieure and Institut national de recherche en informatique et en automatique (INRIA), Paris. In 2010 and 2011, he was a Research Scientist working on medical image analysis with the Ohio State University Medical Center. He is a Research Scientist with Massachusetts Institute of Technology, Cambridge, working on computer vision projects. His research interests are in computer vision, image processing, and machine learning.

Hatice Cinar Akakin received the Ph.D. degree from the Department of Electrical and Electronics Engineering, Bogazici University, Istanbul, Turkey, in 2010.

From 2004 to 2010, she was a Teaching and Research Assistant with Bogazici University Signal and Image Processing Laboratory. During her Ph.D., she got experience in developing algorithms for face image analysis from low-level image processing to high-level interpretations. She was a Postdoctoral Researcher with The Ohio State University, Columbus. She is currently an Academic Staff with the Department of Electrical and Electronics Engineering, Anadolu University, Eskişehir, Turkey. Her research interests are in the areas of computer vision, image and video analysis, and machine learning.

Sanjay E. Sarma received the Ph.D. degree from the University of California, Berkeley, in 1995.

He is a Professor of mechanical engineering with Massachusetts Institute of Technology (MIT), Cambridge. He is the technology visionary credited with developing many standards and technologies that form the foundation of the commercial radio-frequency identification (RFID) industry. His current research interests are RFID, field robotics, signal processing, sensor networks, etc.

He is the recipient of numerous awards, including the National Science Fundation (NSF) Career Initiation Grant Award, Den Hartog Teaching Excellence Award, Joseph H. Keenan Award for Innovation in Education, New England Business and Technology Award, MIT Global Indus Technovator Award, Boston Business Journal's 40 under 40 Award, *RFID Journal* Special Achievement Award, etc.