
2018
Attune NxT | invitrogen
by Thermo Fisher Scientific
Flow Cytometer
Attune NxT | invitrogen
by Thermo Fisher Scientific
Autosampler
Explore
thermofisher.com/attune

Cytometry PART A
ISAC Journal of Quantitative Cell Science

Object-Oriented Segmentation of Cell Nuclei in Fluorescence Microscopy Images

Can Fahrettin Koyuncu,[1] Rengul Cetin-Atalay,[2] Cigdem Gunduz-Demir[1,3]*

[1]Computer Engineering Department, Bilkent University, 06800, Ankara, Turkey

[2]Graduate School of Informatics, Middle East Technical University, 06800, Ankara, Turkey

[3]Neuroscience Graduate Program, Bilkent University, 06800, Ankara, Turkey

Received 25 December 2017; Revised 14 June 2018; Accepted 30 July 2018

Grant sponsor: Türkiye Bilimler Akademisi

This work was supported by the Turkish Academy of Sciences under the Distinguished Young Scientist Award Program (TUBA GEBIP).

*Correspondence to: Cigdem Gunduz-Demir, Computer Engineering Department, Bilkent University, 06800 Ankara, Turkey
Email: gunduz@cs.bilkent.edu.tr

Published online 13 September 2018 in Wiley Online Library (wileyonlinelibrary.com)

DOI: 10.1002/cyto.a.23594

Abstract

Cell nucleus segmentation remains an open and challenging problem especially to segment nuclei in cell clumps. Splitting a cell clump would be straightforward if the gradients of boundary pixels in-between the nuclei were always higher than the others. However, imperfections may exist: inhomogeneities of pixel intensities in a nucleus may cause to define spurious boundaries whereas insufficient pixel intensity differences at the border of overlapping nuclei may cause to miss some true boundary pixels. In contrast, these imperfections are typically observed at the pixel-level, causing local changes in pixel values without changing the semantics on a large scale. In response to these issues, this article introduces a new nucleus segmentation method that relies on using gradient information not at the pixel level but at the object level. To this end, it proposes to decompose an image into smaller homogeneous subregions, define edge-objects at four different orientations to encode the gradient information at the object level, and devise a merging algorithm, in which the edge-objects vote for subregion pairs along their orientations and the pairs are iteratively merged if they get sufficient votes from multiple orientations. Our experiments on fluorescence microscopy images reveal that this high-level representation and the design of a merging algorithm using edge-objects (gradients at the object level) improve the segmentation results. © 2018 International Society for Advancement of Cytometry

Key terms

object-based representation; fluorescence microscopy imaging; nucleus segmentation; nucleus detection

FLUORESCENCE microscopy imaging is an important tool for in vitro cellular experiments. However, when it is done manually, analyzing fluorescent images in a series of such experiments requires a considerable amount of time and these analyses are prone to observer variability. To facilitate rapid analyses with better reproducibility, it is crucial to implement automated systems, for which cell nucleus segmentation is typically the first and one of the most important steps.

Cell nucleus segmentation in fluorescence microscopy images typically starts with differentiating nuclear pixels from background to obtain a binary mask. For that, it is usually adequate to apply simpler techniques (such as thresholding (1) and clustering (2)) on pixel intensities, since there is a huge intensity difference between foreground and background pixels in fluorescent images. Afterward, segmentation continues with identifying cell nuclei on the binary mask. This is quite straightforward when nuclei appear isolated in an image; each connected component on the binary mask corresponds to a nucleus. In contrast, it becomes challenging to segment nuclei in cell clumps, in which the nuclei appear touching or overlapping in the image. In this case, a connected component should be split into multiple nuclei.

Shape-based methods split one component into multiple nuclei using the fact that a typical nucleus is nearly circular. For that, a large group of them identify markers, each of which will correspond to a nucleus center, by finding regional minima on the inverse distance transform of the binary mask (3,4) or applying

ISAC International Society for Advancement of Cytometry

morphological operations (e.g., erosion) on this mask (5). They then grow the identified markers to delineate nucleus boundaries. Another group employs concavity detection algorithms to find concave points on the mask and split the nuclei from these points (6,7). It has also been proposed to split the mask by identifying circular shapes by the Hough transform (8) and ellipse fitting techniques (9). The shape-based methods usually yield promising results when the degree of overlapping is relatively low so that there is no so much deviation in nucleus appearance from its assumed circular shape. Additionally, when the overlapping degree impedes finding a sufficient amount of background pixels adjacent to the boundary of a nucleus, the distance transform may give misleading results and concavity detection may not work.

Gradient-based methods identify individual nuclei in a cluster relying on the fact that nucleus contours have high contrast differences. The voting-based techniques define kernels to obtain the gradient information and get image pixels voted along the directions specified by these kernels. They then identify regions with larger votes as nucleus centers (10–12). It is also possible to use pixel gradients to detect the markers of a marker-controlled watershed algorithm (13) and grow the markers identified on the distance transforms (14–16). Level set algorithms commonly employ the pixel gradients to refine the nucleus boundaries found by the shape-based methods. These algorithms define their energy functions on the gradients and converge the final boundaries by minimizing these energy functions (17,18). However, the use of the gradient information may not always be adequate to correctly split the overlapping nuclei due to the following imperfections in pixel values. First, pixel intensities may be inhomogeneous in a nucleus. This may cause to define spurious edges inside the nucleus, which results in over-segmentation. Second, pixel intensity differences may not be sufficient at the boundary of two overlapping nuclei. This may create a problem of not being able to define an edge in-between these nuclei, which leads to under-segmentation.

In order to response these issues, this article introduces a new cell nucleus segmentation method that relies on using gradient information not at the pixel level but at the object level. To this end, it proposes to decompose an image into smaller homogeneous subregions, define edge-objects at four different orientations to encode the gradient information at the object level and devise an effective algorithm that segments nuclei by merging the smaller subregions using the edge-objects. In this merging algorithm, the edge-objects vote for subregion pairs along the direction specified by their edge types and the subregion pairs are iteratively merged provided that they get sufficient votes from multiple directions. The main contributions of this object-oriented method are the introduction of representing a fluorescence microscopy image in terms of subregions and edge-objects of different types and the implementation of a new merging algorithm that effectively uses this high-level representation to segment nuclei. As it works on a high-level representation and employs object-level gradients, the proposed segmentation method is expected to be less vulnerable to the aforementioned pixel-level imperfections compared to the existing studies that directly work on the pixel intensities/gradients. Note that our previous study also uses the edge-objects for nucleus segmentation (19). However, this use is completely different than the one proposed in this current study. Our previous study constructs a graph on the edge-objects and achieves segmentation by searching predefined patterns on the constructed graph. It does not define any subregions, and thus, obviously, it does not use them in any merging algorithm in conjunction with the edge-objects.

In the literature, there exist studies that also partition an image into subregions and then form nuclei by merging them. All these studies extract features from the subregions and select the subregions to be merged by solving an optimization problem on the extracted features (20–24). Different from our proposed method, these previous studies do not define any kind of high-level objects encoding the gradient information and they do not employ such high-level objects to merge their subregions. Working on 2661 nuclei, our experiments show that this high-level object-based representation together with the proposed merging algorithm yield better results compared to its pixel-based counterparts.

Methodology

The proposed object-oriented method relies on first dividing an image into over-segmented subregions and then merging them with the help of the edge-objects to segment nuclei. The motivation behind this first-divide-then-merge approach is as follows: in principle, one can locate a single subregion for every nucleus. However, due to nonideal conditions in real life, this may not actually happen for many images and the located subregions are commonly over- or under-segmented. Inhomogeneities inside a nucleus may cause to define multiple over-segmented subregions corresponding to this nucleus, whereas insufficient contrast differences at the boundary of two overlapping nuclei may result in representing the two nuclei with the same under-segmented subregion. Hence, we propose to divide the image into homogeneous subregions that are usually smaller than the average nucleus and merge them afterward.

The merging process employs the edge-objects of four different types that are defined at four different orientations. These edge-objects correspond to the left, right, top, and bottom nucleus boundaries according to the orientation they are defined. In the ideal case, for each nucleus, one can define exactly one left edge-object for its left boundary, one right edge-object for its right boundary, one top edge-object for its top boundary, and one bottom edge-object for its bottom boundary and these edge-objects form a closed curve (this hypothetical case is illustrated in Fig. 1a). In this case, the merging process would be quite simple; one could easily form a nucleus by merging the subregions surrounded by the edge-objects of this nucleus. In contrast, there may exist the following deviations from this ideal case: (i) the edge-objects belonging to the same nucleus may not cover all of its boundaries, and thus, they may not form a closed curve, (ii) more

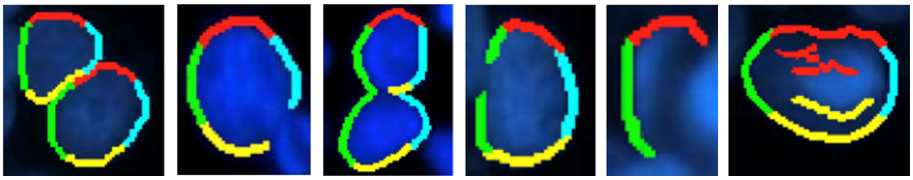

Figure 1. Illustrations of (**a**) the hypothetical case and (**b–f**) the nonideal conditions for the edge-object definition. (b) The edge-objects do not form a closed curve. (c) The same left edge-object is partially shared by two nuclei. (d) Two different left edge-objects are defined for the same nucleus. (e) There is no right and no bottom edge-object defined. (f) Spurious top and bottom edge-objects are defined. Here left, right, top, and bottom edge-objects are shown with green, cyan, red, and yellow, respectively. [Color figure can be viewed at wileyonlinelibrary.com]

than one nucleus may share the same edge-object of the same type, (iii) multiple edge-objects of the same type may correspond to a single nucleus, (iv) the edge-object of at least one type may be missing, and (v) there may exist spurious edge-objects inside a nucleus. We draw illustrations for each of these deviations in Figure 1b–f. To address these nonideal conditions, the proposed method devises an iterative merging algorithm, in which two subregions are merged provided that they share an edge-object for a sufficient number of the edge types. The pseudo-code of the proposed object-oriented algorithm is given in Algorithm 1, and its details are explained in the following subsections. The source codes of its implementation are available at http://www.cs.bilkent.edu.tr/~gunduz/downloads/ObjectOrientedCellSegm/.

Algorithm: NUCLEUS SEGMENTATION Overall framework of the proposed object-oriented algorithm.

Input: image I, superpixel image P, object size threshold t_{size}, maximum distance $d_{\max}$, voting threshold t_{vote}, area threshold t_{area}

Output: segmented nuclei S

1: $B \leftarrow$ OTSUGLOBALTHRESHOLDING (I)

/* *B: binary mask obtained by global thresholding* */

2: **for** perc $\in$ {0.5, 1.0, 1.5, 2.0} **do**

/* *perc: multiplier of the threshold obtained by the Otsu's method for edge-object definition* */

3: $O_{\text{perc}} \leftarrow$ EDGEOBJECTDEFINITION (I, B, t_{size}, perc)

/* O_{perc}*: four sets of the edge-objects obtained for the perc multiplier* */

4: **end for.**

5: $S \leftarrow$ SUBREGIONPARTIONING (P, $O_{0.5}$, B)

6: **for** perc $\in$ {0.5, 1.0, 1.5, 2.0} **do**

7: $S \leftarrow$ SUBREGIONMERGING (S, O_{perc}, $d_{\max}$, t_{vote})

8: **end for**

9: $S \leftarrow$ SMALLNUCLEUSELIMINATION (S, t_{area})

Edge-object Definition

The proposed algorithm defines the following four different types of the edge-objects: left, right, top, and bottom. It derives the edge-objects of each type using a gradient map and a binary mask, both of which are obtained on the L channel[1] of an image. For each type, the gradient map is obtained by convolving the L channel with one of the following Sobel operators.

$$S_{\text{left}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, S_{\text{right}} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix},$$

$$S_{\text{top}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, S_{\text{bottom}} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The same binary mask B is used for all of the edge types and it is obtained by thresholding the L channel with the value automatically calculated by the Otsu's method.[2]

The edge-object definition step defines the left edge-objects as follows. Let G_{left} be the gradient map obtained by convolving the L channel with the Sobel operator S_{left}. First, G_{left} is compared against a threshold t_{left} and pixels with high enough gradients are identified. These identified pixels are masked with the binary mask B and spurious edges on the image background are eliminated. Then, a binary edge map is defined on the remaining pixels. Finally, the m-leftmost pixels of the binary edge map are taken and the connected components of these m-leftmost pixels are considered as the left edge-objects provided that their heights are larger than the size threshold t_{size}. Here, we take the m-leftmost pixels instead of just taking the leftmost pixels since discontinuities may exist in boundaries due to the pixel-based representation of a digital image. In this work, we select $m = 3$ considering the pixel resolution of the images that are used in our experiments. The steps of the left edge-object definition are illustrated in Figure 2.

[1] In this work, we prefer using the *La*b** color space for both edge-object definition and subregion partitioning since it was designed to be perceptually uniform with respect to human color vision. Thus, as the first step, an RGB image is converted to its equivalent in the *La*b** color space and the remaining steps use this converted image.

[2] It is usually sufficient for our method to use a rough binary mask as long as this mask does not miss too many true nucleus pixels. The postprocessing step will correct false pixels up to a certain degree; it will eliminate small regions of false nucleus pixels by small area elimination and fill gaps on false background pixels by majority filtering. Thus, the proposed method uses a quite simple thresholding technique for binarization. However, one may consider to obtain such a mask by employing more advanced methods such as supervised techniques. The investigation of using such techniques could be considered as a future work.

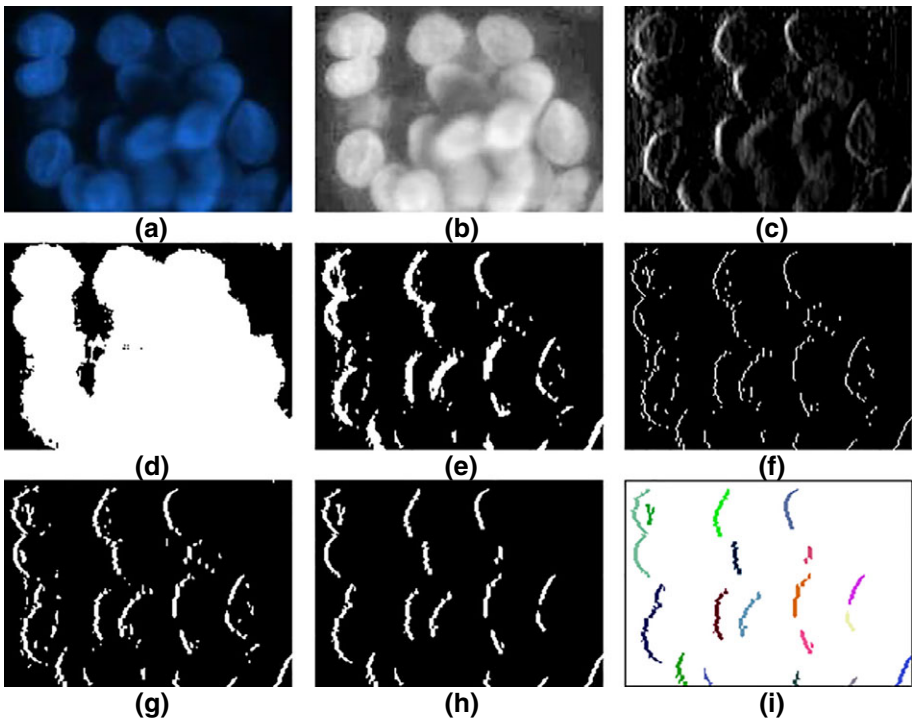

Figure 2. Left edge-object definition on an example subimage: (**a**) original subimage obtained from the HepG2 liver cancer cell line, (**b**) its L channel, (**c**) gradient map G_{left} obtained by convolution, (**d**) binary mask B, (**e**) binary edge map obtained after thresholding G_{left} and masking the result with B, (**f**) leftmost pixels of the binary edge map, (**g**) m-leftmost pixels of the same binary edge map, (**h**) remaining connected components after eliminating the shorter ones, and (**i**) left edge-objects defined for the subimage (each edge-object is shown with a different color). [Color figure can be viewed at wileyonlinelibrary.com]

Here, it is critical to determine a threshold value t_{left} that can identify edge-objects correctly. However, this is not always straightforward and a single threshold may not always work over an entire image especially when the image exhibits variance on the gradient distribution in its different parts. Threshold values smaller than necessary may lead to defining spurious edges whereas too large values may cause to miss some nucleus boundaries. Thus, we propose to use a set of multiple threshold values, for which the subregion merging step is consecutively called one after another (lines 6–7 of Algorithm 1). In particular, a threshold τ is automatically calculated on the gradient map G_{left} by the Otsu's method and then the values starting from the half of this threshold to its double are used. In this work, four sets of the left-edge objects O_{left} are defined using $t_{\text{left}} \in \{0.5\tau, 1.0\tau, 1.5\tau, 2.0\tau\}$[3].

This step defines the objects of the other edge types similarly with the difference that edge-objects shorter than t_{size} are eliminated for the left and right types, whereas those narrower than t_{size} are eliminated for the top and bottom types. At the end of this step, we obtain four sets of the edge-objects $O_{\text{perc}} = \{O_{\text{left}}, O_{\text{right}}, O_{\text{top}}, O_{\text{bottom}}\}$, each of which is calculated using a different threshold percentage constant perc $\in$ {0.5, 1.0, 1.5, 2.0} (lines 2–4 of Algorithm 1).

Subregion Partitioning

The proposed algorithm first partitions an image into homogeneous subregions, which are usually smaller than a typical nucleus, and then merges them with the help of the edge-objects. The first step of this partitioning runs the Simple Linear Iterative Clustering (SLIC) superpixel algorithm (25) on the image. The SLIC algorithm clusters image pixels according to their L, a, and b values in the La^*b^* color space together with their x and y coordinates and defines a superpixel for each of these clusters. After obtaining superpixels by the SLIC algorithm, the second step of this subregion partitioning takes one of the following three actions for each superpixel p.

1. If p is entirely outside the binary mask B, which is also used in the edge-object definition step, it discards this superpixel.
2. Otherwise, if p entirely or partially overlaps with the mask, the overlapping part of p is taken. Then, if any edge-object

[3] We decide to use the threshold percentage *perc* constants of 0.5, 1.0, 1.5, and 2.0 because of the following reasons: Small values for the minimum *perc* cause to define too much spurious edge-objects. On the other hand, large values for the maximum *perc* result in defining almost no useful edge-objects. The use of small intervals in between the consecutive *perc* values increases the computation time without adding too much extra information. Thus, considering all these issues, we select *perc* $\in$ {0.5, 1.0, 1.5, 2.0}.

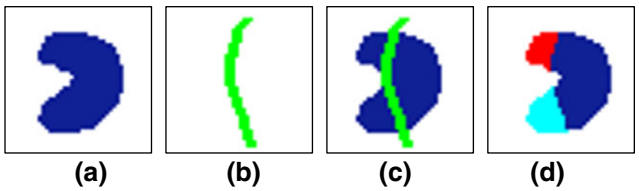

Figure 3. Illustration of splitting a superpixel into multiple subregions: (**a**) a superpixel before splitting, (**b**) a left edge-object that will split the superpixel, (**c**) the left-object superimposed on the superpixel, and (**d**) three subregions obtained after splitting. [Color figure can be viewed at wileyonlinelibrary.com]

cuts this overlapping part into multiple components, the second step further splits p using this edge-object and defines multiple subregions corresponding to the superpixel, as illustrated in Figure 3. It repeats this split operation for all such edge-objects. Note that this second step considers the edge-objects in the set of $O_{0.5}$, which is defined using the lowest Otsu threshold, for further partitioning of the superpixels (line 5 of Algorithm 1).

3. Otherwise, it considers p as one subregion.

At the end of this step, an image is represented with the subregions and the edge-objects of four different types. The next step will merge the subregions using the edge-objects to form nuclei, and the last step will postprocess these nuclei to obtain the final segmentation (Fig. 4).

Subregion Merging

The merging algorithm involves an iterative procedure, each of whose iterations starts with assigning the present subregions to the edge-objects within a distance d. Afterward, based on these assignments, pairs of the adjacent subregions that share a sufficient number of the edge-objects are determined and their merging scores are calculated. Starting from the best one, such pairs are merged with respect to their scores and the pairs are updated also considering the newly emerged subregions. Each iteration continues until there remains no subregion pair to be merged. The next iteration increments the value of d by one and repeats the same steps. This procedure continues its iterations from $d = 1$ to $d_{\max}$. The pseudo-code of this procedure is given in Algorithm 2; the details of its steps are explained below.

In the first step of each iteration, the subregions are assigned to the edge-objects (lines 2–7 of Algorithm 2). For a subregion, this assignment is done separately for each edge type. To assign a subregion s_i to a left edge-object, the vote $v(s_i,o_j)$ that this subregion takes from each left edge-object o_j is calculated and the one with the maximum vote is selected. If $v(s_i,o_j) = 0$ for all o_j, there is no left edge-object assignment for the subregion s_i. To calculate $v(s_i,o_j)$, for each row of o_j, the image is scanned towards right starting from the leftmost pixel of o_j in this row and this vote is incremented by one if the scan meets a pixel of s_i along this row. The scan continues until it hits another object or reaches the distance d. This voting is illustrated in Figure 5. For the other edge types, the

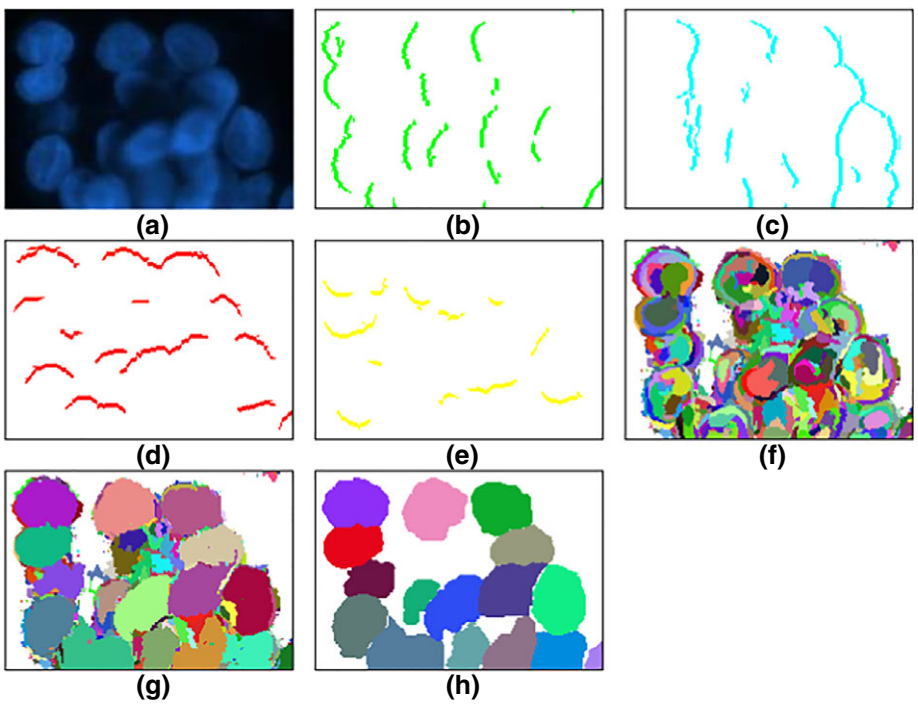

Figure 4. Illustration of the edge-objects and the subregions before and after merging: (**a**) original subimage obtained from the HepG2 liver cancer cell line, (**b**) left edge-objects, (**c**) right edge-objects, (**d**) top edge-objects, (**e**) bottom edge-objects, (**f**) subregions at the end of the subregion partitioning step, (**g**) nuclei obtained by merging the subregions, and (**h**) final segmentation after postprocessing. [Color figure can be viewed at wileyonlinelibrary.com]

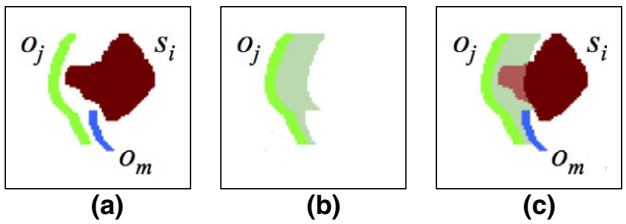

Figure 5. Illustration of calculating the vote $v(s_i,o_j)$ that the left edge-object o_j gives the subregion s_i. (**a**) The subregion s_i and the two edge-objects o_j and o_m are shown in maroon, green, and blue, respectively. (**b**) The area scanned for the rows of o_j, which is shown with light green color. For the upper rows of o_j, the scan continues until it reaches the distance d. For its lower rows, the scan stops earlier since it hits the blue object o_m. (**c**) The part of the subregion s_i that overlaps the scanning area, which is shown with light maroon color. The vote $v(s_i,o_j)$ is incremented by one for each row of this overlapping part. [Color figure can be viewed at wileyonlinelibrary.com]

assignment is done in a similar way with the following differences. The image is scanned row wise for the left and right types and column wise for the top and bottom types. Additionally, the image is scanned toward the opposite direction of the specified edge type.

Algorithm: SUBREGION MERGING Procedure that iteratively merges subregion pairs using the edge-obejcts.

Input: subregions S, edge-objects O, maximum distance $d_{\max}$, voting threshold t_{vote}

Output: merged subregions S

1: **for** $d = 1 \rightarrow d_{\max}$ **do**

/* *d: distance within which a subregion is assigned to the edge-objects* */

2: **for all** subregions s_i of S **do**

3: left(s_i) ← LEFTASSIGNMENT (O, s_i, d)

4: right(s_i) ← RIGHTASSIGNMENT (O, s_i, d)

5: top(s_i) ← TOPASSIGNMENT (O, s_i, d)

6: bottom(s_i) ← BOTTOMASSIGNMENT (O, s_i, d)

/* *left(s_i), right(s_i), top(s_i), and bottom(s_i): left, right, top, and bottom edge-objects assigned to the subregion* s_i */

7: **end for**

8: $\Phi = \emptyset$

/* Φ*: candidate set of subregion pairs for merging* */

9: **for all** adjacent subregions s_i and s_k of S **do**

10: **if** [left(s_i) = left(s_k) **and** $\sigma_{\text{left}}(s_{ik}) \geq t_{\text{vote}}$ **or** right(s_i) = right(s_k) **and** $\sigma_{\text{right}}(s_{ik}) \geq t_{\text{vote}}$] **and** [top($s_i$) = top($s_k$) **and** $\sigma_{\text{top}}(s_{ik}) \geq t_{\text{vote}}$ **or** bottom(s_i) = bottom(s_k) **and** $\sigma_{\text{bottom}}(s_{ik}) \geq t_{\text{vote}}$] **then**

/* $\sigma_{left}(s_{ik})$, $\sigma_{right}(s_{ik})$, $\sigma_{top}(s_{ik})$, and $\sigma_{bottom}(s_{ik})$*: score of merging the subregion pair* s_i *and* s_k *obtained for the left, right, top, and bottom directions* */

11: $\sigma(s_{ik}) \leftarrow \sigma_{\text{left}}(s_{ik}) + \sigma_{\text{right}}(s_{ik}) + \sigma_{\text{top}}(s_{ik}) + \sigma_{\text{bottom}}(s_{ik})$

/* $\sigma(s_{ik})$*: total score of merging the subregion pair* s_i *and* s_k */

12: $\Phi = \Phi \cup \{<s_{ik}, \sigma(s_{ik})>\}$

13: **end if**

14: **end for**

15: **for all** subregion pairs of Φ **do**

16: $s_{ik} \leftarrow$ select the pair with the highest $\sigma(s_{ik})$

/* s_{ik}*: subregion pair with the highest total merging score* */

17: merge the subregions s_i and s_k18: update the set Φ

19: **end for**

20: **end for**

The next step determines the subregion pairs to be merged (lines 8–14 of Algorithm 2). For each edge type, every pair of adjacent subregions that share an edge-object is identified and a score that quantifies the degree of this sharing is calculated. If they do not share an edge-object, this score is set to 0. The subregions s_i and s_k are said to share an edge-object if they are assigned to the same object o_j. In this case, the object o_j will vote for the boundary between the subregions s_i and s_k. This vote calculation is very similar to the aforementioned one, except that each row/column of the object will increment the vote by one if the corresponding scan meets a boundary pixel (instead of a pixel of a subregion) and the scan stops only if it hits another object (instead of also reaching the distance d). After calculating the votes (e.g., the vote $v_{\text{left}}(s_{ik})$ that the shared left edge-object gives to the merge of the subregions s_i and s_k), the merging scores are defined as

$$\sigma_{\text{left}}(s_{ik}) = \text{mean}\left\{\frac{v_{\text{left}}(s_{ik})}{\text{height}(s_i)}, \frac{v_{\text{left}}(s_{ik})}{\text{height}(s_k)}\right\}$$

$$\sigma_{\text{right}}(s_{ik}) = \text{mean}\left\{\frac{v_{\text{right}}(s_{ik})}{\text{height}(s_i)}, \frac{v_{\text{right}}(s_{ik})}{\text{height}(s_k)}\right\}$$

$$\sigma_{\text{top}}(s_{ik}) = \text{mean}\left\{\frac{v_{\text{top}}(s_{ik})}{\text{width}(s_i)}, \frac{v_{\text{top}}(s_{ik})}{\text{width}(s_k)}\right\}$$

$$\sigma_{\text{bottom}}(s_{ik}) = \text{mean}\left\{\frac{v_{\text{bottom}}(s_{ik})}{\text{width}(s_i)}, \frac{v_{\text{bottom}}(s_{ik})}{\text{width}(s_k)}\right\}$$

where each vote is normalized with the size of the subregions. Here, we use normalization not to create any bias towards merging larger subregions, for which the number of boundary pixels is expected to be higher. If the pair of s_i and s_k gets sufficient vote from at least one vertical edge type (left or right) and at least one horizontal edge type (top or bottom), the total merging score $\sigma(s_{ik})$ is calculated as the sum of all of its scores and the pair is added to the merge set Φ. In other words, if $\sigma_{\text{left}}(s_{ik}) \geq t_{\text{vote}}$ or $\sigma_{\text{right}}(s_{ik}) \geq t_{\text{vote}}$ and $\sigma_{\text{top}}(s_{ik}) \geq t_{\text{vote}}$ or $\sigma_{\text{bottom}}(s_{ik}) \geq t_{\text{vote}}$, the merging score $\sigma(s_{ik}) = \sigma_{\text{left}}(s_{ik}) + \sigma_{\text{right}}(s_{ik}) + \sigma_{\text{top}}(s_{ik}) + \sigma_{\text{bottom}}(s_{ik})$. Otherwise, this pair will not be qualified for merging.

As the last step, all pairs in Φ are iteratively merged with respect to their total merging scores (lines 15–19 of Algorithm 2). After merging a pair of the subregions s_i and s_k, the newly emerged subregion s_{ik} is reassigned to the edge-objects, the merging scores between this new subregion and its neighbors are recalculated, and the merge set Φ is updated accordingly. Each iteration continues until there is no subregion pair left for merging.

As explained in the "Edge Object Definition" section, the SUBREGIONMERGING procedure is called for different sets of the edge-objects O_{perc}, each of which is calculated using a different Otsu threshold percentage constant perc $\in$ {0.5, 1.0, 1.5, 2.0} (see lines 6–8 of Algorithm 1). As a final step, subregions smaller than an area threshold t_{area} will be eliminated to obtain the final segmented nuclei. Here, the majority filter is applied afterward to obtain smoother boundaries. Note that this majority filter only slightly affects the segmentation performance, but it yields smoother boundaries. In this work, we select the radius of this filter as 3 considering the pixel resolution of the images used in our experiments.

Experiments

Dataset

We test our object-oriented algorithm on 2,661 cell nuclei of 37 fluorescence microscopy images. The cells were taken from the Huh7 and HepG2 liver cancer cell lines and stained with nuclear Hoechst 33258. The images were taken under a Zeiss Axioscope fluorescent microscope with a Carl Zeiss AxioCam MRm monochrome camera with a 20× Carl Zeiss objective lens. For Hoechst 33258 florescent dye, a bisbenzimide DNA intercalator can be observed in the blue region upon UV region excitation. Hoechst 33258 dye was excited with 365 nm, the emitted blue light (420 nm) was acquired, and the beam splitter was 395 nm. During the image acquisition, binning was set to 1 × 1, the gain and the offset were set to default 0, and the integration time was 10–40 ms. The images were saved in the jpg image format, and their pixel resolution was set to 768 × 1,024.

We use 785 nuclei of 10 randomly selected images (five Huh7 and five HepG2 cell line images) in the training set, on which the model parameters are estimated. The nuclei in the remaining 27 images are used for testing. HepG2 cells tend to grow in more overlayers than Huh7 cells. This leads to more overlapping nuclei in the images of the HepG2 cell line. Thus, we separately test our algorithm for these cell lines. The Huh7 cell line test set includes 891 nuclei of 11 images and the HepG2 cell line test set includes 985 nuclei of 16 images. The nuclei in these images were manually annotated by our biologist collaborator. The image sets and their annotations are publicly available at http://www.cs.bilkent.edu.tr/~gunduz/downloads/NucleusSegData/.

Evaluation

Each algorithm is quantitatively evaluated by calculating the precision, recall, and *F*-score metrics at both the nucleus and pixel levels. The nucleus-level calculation finds true positive nuclei as follows: it matches a nucleus *N* segmented by the algorithm with an annotated nucleus *A* if at least half of the *N*'s pixels overlap with those of *A*. Similarly, it matches each annotated nucleus with a segmented one. It then considers a segmented nucleus as true positive if there exists one-to-one match between this segmented nucleus and an annotated one. Afterward, considering the correctly identified pixels of only the true positive nuclei as true positive pixels, the pixel-level precision, recall, and *F*-score metrics are calculated. Note that in this work, we used the same nucleus- and pixel-level quantitative evaluation with our previous studies (13,19), the results of which will be provided for comparison.

Parameter Selection

Table 1 lists the external parameters of the proposed method. We select the values of these parameters on the training set; in this selection, we do not use the test sets at all. For that, we consider a set of values for each parameter (which are also given in Table 1), take the results for all possible combinations of different parameters, and select the combination that yields the highest *F*-score for the training nuclei. The selected values are $t_{size} = 5$, $d_{max} = 20$, $t_{vote} = 0.1$, and $t_{area} = 400$.

Results and Discussion

Quantitative segmentation results of the proposed object-oriented method and its computational times are given in Table 2, separately for the Huh7 and HepG2 cell line test sets. This table shows that the object-oriented algorithm improves segmentation results at both the nucleus and pixel levels. This improvement is higher for the HepG2 cell line test set, which includes more overlapping nuclei. When the results are visually examined, it is observed that the proposed method is able to determine nucleus locations with high success for both less and more overlapping nuclei (some examples are given in Fig. 6). This is consistent with the nucleus-level evaluation results.

In order to understand its effectiveness, we compare our object-oriented method with two of our previous methods (13,19) and the other three proposed by other research groups (3,10,11). The quantitative and visual results of these comparison methods are also provided in Table 2 and Figure 6. These methods could be grouped into three. The first group includes the adaptive h-minima (3) and iterative h-minima (13) methods, which are marker-controlled watersheds. Both of these methods apply h-minima transform to a distance/gradient map to suppress its noise and then identify the regional minima on the noise-suppressed map as their markers. The former one determines and uses a single h value to identify its markers and adaptively changes it to refine the shapes of the identified markers (3). In contrast, the latter method iteratively identifies its markers using a set of multiple h values (13). Nucleus-level evaluation given in Table 2 shows that using a single *h* value (3) is less efficient to correctly identify many markers, each of which corresponds to a nucleus in the result. When multiple *h* values are used, more correct markers are found, and as a result, the latter comparison method (13) yields nucleus-level evaluation comparable with our method (it gives 89.29 and 83.22% *F*-scores for the Huh7 and HepG2 cell line test sets, respectively, whereas our method gives just 90.75 and 84.21%). In contrast, pixel-level evaluation reveals that the proposed object-oriented method gives much more successful results than both of these comparison methods to delineate the nucleus' boundaries, as also observed in the visual results. For pixel-level evaluation, the

Table 1. A list of the external model parameters together with their values considered in parameter selection

PARAMETER	EXPLANATION	VALUES CONSIDERED
t_{size}	Minimum height/width for a component to be an edge object	{**5**, 10, 15}
d_{max}	Maximum distance within which an edge-object can vote for a subregion	{15, **20**, 25}
t_{vote}	Minimum score that a subregion pair should take from at least one vertical (left or right) and at least one horizontal (top or bottom) edge type to be qualified for merging	{**0.1**, 0.2, 0.3}
t_{area}	Minimum area for a subregion to be a nucleus	{200, 300, **400**, 500}

The selected values are indicated by bold fonts.

proposed method increases the *F*-score of iterative h-minima from 78.46 to 83.98% for the Huh7 cell line test set and from 71.77 to 76.45% for the HepG2 cell line test set. This may be attributed to the following reason. After identifying its markers, a marker-controlled watershed algorithm grows these markers pixel-by-pixel usually with respect to pixel gradients and/or distance transforms. This pixel-by-pixel growing is, however, more susceptible to pixel-level noise and imperfections. In contrast, the proposed object-oriented method relies on subregion-level merging with the help of the edge-objects, which are defined to encode gradients at the object level. This object-level processing results in delineating the nucleus' boundaries more successfully.

The second group of comparison algorithms relies on using pixel-level gradients (10,11). The iterative voting method defines a series of oriented kernels to obtain the gradient information and determines nucleus centers by getting image pixels iteratively voted along the directions specified by these kernels (10). The single-pass voting method improves the nucleus seed detection algorithm by defining a voting area on the eroded binary mask of an image. This method considers only the boundary regions of this binary mask instead of traversing the entire image (11). Table 2 shows that our proposed method leads to higher *F*-scores compared to these two voting-based methods. This indicates the effectiveness of using the gradient information at the object-level instead of using pixel-level gradients. This table also shows that both of these comparison methods yield lower nucleus-level precision and recall values. These lower values indicate the detection of less true positives (correctly located nuclei), but also lower precisions are the indicators of more false positives (incorrectly located nuclei) and lower recalls

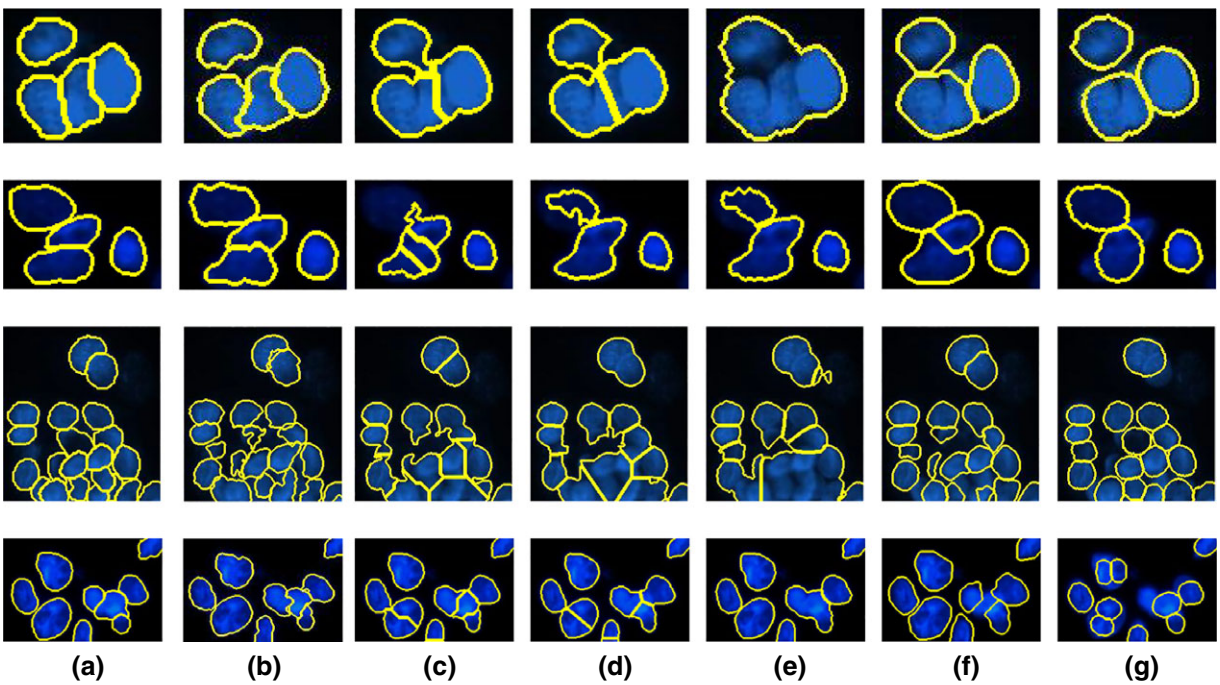

Figure 6. Visual results obtained by the algorithms for various subimages: (**a**) annotated nuclei in the gold standard, (**b**) results by the proposed object-oriented method, (**c**) results by the iterative voting method (10), (**d**) results by the single-pass voting method (11), (**e**) results by the adaptive h-minima method (3), (**f**) results by the ARGraphs method (19), and (**g**) results by the iterative h-minima method (13). The subimage sizes have been scaled for better visualization. [Color figure can be viewed at wileyonlinelibrary.com]

Table 2. Comparison of the algorithms in terms of accuracy measures and computational times on the (a) Huh7 and (b) HepG2 cell line test sets

(a)

	NUCLEUS LEVEL			PIXEL LEVEL			
	PRECISION	RECALL	*F*-SCORE	PRECISION	RECALL	*F*-SCORE	COMPUTATIONAL TIME (S)
Object oriented	92.33	89.23	90.75	80.25	88.08	83.98	14.82 ± 10.38
Iterative voting (10)	81.28	80.92	81.10	81.26	68.48	74.33	10.03 ± 7.96
Single-pass voting (11)	87.82	85.75	86.77	83.61	71.03	76.81	6.52 ± 5.85
Adaptive h-minima (3)	88.27	83.61	85.87	82.57	79.47	80.99	1.81 ± 1.29
ARGraphs (19)	88.14	88.44	88.29	78.28	85.51	81.74	5.86 ± 2.08
Iterative h-minima (13)	89.24	89.34	89.29	83.58	73.93	78.46	2.02 ± 1.44

(b)

	NUCLEUS LEVEL			PIXEL LEVEL			
	PRECISION	RECALL	*F*-SCORE	PRECISION	RECALL	*F*-SCORE	COMPUTATIONAL TIME (S)
Object-oriented	87.00	81.00	84.21	71.96	81.53	76.45	14.11 ± 11.84
Iterative voting (10)	75.89	73.19	74.52	70.67	61.12	65.55	9.42 ± 6.11
Single-pass voting (11)	77.04	72.89	74.91	71.70	59.12	64.80	4.85 ± 4.47
Adaptive h-minima (3)	80.37	69.44	74.50	67.16	66.33	66.74	1.75 ± 1.36
ARGraphs (19)	81.41	79.19	80.28	65.75	75.37	70.24	5.84 ± 2.69
Iterative h-minima (13)	86.35	80.30	83.22	80.09	65.02	71.77	1.38 ± 1.99

are those of more false negatives (missing nuclei). The increase in the number of false positives and false negatives might be the result of noise and imperfections in pixel values, which will directly affect the computation of the gradients at the pixel-level. Intensity inhomogeneities in a nucleus may lead to defining spurious edges, which increase the number of false positives, whereas insufficient pixel intensity differences at the nucleus' boundaries may cause not to identify existing nuclei, which increases the number of false negatives. The use of the gradient at the object-level alleviates the negative effects of these imperfections, which might be the reason of obtaining higher nucleus-level precision and recall values.

The last comparison group includes the ARGraphs method (19), which we implemented in our previous work. This method constructs an attributional relational graph on the edge-objects and identifies nucleus centers by searching predefined patterns on this graph. This previous method also uses the edge-objects, but this use is completely different than the one proposed in this current work. ARGraphs does not define any subregions, and thus, obviously, it does not use these sub-regions in conjunction with the edge-objects. Moreover, it does not make use of any first-divide-then-merge approach to form nuclei from the subregions and the edge-objects. We use this comparison method to understand the effects of developing such kind of approach in segmenting the nuclei. Table 2 demonstrates that this newly proposed approach improves the *F*-scores both at the nucleus and pixel levels. This improvement is higher for the HepG2 cell line test set, in which cells tend to grow in overlayers. This reveals the effectiveness of our first-divide-then-merge approach, which first divides an image into subregions and then merges them by the edge-objects, to more correctly identify overlapping nuclei in more overlayered cell clumps. This is also consistent with the visual results given in Figure 6.

Table 2 also provides the average computational time to segment nuclei in a given image and its standard deviation. These computational times are obtained on a computer with a 2.9 GHz Intel Core i5 processor and 16 GB of RAM. We implement our object-oriented method mostly in Matlab but when faster computations are needed, we write the code in C++ and compile it by the MEX compiler of Matlab. Its average computational time is approximately 15s, which is higher than those of the other comparison methods. The most expensive part of our method is the iterative subregion merging procedure (lines 15–19 of Algorithm 2). At each iteration, this procedure selects one subregion pair from the candidate set, merges them, and updates the assignments and the voting scores of the remaining candidates. This part takes longer time especially when the number of subregions is high. Although it is implemented in C++, it is still possible to make this part faster by more effectively coding it. It is also possible to obtain further speedups by implementing the entire algorithm in C++. This is considered as a future work of this implementation.

Conclusion

This article presents a new object-oriented method for segmenting cell nuclei in fluorescence microscopy images. This method relies on the use of gradient information at the object-level, instead of directly using pixel-level gradients. To this end, it proposes to partition an image into smaller subregions, define edge-objects at four different orientations for encoding

the gradient information at the object-level, and devise an effective merging algorithm that forms nuclei from the subregions with the help of the edge-objects. In this subregion-level merging, the edge-objects vote for subregion pairs along the direction specified by their edge types and the subregion pairs are iteratively merged provided that they get sufficient votes from multiple directions. This high-level representation together with this high-level merging is expected to be less susceptible to pixel-level noise and imperfections compared to the methods that directly work on pixel values. Our experiments on fluorescence microscopy images are consistent with this expectation. They demonstrate that the proposed object-oriented method leads to better segmentation results compared to pixel-level cell nucleus segmentation algorithms.

The proposed method defines four different types for the edge-objects but does not define any type for the subregions. It is also possible to assign types to subregions, based on their characteristics, and incorporate them into the merging algorithm. This could be considered as a future work of this study. In this work, we focus on the fluorescence microscopy images. As another future work, one may consider to extend this object-oriented method to other types of microscopy images. For instance, it can be extended to 3D nucleus segmentation by defining 3D edge-objects and 3D subregions. For that, the third axis (depth) can be employed to identify the edge-objects of six different types (left, right, top, bottom, front, and back object types) and supervoxels can be used to define the subregions instead of superpixels. For obtaining the supervoxels, one can use the option provided by the SLIC algorithm (25). After defining them, the merging step can be modified to consider object assignments for the new types and to use the votes of the edge-objects from six directions. This may be considered as another future work of the proposed study.

Literature Cited

1. Chen X, Zhou X, Wong ST. Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy. IEEE Trans Biomed Eng. 2006;53(4):762–766.
2. Dima AA, Elliott JT, Filliben JJ, Halter M, Peskin A, Bernal J, Kociolek M, Brady MC, Tang HC, Plant AL. Comparison of segmentation algorithms for fluorescence microscopy images of cells. Cytometry Part A. 2011;79A(7):545–559.
3. Cheng J, Rajapakse JC. Segmentation of clustered nuclei with shape markers and marking function. IEEE Trans Biomed Eng. 2009;56:741–748.
4. Jung C, Kim C. Segmenting clustered nuclei using h-minima transform-based marker extraction and contour parameterization. IEEE Trans Biomed Eng. 2010; 57(10):2600–2604.
5. Yang X, Li H, Zhou X. Nuclei segmentation using marker-controlled watershed, tracking using mean-shift, and Kalman filter in time-lapse microscopy. IEEE Trans Circuits Syst I. 2006;11:2405–2414.
6. Chang H, Han J, Borowsky A, Loss L, Gray JW, Spellman PT, Parvin B. Invariant delineation of nuclear architecture in glioblastoma multiforme for clinical and molecular association. IEEE Trans Med Imaging. 2013;32(4):670–682.
7. Bai X, Wang P, Sun C, Zhang Y, Zhou F, Meng C. Finding splitting lines for touching cell nuclei with a shortest path algorithm. Comput Biol Med. 2015;63: 277–286.
8. Maitra M, Gupta RK, Mukherjee M. Detection and counting of red blood cells in blood cell images using Hough transform. Int J Comput Appl. 2012;53(16):13–17.
9. Ge J, Gong Z, Chen J, Liu J, Nguyen J, Yang Z, Wang C, Sun Y. A system for counting fetal and maternal red blood cells. IEEE Trans Biomed Eng. 2014;61(12):2823–2829.
10. Parvin B, Yang Q, Han J, Chang H, Rydberg B, Barcellos-Hoff MH. Iterative voting for inference of structural saliency and characterization of subcellular events. IEEE Trans Image Process. 2007;16(3):615–623.
11. Xu H, Lu C, Mandal M. An efficient technique for nuclei segmentation based on ellipse descriptor analysis and improved seed detection algorithm. IEEE J Biomed Health Inf. 2014;18(5):1729–1741.
12. Xing F, Su H, Neltner J, Yang L. Automatic ki-67 counting using robust cell detection and online dictionary learning. IEEE Trans Biomed Eng. 2014;61(3):859–870.
13. Koyuncu C, Akhan E, Cetin-Atalay R, Gunduz Demir C. Iterative h-minima based marker-controlled watershed for cell nucleus segmentation. Cytometry Part A. 2016; 89(4):338–349.
14. Jeong MR, Ko BC, Nam JY. Overlapping nuclei segmentation based on Bayesian networks and stepwise merging strategy. J Microsc. 2009;235(2):188–198.
15. Yang H, Ahuja N. Automatic segmentation of granular objects in images: Combining local density clustering and gradient-barrier watershed. Pattern Recogn. 2014; 47(6):2266–2279.
16. Lu C, Xu H, Xu J, Gilmore H, Mandal M, Madabhushi A. Multi-pass adaptive voting for nuclei detection in histopathological images. Sci Rep. 2016;6:33985.
17. Maska M, Danek O, Garasa S, Rouzaut A, Munoz-Barrutia A, Solorzano CO. Segmentation and shape tracking of whole fluorescent cells based on the Chan–Vese model. IEEE Trans Med Imaging. 2013;32(6):995–1006.
18. Wu P, Yi J, Zhao G, Huang Z, Qiu B, Gao D. Active contour-based cell segmentation during freezing and its application in cryopreservation. IEEE Trans Biomed Eng. 2015;62(1):284–295.
19. Arslan S, Ersahin T, Cetin-Atalay R, Gunduz-Demir C. Attributed relational graphs for cell nucleus segmentation in fluorescence microscopy images. IEEE Trans Med Imaging. 2013;32(6):1121–1131.
20. Su H, Yin Z, Huh S, Kanade T. Cell segmentation in phase contrast microscopy images via semi-supervised classification over optics-related features. Med Image Anal. 2013;17(7):746–765.
21. Bise R, Sato Y. Cell detection from redundant candidate regions under nonoverlapping constraints. IEEE Trans Med Imaging. 2015;34(7):1417–1427.
22. Arteta C, Lempitsky V, Noble JA, Zisserman A. Detecting overlapping instances in microscopy images using extremal region trees. Med Image Anal. 2016;27:3–16.
23. Zhang C, Yarkony J, Hamprecht FA. Cell detection and segmentation using correlation clustering. In: Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Vol. 8673; 2014. pp. 9–16.
24. Genctav A, Aksoy S, Onder S. Unsupervised segmentation and classification of cervical cell images. Pattern Recogn. 2012;45(12):4151–4168.
25. Achanta R, Shaji A, Smith K, Lucchi A, Fua P, Susstrunk S. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans Pattern Anal Mach Intell. 2012;34(11):2274–2281.