Segmentation of Cells from Spinning Disk Confocal Images Using a Multi-stage Approach

Saad Ullah Akram[1,2], Juho Kannala[1], Mika Kaakinen[2,3], Lauri Eklund[2,3], and Janne Heikkilä[1]

[1] Center for Machine Vision Research, University of Oulu, Finland
[2] Biocenter Oulu, University of Oulu, Finland
[3] Faculty of Biochemistry and Molecular Medicine, University of Oulu, Finland
{sakram,jkannala,jth}@ee.oulu.fi, {mika.kaakinen,lauri.eklund}@oulu.fi

Abstract. Live cell imaging in 3D platforms is a highly informative approach to visualize cell function and it is becoming more commonly used for understanding cell behavior. Since these experiments typically generate large data sets their analysis manually would be very laborious and error prone. This has led to the necessity of automatic image analysis tools. Cell segmentation is an essential initial step for any detailed automatic quantitative analysis. When the images are captured from the 3D culture containing proliferating and moving cells, cell-cell interactions and collisions cannot be avoided. In these conditions the segmentation of individual cells becomes very challenging. Here we present a method which utilizes the edge probability map and graph cuts to detect and segment individual cells from cell clusters. The main advantage of our method is that it is capable of handling complex cell shapes because it does not make any assumptions about the cell shape.

1 Introduction

Imaging of living cells in 3D environments is becoming more and more essential tool to experimentally approach the fundamental questions about cell dynamics, molecular regulation of cell migration, cell invasion and cell fates. 3D scaffolds provide more realistic platform for cell cultures with respect to the physical and biochemical properties of the micro-environment compared to 2D models. This aspect is especially important in tumor cell cultures in which the composition of the micro-environment contributes to the cell behavior and drug response [1]. As in 2D cultures the cells can be labeled with fluorescent dyes to help resolve them from the background. However, in order to get statistically meaningful results especially for high-throughput screening the cell density should be considerably high. In addition many cell types, such as epithelial and endothelial cells preferentially form cell-cell contacts. In such conditions the incidences of cell collisions and cell clustering increase. Many existing solutions for segmenting individual cells from cell clusters in 3D environments have been applied to cell nuclei, which, unlike the cells themselves, retain rather constant shape over time [2–4]. Shortcoming in nuclear labeling is that only a small fraction of cell

is made visible and no conclusion on cell shape or extremities that mediate cell-cell interactions can be made. Moreover, use of nuclear dyes, which are typically excited with high energy low wavelength light, may adversely affect cell viability. Due to these factors, we decided to develop a segmentation method which can efficiently detect individual cells from cell clusters in 3D time-lapse image sequences without using nuclear labeling.

Confocal microscopy techniques have become a standard method for 3D imaging of fluorescently labeled cells, however, they have lower resolution in z-direction compared to resolution within a slice. Spinning disk confocal system allow lower photo toxicity and fluorophore bleaching, and faster image acquisition rates than laser scanning confocal microscopes, thus advancing long term live cell imaging in 3D. However, spinning disk provides lower confocality resulting in some blurring in z-direction due to light emitted by out of focus fluorescently labeled structures. Deconvolution can reduce this type of blurring but does not always completely eliminate it. This blurring as shown in Fig. 1 makes it more challenging to find accurate boundaries between cells.

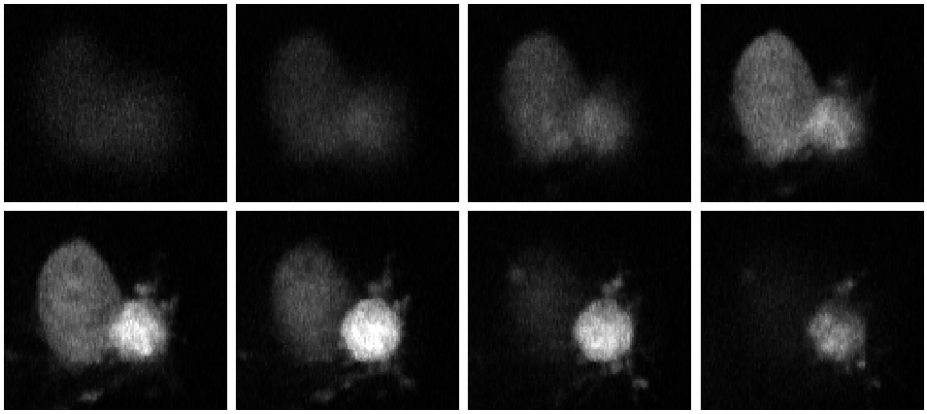

Fig. 1. Optical slices of cells expressing green fluorescent protein (GFP) captured using spinning-disk confocal microscope. From left to right: Slice # 1,5,9,13,17,21,25 and 29 of a cell cluster. Slices towards cluster extremities are very blurred.

2 Related Work

Cell segmentation is a very broad research area due to wide variability in imaging modalities, cell lines, stains, cell densities, acceptable level of manual oversight, etc. We restrict ourselves to only fluorescent microscopy case. The initial cell segmentation approaches used intensity thresholding, which is still popular [5] and common in general purpose software due to its simplicity, acceptable quality in many restricted situations and lack of any other general user-friendly method.

In dense samples, cells often come in contact with each other and form clusters. To separate individual cells from these clusters is one of the major challenges in cell segmentation. Difficulty of this task is due to lack of strong edges between cells (in case their intensity is similar), presence of strong edges within cells (due to non-homogeneous staining) and ability of cells to take wide variety of shapes. One common approach [2, 6] to solve this issue is to assume that cells have a convex shape and when they form a cluster it results in creation of concave regions. Points in these concave regions on cell boundaries are detected and then lines are drawn between these points that minimize some cost function. This approach [2] has been used to segment 3D cells by processing image stack slice by slice and considering split lines in neighboring slices to maintain spatial coherence in z-direction. These methods usually require setting multiple parameters, which is difficult for many challenging sequences.

Another common approach is to detect some key-points or regions, one for each cell, and use them to initialize level sets, watershed, or graph-cuts, for instance to obtain cell segmentation. The performance of these methods depends heavily on detection of seeds, which control where and how many cells will be detected. Some simpler methods use local peaks of intensity and distance transform [7] for seed detection. These methods suffer from over segmentation as they are sensitive to cell texture and small shape distortions. Most popular methods for detecting seeds use multi-scale blob detectors based on LoG (laplacian of gaussian) [8, 9] or Hessian of gaussian [10]. They either use same scale along all axes [8] or in more general case [9] allow blobs to have different sizes along different dimensions in addition to allowing them to rotate.

Another promising method [11] creates a tree of cell candidate regions using MSER, which are scored based on a learnt cell appearance model. Then dynamic programming is used to pick non-overlapping candidates which maximize the total score. This approach unlike other methods does not make any assumption about cell shapes and learns them from training data and as a result has the potential of handling more challenging sequences.

Graph cuts have been used in the past for nuclei segmentation. Both [7, 8] first used graph cuts to find the binary labeling, then they detected seeds and found initial rough segmentation. Finally they refined the segmentation boundary by using the initial segmentation and seeds to create a new multi-label graph, which they solved using α-*expansion* and α-β-*swap* algorithms [12]. Our method has similar structure but differs from them in details of how binary labeling, cell seeds and final segmentation are found. Since we use edge probabilities instead of blob detection [8] and distance transform [7] for cluster splitting, our method can cope with more challenging shapes.

3 Method

This section outlines our cell segmentation method, which is also shown in Fig. 2. Our method consists of four (for 2D images) or five (for 3D image stacks) stages. First, image is pre-processed using median filtering and edge detector is used to

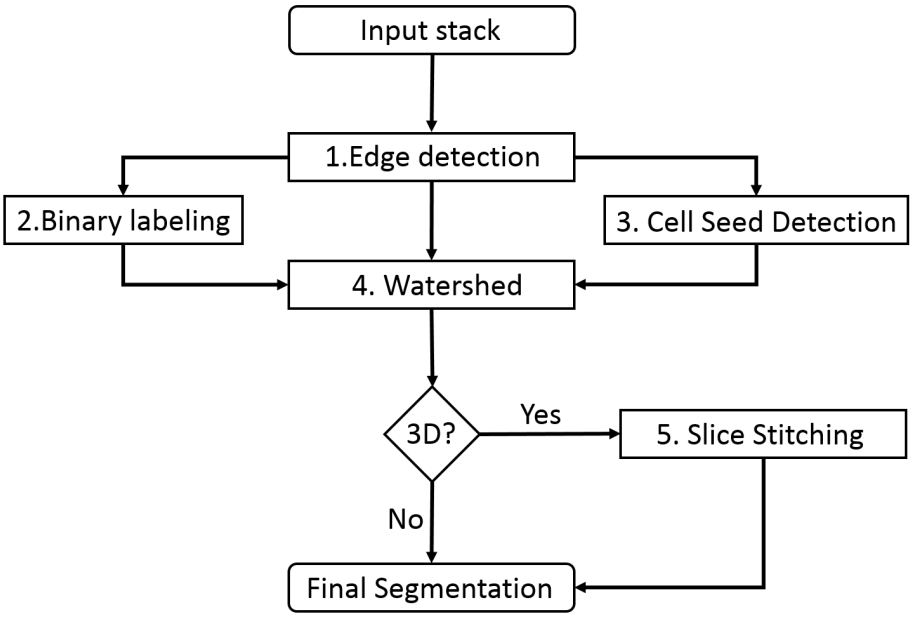

Fig. 2. Flow chart showing the overview of our method.

compute edge probability at each pixel. Next, cells are separated from background using graph cuts, in which the intensity in local neighborhood of a pixel is used to set terminal edge weights. In the third stage, seeds for individual cells are found using another graph cuts stage, in which edge probability map is used to adjust terminal edge weights. This is our main contribution as we propose a new method for locating seeds for individual cells within a cell cluster without making any assumption about cell shape. In the fourth stage, marker-controlled watershed is used to find boundaries of individual cells. Since we are working with an-isotropic 3D data, with low z-direction resolution and some blurring, direct extension of this method to 3D faces some additional challenges. To overcome them, we first segment all 2D slices in a 3D stack and then in the fifth stage 3D cell segmentation is obtained by connecting segmented objects within adjacent segmented slices.

3.1 Edge Probability Map

We have used a state of the art edge detector [13] to generate an edge probability map. This edge detector computes a large set of features from image gradients within a small patch around a pixel at multiple scales and uses structured random trees to output edge mask in a small section of input patch. The contribution of multiple trees and output patches which overlap a given pixels are averaged to obtain edge probability at that pixel. We train it to output edge probabilities at

each pixel, E, as well as edge probability in both horizontal, E_h, and vertical , E_v, direction. The edge detector in its current form works only on 2D images. So for 3D image stacks, we first compute horizontal and vertical edge probabilities by processing each 2D slice separately and then obtain edge probabilities in z-direction, E_z, by slicing the stack parallel to yz-plane and applying edge detector. Figure 3 shows the results of edge detector in all three directions, as can be seen the edge probabilities in z-direction, E_z, are more blurred than E_h or E_v due to factors mentioned in Section 1.

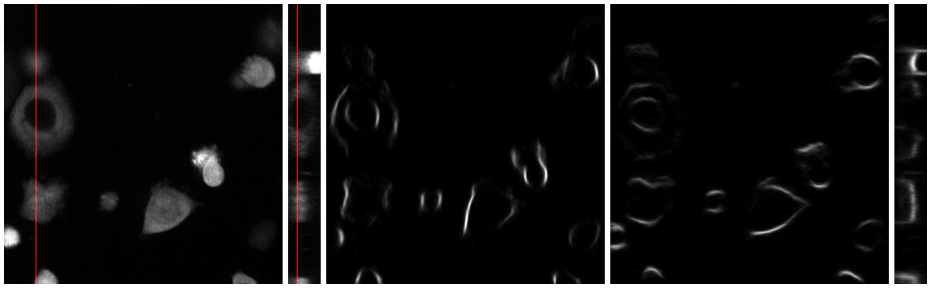

Fig. 3. From left to right: Slice (XY) # 13 and vertical slice (YZ) # 60 from a 3D stack, E_h, E_v and E_z. Red line marks the intersection of XY and YZ slices.

3.2 Graph Structure

We use two separate graphs for binary labeling and cell seed detection. This section details the common elements of these graphs. We represent a 2D image (slice in the case of 3D image stack) by a two terminal grid based graph. Every pixel in the image is represented by a node, referred to as grid node in the text. Each grid node is connected to its neighboring grid nodes, we have used 4-connected neighborhood. Each grid node is also connected to two terminal nodes, source node is labeled cell/foreground and the sink node is labeled background. The details of how the costs of edges between grid nodes and terminal nodes are selected are provided in Sections 3.3 and 3.4.

The cost of edge between neighboring grid nodes is used to enforce spatial smoothness. It is the cost which has to be paid for assigning different labels to two neighboring pixels. If both neighboring pixels are assigned same label then no cost is paid. At the boundary between multiple touching cells and between cells and background, this cost has to be low so that these pixels are not over-penalized. Everywhere else this cost has to be high. We have used the difference of edge probabilities at neighboring grid nodes to set this cost according to:

$$edge(p,q) = C * \exp(\frac{-|E_d(p) - E_d(q)|}{\sigma}) \quad (1)$$

where $edge(p, q)$ is the cost of edge between two neighboring grid nodes, p and q, $E_d \in \{E_h, E_v\}$ is the edge probability value at p and q in the direction, d, of edge between them. C is the maximum cost that a grid edge can take and it is used to adjust the relative importance of terminal edges and neighborhood edges. σ controls how quickly cost decreases as edge probability difference increases. Figure 4 shows the costs of edges between grid nodes which are neighbors in horizontal and vertical direction. Bright values indicate high cost edges which can not be easily cut and dark values indicate low cost edges, which can be easily cut to assign different label to neighboring nodes.

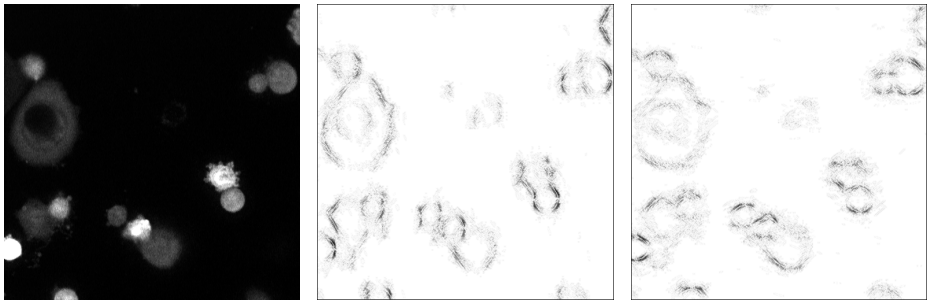

Fig. 4. Edge costs. From left to right: Slice from a 3D stack, $edge_h(p, q)$ and $edge_v(p, q)$.

3.3 Binary Labeling

Binary labeling, which separates cell pixels from background pixels, is needed to ensure that cell seeds do not expand into background regions when watershed is applied to obtain cell boundaries. This stage is relatively easy but it still involves some challenges due to photo bleaching and in-homogeneous absorption of fluorescent dyes. There are cells which are very dark and can only be spotted by a human observer when carefully going through the image. Attempting to detect these cells has a trade-off, as any attempt to detect these dim cells results in inclusion of blurred regions around cells, especially in z-direction, in the final cell segmentation. This effect can be seen in Fig. 3, where blurred region above a bright cell (top left corner in XY slice) has higher intensity than one darker cell (above bottom right corner in XY slice).

Cell density and intensity profiles of individual cells differ significantly both between sequences and within any particular sequence and the distribution of image stack intensities has a very large peak at the beginning (dark background pixels) which gradually dies as intensity increases and a second much smaller peak at the highest intensity (due to overexposure). These issues make it difficult to automatically select a good cell intensity model, which is needed to set the cost of terminal edges of graphs. Otsu thresholding and clustering based solutions usually resulted in much higher threshold and they were only able to

detect bright cells. To ensure that we are able to detect dim cells, we have computed terminal edge costs using a small window (larger than most cells) around each pixel. At each pixel, contrast in the window is computed. If the contrast is higher than a threshold then mean of the window is used as parameter, τ, otherwise region is assumed to be homogeneous (and consisting only of background) and parameter, τ, is set to sum of mean of the window and a parameter γ, which is selected empirically. For 3D image stacks, we use a 3D window to compute the terminal edge costs. Computing terminal cost using intensities within a local patch around the pixel helps in suppressing blurred regions around cells especially in z-direction for 3D image stacks. The terminal edge cost for this step are computed using:

$$edge(BG, p) = \exp(\frac{-I(p)}{\tau(p)}) \tag{2}$$

$$edge(FG, p) = 1 - \exp(\frac{-I(p)}{\tau(p)}) \tag{3}$$

where $edge(FG, p)$ is the cost of edge connecting the grid node p to $source$ $(cell/foreground)$ node, $edge(BG, p)$ is the cost of edge connecting the grid node p to $sink(background)$ node, $I(p)$ is the intensity at grid node p and $\tau(p)$ is the parameter value at p, which incorporates neighborhood information.

The edge cost between neighboring grid nodes is computed using Eq. (1) but a low value of C is chosen so that fine structures at cell boundaries are preserved. The overall function which has to be minimized by min-cut in this stage is:

$$E = \sum_{p \in P} (edge(L_p, p) + \sum_{q \in N(p)} edge(p, q) * (L_p \neq L_q)) \tag{4}$$

where P contains all pixels in the image, L_p and $L_q \in \{FG, BG\}$ are the labels for pixels at location p and q. $N(p)$ is the set of all 4-connected neighbors of node p. The first term, data term, in the above energy function forces bright pixels to be labeled as cell and dark pixels as background, while the second term, boundary term, ensures that there is spatial continuity in the labels of pixels by penalizing neighboring pixels which have different labels. Min-cut of this graph which minimizes Eq. (4), separates pixels into two classes, cell and background (cell matrix). Figure 5 shows the costs of terminal edges and final labeling for one slice from a 3D stack.

Since we are using a two terminal graph, its minimal cut can be computed quickly. However for large image stacks, minimum cut solvers can still take significant memory so we have used Grid Cut [14], a fast multi-core max-flow/min-cut solver with a low memory footprint, to find the min-cut of our graphs.

3.4 Cell Seed Detection

In this section, we describe how we use edge probability map to create a barrier between touching cells within a cell cluster and use it to find cell seeds. We create a second grid graph as described in Section 3.2. The cost of edges between

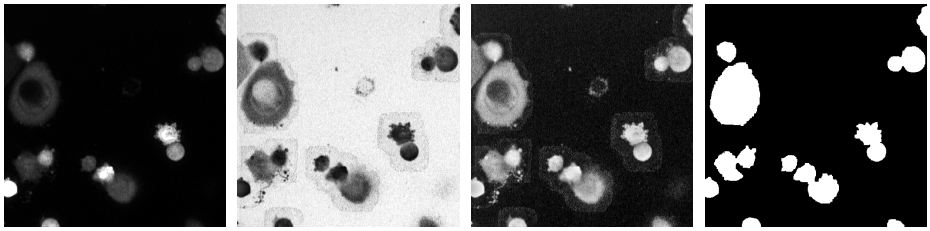

Fig. 5. Binary segmentation. From left to right: Slice from a 3D stack, edge(BG,p), edge(FG,p), binary labeling.

neighboring grid nodes is computed using Eq. (1) but for this graph we choose a higher value of C so that high edge probabilities within cell body due to inhomogeneous staining do not result in detection of multiple seeds. To separate touching cells, we need to create a barrier between them. We create this barrier by using the edge probability map, which has high values at boundaries between cells and between cells and background. We increase the cost of edge between a grid node and background terminal node where edge probability is high. Similarly we decrease the cost of edge between a grid node and foreground terminal node where edge probability is high. This modification in terminal edge costs compensates for extra cost that has to be paid for assigning background labels to boundary pixels between cells. The cost of edges between terminal nodes and grid nodes for this graph are computed using:

$$edge(BG, p) = \exp(\frac{-I(p)}{\tau(p)}) + D * E(p) \tag{5}$$

$$edge(FG, p) = 1 - \exp(\frac{-I(p)}{\tau(p)}) - D * E(p) \tag{6}$$

$E(p)$ is the edge probability at pixel p obtained from edge detector, D is the parameter which controls how much adjustment is needed to terminal edge costs and is selected experimentally. If the edge probability map is very accurate and sharp then a low value suffices but when the edges are blurred and their probability is not very high, then a higher value has to be set to counteract the influence of high intensity at boundary pixels between cells and force the pixels to belong to background. The overall energy function which has to be minimized is:

$$\begin{aligned} E = \sum_{p \in P} ((\exp(\frac{-I(p)}{\tau}) + D * E(p)) * (L_p == FG) \\ + (1 - \exp(\frac{-I(p)}{\tau}) - D * E(p)) * (L_p == BG) + \sum_{q \in N(p)} edge(p, q) * (L_p \neq L_q)) \end{aligned} \tag{7}$$

This function is minimized by the labeling which assigns pixels with low intensity and pixels with high edge probabilities, the background label and pixels with high intensity and low edge probability, the foreground (cell) label. We find the connected components within pixels labeled as cell and use them as cell seeds. Figure 6 depicts the costs of terminal nodes for one slice from a 3D stack in addition to the final detected seeds, which are labeled by seven repeating colors.

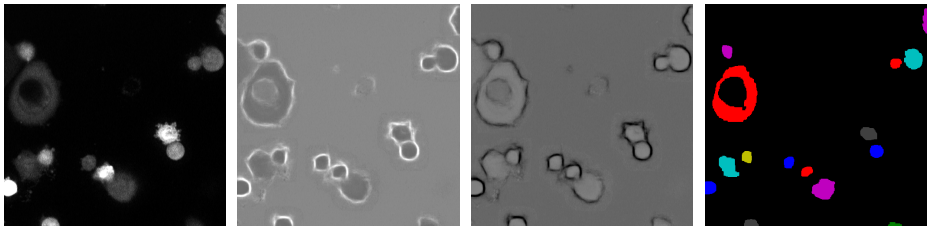

Fig. 6. Cell seed detection. From left to right: Slice from a 3D stack, edge(BG,p), edge(FG,p) and labeled cell seeds.

3.5 Watershed

Once we have obtained cell seeds, we can use multi-label graph cuts or watershed for finding the cell region/boundaries. Multi-label graph cuts have been used in the past [8] for this task but they require a cell intensity and shape model to compute the cost of assigning a pixel to each of the terminal nodes. When cells intensity profiles and shapes vary significantly, then use of simple model like Gaussian model, especially for shape modeling, results in poor performance. This is why we have decided to use watershed for this task. There are two obvious candidates for topographic relief images, edge probability map and intensity image, which can be used by watershed but they both have their weaknesses. There is not always enough difference between the intensity of touching cells to prevent expansion of one cell into another. There are also sometimes holes in the edge probability map, which allow one cell to expand into another. We have therefore decided to combine these two images according to $R = E - I$ and use the resulting image, R, as the topographic relief for watershed. Figure 7 shows these three different topographic reliefs. This results in better performance by combining the strengths of both intensity (good barrier when intensity of touching cells is different) and edge probability map (good barrier where edge probability map is accurate). We also use the binary labeling from Section 3.3 to create a fence (by setting high value to boundary pixels) around foreground pixels to restrict cells from expanding into background region. Then local minima are created at cell seed locations and watershed transform is used to find the cell boundary within a cell cluster.

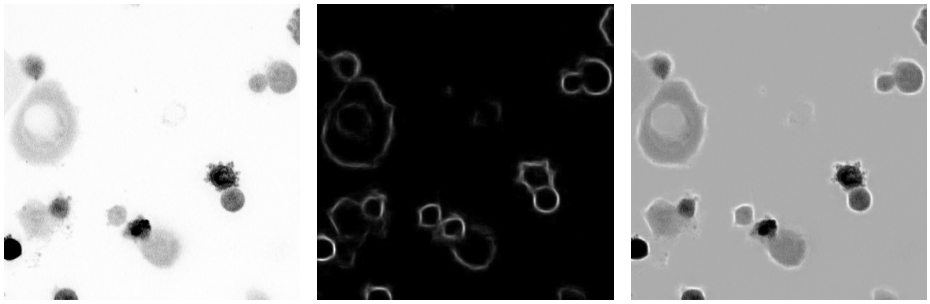

Fig. 7. Watershed relief. From left to right: Negative intensity image, edge probability image and combination of edge and intensity image, R.

3.6 Slice Stitching

The method described so far can be extended to 3D and used for isotropic 3D stacks. However, we are working with an-isotropic 3D stacks, with lower z-direction resolution and blurring as mentioned in Section 1 and shown in Fig. 1 and 3. This often results in either significant distortion in shape of some cells or failure to separate individual cells from a cell cluster. In order to solve these problems we have decided to process 3D stacks slice by slice in 2D to obtain initial 3D segmented stack and then merging the segmented objects in adjacent slices to obtain final 3D segmentation of cells.

When computing edge probability map for each optical slice independently, there arise situations, due to local variations in intensities, in which boundary between cells within a cluster in some slices do not have high probabilities. In addition to that near the extremities of cell clusters boundaries are also much less pronounced due to blurring. These factors result in segmentation errors in few slices. We deal with these errors by using segmentation of maximum intensity image to guide slice merging process. For each detected cell in maximum intensity projection, the best match based on overlap, ratio of area of their intersection and union, is found in slice M of initial 3D segmented stack. This best matched object is usually one of the sharpest slice of that cell. This cell is then expanded in both z-directions by searching for best matching object, based on overlap and chi-squared distance of their intensity histograms, in neighboring slice on each side, slice $M + 1$ and $M - 1$. If a match is found on any side then it is assigned the same label as that of current cell and placed in final segmented stack. The cell template on that side is also updated for further search. If no good match is found then search is performed in next couple of slices. If a match is found then previous cell template is used to fill in the missing slices, otherwise search is terminated in that direction. Once all the objects in maximum intensity projected image are processed, then search is performed for leftover connected objects in initial 3D segmented stack. If any object persists, has good overlap with objects in neighboring slices and they have similar intensity profile, for multiple slices then it is also included in the final segmented 3D stack.

4 Results

We have used two sequences, $S1$ and $S2$, for evaluating our segmentation method. Both contain squamous cell carcinoma cells (HSC3) genetically labeled with GFP. Imaging was done with EC Plan NeoFluar 40x/0.75 air objective and Zeiss Cell Observer spinning disc confocal microscope equipped with CO_2, humidity and temperature controlled top stage incubator. Sequence $S1$ contains 59 image stacks, consisting of 30 slices of 512x512 resolution with voxel size of 0.3x0.3x0.6 μm. Whereas sequence $S2$ contains 156 image stacks, consisting of 47 slices of 512x512 resolution with voxel size of 0.3x0.3x1.5 μm.

We have performed quantitative analysis only for sequence $S1$ because it has sparser cell density and more rigid cell shapes. Since it is very labor intensive to generate detailed (pixel level labeling) good quality ground truth, we have used bounding boxes of cells for evaluating our segmentation method. The ground truth was generated by marking a rectangle on the maximum intensity image from the top view, then two side views (maximum intensity projections) of the selected volume were displayed and the cell's bounds in z-direction were marked in the view which best separated it from other cells and background. The bounding boxes accurately marked the bounds of most cells but there were instances in which better bounding boxes could have been drawn by inspection of all slices at the expense of lot more manual effort. We have excluded cells which were mostly outside the volume being imaged or that were on the boundary but below a certain size from evaluation.

A cell detection is considered a potential true detection (true positive) if overlap of its bounding box, D, with bounding box of any cell in ground truth data, GT, is above a threshold, 0.5. Munkres assignment algorithm [15] with cost $1-overlap(D, GT)$ is used to find bipartite matching between detected cells and ground truth cells. The overlap is computed using:

$$overlap(D, GT) = \frac{|D \cap GT|}{|D \cup GT|} \tag{8}$$

We evaluate the performance of segmentation using precision and recall. Precision measures the proportion of detected cells that are also present in ground truth data, while recall measures the proportion of cells in the ground truth that are correctly detected. To get a single number for comparing different segmentation methods and selecting parameters, we have used F-measure:

$$F = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{9}$$

There are neither many openly available cell segmentation methods nor commonly used datasets, which makes it difficult to compare different methods. We have used segmentation method [8] provided in Farsight toolkit as a baseline because its code is openly available and it uses a method from a popular line of research for separating individual cells from cell clusters (detecting individual cells using blob detection). This method can automatically tune its parameters

but on our sequences automatic parameters resulted in poor performance, it suffered from over-segmentation, so we manually adjusted its parameters. We have quantitatively analyzed the performance of direct extension of our approach to 3D (using 3D grid graph with 6-connected neighborhood and without slice merging stage), referred to as "3D Graph Cuts". We also provide results of two other segmentation methods, MINS (Modular Interactive Nuclear Segmentation) [10] and 3D MLS (3D Multiple Level Sets) [16].

Table 1 presents the quantitative results for sequence $S1$ and Fig. 8 shows the segmentation for two image stacks from the same sequence. Our segmentation method outperforms other methods in terms of both recall and precision with a clear margin. Farsight and MINS resulted in over segmentation in cases where a structure within a cell appeared very dim and in under-segmentation sometimes when cells within a cluster did not have very spherical shape. 3D MLS often failed to separate cells when their intensity difference was not very high and it often produced false positive detections in cell regions with somewhat distinct intensities. 3D graph cuts resulted in slightly worse performance compared to Farsight and MINS. It sometimes produced false detection above/below very bright cells in the blurred regions as can be seen in the third column of Fig. 8, where most of the false positive detections (red bounding boxes) are partly due to this issue. It also sometimes resulted in under-segmentation in situations similar to those shown in Fig. 1. Our method was able to cope with these issues better by ignoring blurred joint objects in slices near extremities of cell clusters. However there were still situations where our method failed to prevent cells from expanding into blurred regions above/below them and resulted in errors (upper left corner of images in second column of Fig. 8). One drawback of our approach was that it produced rough surfaces of cells in z-direction compared to other methods.

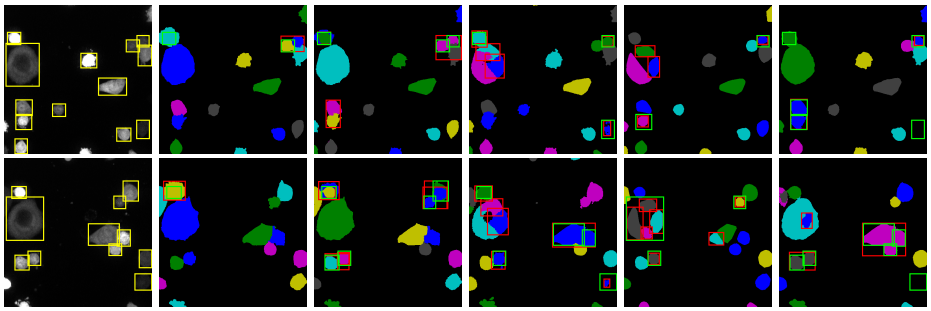

Fig. 8. Maximum intensity projection (view from top) for sequence $S1$. From left to right: Original stack, our method, 3D Graph Cuts, Farsight, MINS and 3D MLC. Bounding boxes showing the ground truth (yellow), false negatives (green) and false positives (red) are marked.

Table 1. Comparison of our method with Farsight, 3D Graph Cuts, MINS and 3D MLS.

	TP	FP	FN	Precision	Recall	F-measure	Time (s)
Our Method	442	37	52	0.92	0.89	0.91	17.63
3D Graph Cuts	361	203	133	0.64	0.73	0.68	31.82
Farsight [8]	373	176	121	0.68	0.76	0.72	97.43
MINS [10]	391	226	103	0.63	0.79	0.70	86.47
3D MLS [16]	335	273	159	0.55	0.68	0.61	27.90

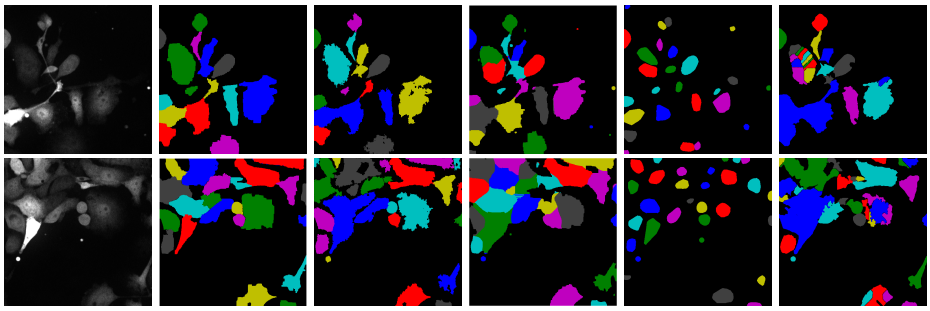

Fig. 9. Maximum intensity projection (view from top) for sequence $S2$. From left to right: Original stack, our method, 3D Graph Cuts, Farsight, MINS and 3D MLC.

We also tested our segmentation method on sequence $S2$, which was more challenging as cells had very flexible shape. For this sequence our segmentation method produced far better results compared to other methods. Figure 9 shows the maximum intensity projection (top view) of couple of 3D image stacks along with results from our method, 3D Graph Cuts, Farsight, MINS and 3D MLS. Farsight and MINS were able to detect many cells correctly but were not able to obtain accurate segmentations. MINS produced very compact segmentations even when cells had very non-compact shapes and Farsight often failed to prevent cell segmentations from encroaching into other cells and background despite the presence of strong boundary in many instances. 3D MLS failed to detect many actual cells (under-segmentation) and produced many false detections (over-segmentation), so its performance was worst among all the methods. 3D graph cuts also suffered heavily from under-segmentation errors. Our method was able to detect cells with decent accuracy most of the time and resulted in good segmentation of most cells. However, in situations where cells had formed long extensions, our method sometimes resulted in over segmentation or segmenting those extensions as part of other nearby cells.

We also compared the running times of our method with other methods for sequence $S1$ and Table 1 lists the average time required to process one 3D image stack on Intel Core i7-3632QM with 8GB RAM. Both our method and 3D graph cuts were written in Matlab and used C++ packages for finding min-cut

of graphs. MINS was also written in Matlab while both Farsight and 3D MLS were written in C++. MINS and Farsight had similar range of running times, which was more than 3 times slower than 3D MLS. Our method was faster than all these methods.

The performance of our method depends heavily on edge detector's performance. We could not directly evaluate the performance of edge detector due to difficulty of obtaining ground truth data but we replaced this edge detector with conventional gradient detectors (sobel, prewitt) and they resulted in significant decrease, 13% decrease in F-measure, in performance of segmentation. Most of the additional errors were under-segmentation errors, which occurred when touching cells had a small intensity difference.

Our method was able to handle challenging (in terms of shape variability) sequences better than other methods that we tested. However there were some situations in which it produced errors. It usually resulted in under-segmentation in situations where cells had a very weak boundary between them and edge detector could not produce strong response at those boundaries. It also occasionally resulted in over-segmentation of cells, which were in-homogeneously stained and contained multiple regions of significantly different intensities and as a result caused the edge detector to produce high edge probabilities within cells.

5 Discussion and Conclusion

We have proposed a novel cell detection and segmentation method, which is able to robustly separate touching cells in confocal microscopy image sequences. Experimental evaluation has shown that our method is able to cope with sequences with very flexible cell shapes. We will in future explore the possibility of incorporating prior knowledge of cell shape within this processing pipeline which can boost performance for sequences with rigid cell shapes. Another possible extension of our work, which can result in some performance boost is to adapt the edge detector to microscopy data by extracting better features for edge detection and using training data consisting only of fluorescent microscopy images.

References

1. Kimlin, L., Kassis, J., Virador, V.: 3d in vitro tissue models and their potential for drug screening. Expert Opinion on Drug Discovery **8** (2013) 1455–1466 PMID: 24144315.
2. Indhumathi, C., Cai, Y., Guan, Y., Opas, M.: An automatic segmentation algorithm for 3d cell cluster splitting using volumetric confocal images. Journal of Microscopy **243** (2011) 60–76
3. Lin, G., Chawla, M.K., Olson, K., Guzowski, J.F., Barnes, C.A., Roysam, B.: Hierarchical, model-based merging of multiple fragments for improved three-dimensional segmentation of nuclei. Cytometry Part A **63A** (2005) 20–33
4. Ortiz de Solórzano, C., García Rodriguez, E., Jones, A., Pinkel, D., Gray, J.W., Sudar, D., Lockett, S.J.: Segmentation of confocal microscope images of cell nuclei in thick tissue sections. Journal of Microscopy **193** (1999) 212–226

5. Meijering, E.: Cell segmentation: 50 years down the road [life sciences]. Signal Processing Magazine, IEEE **29** (2012) 140–145
6. Farhan, M., Yli-Harja, O., Niemist, A.: A novel method for splitting clumps of convex objects incorporating image intensity and using rectangular window-based concavity point-pair search. Pattern Recognition **46** (2013) 741 – 751
7. Dank, O., Matula, P., Ortiz-de Solrzano, C., Muoz-Barrutia, A., Maka, M., Kozubek, M.: Segmentation of touching cell nuclei using a two-stage graph cut model. In Salberg, A.B., Hardeberg, J., Jenssen, R., eds.: Image Analysis. Volume 5575 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2009) 410–419
8. Al-Kofahi, Y., Lassoued, W., Lee, W., Roysam, B.: Improved automatic detection and segmentation of cell nuclei in histopathology images. Biomedical Engineering, IEEE Transactions on **57** (2010) 841–852
9. Kong, H., Akakin, H., Sarma, S.: A generalized laplacian of gaussian filter for blob detection and its applications. Cybernetics, IEEE Transactions on **43** (2013) 1719–1733
10. Lou, X., Kang, M., Xenopoulos, P., Muoz-Descalzo, S., Hadjantonakis, A.K.: A rapid and efficient 2d/3d nuclear segmentation method for analysis of early mouse embryo and stem cell image data. Stem cell reports **2** (2014) 382397
11. Arteta, C., Lempitsky, V., Noble, J., Zisserman, A.: Learning to detect cells using non-overlapping extremal regions. In Ayache, N., Delingette, H., Golland, P., Mori, K., eds.: Medical Image Computing and Computer-Assisted Intervention MICCAI 2012. Volume 7510 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 348–356
12. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23** (2001) 1222–1239
13. Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. In: ICCV. (2013)
14. Jamriska, O., Sykora, D., Hornung, A.: Cache-efficient graph cuts on structured grids. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. (2012) 3673–3680
15. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics **5** (1957) 32–38
16. Chinta, R., Wasser, M.: Three-dimensional segmentation of nuclei and mitotic chromosomes for the study of cell divisions in live drosophila embryos. Cytometry Part A **81A** (2012) 52–64