Fast Background Removal in 3D Fluorescence Microscopy Images Using One-Class Learning*

Lin Yang[1,**], Yizhe Zhang[1], Ian H. Guldner[2], Siyuan Zhang[2], and Danny Z. Chen[1]

[1] Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
[2] Department of Biological Sciences, Harper Cancer Research Institute, University of Notre Dame, Notre Dame, IN 46556, USA
lyang5@nd.edu

Abstract. With the recent advances of optical tissue clearing technology, current imaging modalities are able to image large tissue samples in 3D with single-cell resolution. However, the severe background noise remains a significant obstacle to the extraction of quantitative information from these high-resolution 3D images. Additionally, due to the potentially large sizes of 3D image data (over 10^{11} voxels), the processing speed is becoming a major bottleneck that limits the applicability of many known background correction methods. In this paper, we present a fast background removal algorithm for large volume 3D fluorescence microscopy images. By incorporating unsupervised one-class learning into the percentile filtering approach, our algorithm is able to precisely and efficiently remove background noise even when the sizes and appearances of foreground objects vary greatly. Extensive experiments on real 3D datasets show our method has superior performance and efficiency comparing with the current state-of-the-art background correction method and the rolling ball algorithm in ImageJ.

1 Introduction

With the recent advances of optical tissue clearing technology, current imaging modalities are able to image large tissue samples (e.g., the whole mouse brain) in 3D with single-cell resolution [10]. This creates new opportunities for biomedical research, ranging from studying how the brain works to developing new medicines for cancer. But, it also brings new challenges to extract information from such large volume 3D images. Due to large imaging depth and tissue auto-fluorescence, background noise in such 3D images is often strong and highly inhomogeneous [1], not only preventing effective 3D visualization (Fig. 1(a) and 1(d)) but also causing difficulties to many image processing tasks, such as segmentation (Fig. 1(c) and 1(f)), registration, and tracking. In this paper, we present a fast background removal algorithm that is capable of precisely identifying and removing background noise in such large volume 3D images.

* Source code is available at `https://github.com/linyang519522` or upon request.
** Corresponding author.

N. Navab et al. (Eds.): MICCAI 2015, Part III, LNCS 9351, pp. 292–299, 2015.
DOI: 10.1007/978-3-319-24574-4_35

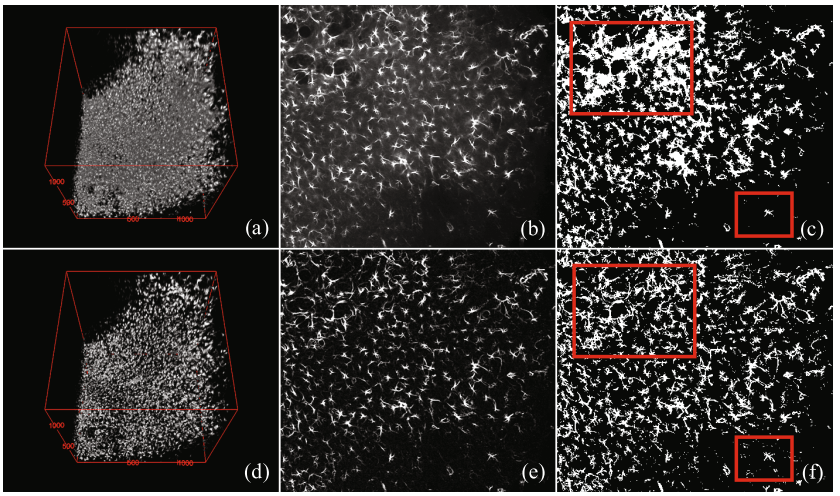

Fig. 1. (a) and (d): 3D visualization results before and after background removal by our algorithm; (b) and (e): selected 2D slices inside the sample 3D image before and after background removal; (c) and (f): segmentation results of (b) and (e) based on thresholding (more accurate segmentation is achieved in both high intensity (upper-left window) and low intensity (bottom-right window) areas after background removal).

Many methods are known for removing undesired background noise in microscopy images, such as spatial filtering [4], rolling ball algorithms [2], fitting smoothly varying function [6], entropy minimization [5], and matrix rank minimization [8]. However, due to the rapidly increasing sizes of image data, the time and space complexities of some sophisticated methods (e.g., optimization based methods and function fitting methods) are too high. With whole-body imaging at single-cell resolution [9], image sizes will continue to grow. Large 3D imaging fields yield not only large data sizes but also wide variations in the sizes of foreground objects. Although simple methods like spatial filtering and rolling ball algorithms are quite fast, significant variance of foreground objects can greatly affect their performance. For example, in some parts of our brain tumor images, cells are uniformly distributed, for which small window/ball sizes give the best performance; in other parts, tumor cells are clustered closely together and form much larger target objects, for which small window/ball sizes may include tumor cell clusters as part of the background (to be removed). We call those "small" windows that cause removal of foreground objects the *undersized windows*.

Taking advantage of the efficiency of spatial filtering methods, we develop a new spatial filtering algorithm for background removal, which is capable to overcome the "undersized window" issue. Two possible approaches may be considered for the window size issue: (1) intelligently choosing the window sizes at different parts of the images (e.g., a small window size for single cells, a big window size for large clusters); (2) first using a fixed small window size, then identifying "undersized windows", and recomputing their background accordingly. The first

approach requires certain effective foreground object detection, which may not be efficient and accurate. In [6], a method in the second approach was given, which identified "undersized windows" by automatic thresholding with a kernel density estimator. However, it still fails when foreground objects are too big [6], possibly caused by a significant amount of "undersized windows". By utilizing the recently developed unsupervised one-class learning model [7], we tackle this critical window size issue even when foreground objects are ~ 100 times larger than the pre-selected window size. Extensive experiments on real 3D datasets show our method has better accuracy and is ~ 160 times faster than the current state-of-the-art background correction method [8]. Also, it has much better accuracy yet is only ~ 1 time slower than the prevailing rolling ball algorithm in ImageJ [2]. Tables 1 and 2 show detailed comparison results.

2 Method

Our method includes three main steps: (1) estimate background noise using a fixed small window size; (2) detect "undersized windows" occurred in Step (1); (3) recompute background noise in the detected "undersized windows".

2.1 Estimating Background Noise

Common background/shading correction methods use additive models or multiplicative models to handle noise [6,8]. In our images, because multiplicative noise is significantly smaller than additive noise, we utilize an additive model $I(x, y, z) = F(x, y, z) + B(x, y, z)$, where $I(x, y, z)$ denotes the observed signal, $F(x, y, z)$ denotes the foreground signal, and $B(x, y, z)$ denotes the background noise at a voxel (x, y, z).

Our method first uses a fixed small window size w to estimate background noise in the image. Although this window size may result in a large number of "undersized windows", we will deal with them using the algorithm in Section 2.2. Because the background noise varies slowly, meaning it has low frequency in the Fourier domain, we only need to estimate the background noise for a subset of all voxels (which we call *sampled voxels*). By the Nyquist-Shannon sampling theorem, as long as our sampling frequency is more than twice as high as the frequency of the background noise, we can recover the true background noise at every voxel. Thus, we use a point grid to choose the sampled voxels. Then, centered at each sampled voxel (x_s, y_s, z_s), a $w \times w \times w$ window $W(x_s, y_s, z_s)$ is defined and p percentile of the intensities of all voxels inside $W(x_s, y_s, z_s)$ is used to estimate the background noise $B(x_s, y_s, z_s)$. In order to maximize the efficiency and quality, the interval of sampled points is chosen to be equal to the window size so that there is neither gap nor overlap between neighboring windows (Fig. 2(a)). Because the intensity of each voxel is in a small set of integers (e.g., $[0, \ldots, 255]$), this step can be computed efficiently in linear time. Since a smaller w will yield a larger sampled set and better estimation of the background (when it is not an "undersized window"), we tend to use a very small w, say 3 or 5.

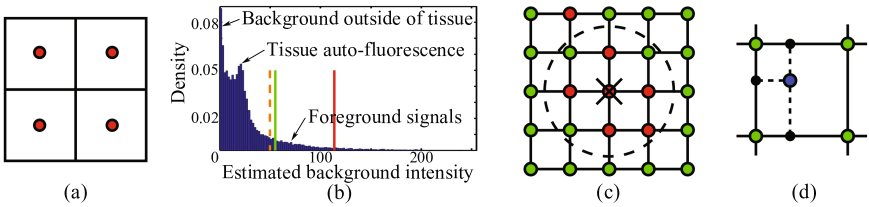

Fig. 2. (a) Each red point is a sampled voxel and the cubic window around it is used to estimate its background intensity; (b) an example histogram of estimated background noise (the green line is the threshold selected by our method, the red line is the threshold selected by median + 3 × std, and the orange dashed line is the manually selected threshold); (c) red points denote the detected "undersized windows", and green points denote the correctly estimated windows (we re-estimate the background noise of the point marked by X); (d) the blue point is a voxel amid sampled voxels, whose value is computed by linear interpolation based on its neighboring sampled voxels.

2.2 Detecting Undersized Windows

In this section, we formulate "undersized window" detection as a one-class learning problem [7], and show how to utilize the properties of our specific problem to make our one-class learning solution more accurate and efficient than in [7].

A key task to our method is to precisely detect "undersized windows". Specifically, given sampled windows $W(x_s, y_s, z_s)$, $s \in \{1, \ldots, n\}$, find all "undersized windows" $W(x_u, y_u, z_u)$ among them. A straightforward way is to use some simple histogram thresholding schemes to determine a threshold on $B(x_s, y_s, z_s)$ (e.g., using the median and standard deviation, finding the local maxima in the second derivative and the kernel density estimator [6]). However, the distribution of $B(x_s, y_s, z_s)$ could be quite complex and greatly affect the performance of such methods (Fig. 2(b)). Thus, we need to develop a more robust approach to detect these "undersized windows". Because background noise (although quite inhomogeneous to affect image processing tasks) is more homogeneous than foreground signals, correctly estimated $B(x_s, y_s, z_s)$'s are much closer to one another while $B(x_u, y_u, z_u)$'s (which are actually foreground signals) have larger variance. This property allows us to treat $B(x_u, y_u, z_u)$'s as outliers among $B(x_s, y_s, z_s)$'s. Thus, we reformulate our problem as follows: Given an unlabeled dataset $\mathcal{X} = \{x'_i = B(x_i, y_i, z_i)\}_{i=1}^n$, find a classification function $f : \mathbb{R} \to \mathbb{R}$ that is able to determine the outliers in $\mathcal{X}$. Our solution for this problem is based on the state-of-the-art unsupervised one-class learning model [7].

We first briefly review the method in [7]. Its basic idea is to use a self-guided labeling procedure to identify suspicious outliers and then train a large margin one-class classifier to separate such outliers from reliable positive samples. This procedure is achieved by minimizing the following objective function: $\min_{f \in \mathcal{H}, \{y_i\}} \sum_{i=1}^n (f(\boldsymbol{x}'_i) - y_i)^2 + \gamma_1 ||f||^2_{\mathcal{M}} - \frac{2\gamma_2}{n^+} \sum_{i, y_i > 0} f(\boldsymbol{x}'_i)$ such that $y_i \in \{c^+, c^-\}, \forall i \in [1, \ldots, n]$ and $0 < n^+ = |\{i \mid y_i > 0\}| < n$. Here, $f(\boldsymbol{x}') = \sum_{i=1}^n \kappa(\boldsymbol{x}', \boldsymbol{x}'_i)\alpha_i$ is the target classification function, where α_i is the expansion coefficient of the functional base $\kappa(\cdot, \boldsymbol{x}'_i)$. In our case, we use an RBF kernel with

bandwidth $\sigma^2 = \sum_{i,j} ||\boldsymbol{x}'_i - \boldsymbol{x}'_j||^2/n^2$. y_i is the soft label assignment for each input data $\boldsymbol{x}'_i$, by choosing $(c^+, c^-) = (\sqrt{\frac{n-n^+}{n^+}}, -\sqrt{\frac{n^+}{n-n^+}})$, the model treats the positive samples and outliers in a more balanced way. The first term regularizes the target classification function to make it consistent with the label assignment. The second term $||f||^2_{\mathcal{M}}$ is the manifold regularizer that regularizes the smoothness of the intrinsic manifold structure $\mathcal{M}$. It is constructed by a neighborhood graph with affinity matrix $\mathbf{W}$. $\mathbf{W}$ is defined as follows:

$$W_{ij} = \begin{cases} \exp(-\frac{||\boldsymbol{x}'_i - \boldsymbol{x}'_j||^2}{\varepsilon^2}), & i \in \mathcal{N}_j \text{ or } j \in \mathcal{N}_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

ε is the bandwidth parameter, and $\mathcal{N}_i$ is the set of indices of $\boldsymbol{x}'_i$'s k nearest neighbors in $\mathcal{X}$. We have $||f||^2_{\mathcal{M}} = \frac{1}{2}\sum_{i,j=1}^{n}(f(\boldsymbol{x}'_i) - f(\boldsymbol{x}'_j))^2 W_{ij}$. The last term maximizes the average margin of judged positive samples. Finally, $\gamma_1, \gamma_2 > 0$ are trade-off parameters that control the relative importance between these parts.

The above minimization problem involves a continuous function f and discrete variables $\{y_i\}$, which is very difficult to optimize. In [7], an alternative optimization algorithm was given, but it cannot guarantee finding a global minimum. Interestingly, for our specific problem, we are able to find a global minimum in a highly effective way. Note that in our setting, the data lie in a small 1D discrete space ($x'_i \in [0, \ldots, 255]$), and we seek a cut between these data points. Since there are at most 257 ways to cut our data (i.e., $C \in \{-0.5, 0.5, 1.5, \ldots, 255.5\}$), we can simply compute the optimal objective function value for each one of them and find the best cut $C_{optimal}$ that minimizes the objective function. After some transformations (see more details in [7]), the original minimization problem for any given cut point C can be put into the following matrix form:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \ \boldsymbol{\alpha}^\top \mathbf{K}(\mathbf{I} + \gamma_1(\mathbf{D} - \mathbf{W}))\mathbf{K}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top \mathbf{K}\tilde{\boldsymbol{y}} \\ &s.t. \ ||\boldsymbol{\alpha}|| = 1, 0 < n^+ = |\{i \mid x'_i \le C\}| < n \\ &\quad y_i = c^+ + \gamma_2/n^+, \forall i \in \{i \mid x'_i \le C\}, \ y_j = c^-, \forall j \in \{j \mid x'_j > C\} \end{aligned} \tag{2}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^\top \in \mathbb{R}^n$, $\tilde{\boldsymbol{y}} = [y_1, \ldots, y_n]^\top$, and the kernel matrix $\mathbf{K} = [\kappa(\boldsymbol{x}'_i, \boldsymbol{x}'_j)]_{1 \le i,j \le n} \in \mathbb{R}^{n \times n}$. $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} W_{ij}$. To take care of the constraint $0 < n^+ < n$, we add to $\mathcal{X}$ two artificial data points with values -1 and 256. Problem (2) was well studied [7], and its minimum is achieved when $\boldsymbol{\alpha} = (\mathbf{K}(\mathbf{I} + \gamma_1(\mathbf{D} - \mathbf{W}))\mathbf{K} - \lambda^* \mathbf{I})^{-1}\mathbf{K}\tilde{\boldsymbol{y}}$, where λ^* is the smallest real-valued eigenvalue of the matrix $\begin{bmatrix} \mathbf{K}(\mathbf{I} + \gamma_1(\mathbf{D} - \mathbf{W}))\mathbf{K} & -\mathbf{I} \\ -(\mathbf{K}\tilde{\boldsymbol{y}})(\mathbf{K}\tilde{\boldsymbol{y}})^\top & \mathbf{K}(\mathbf{I} + \gamma_1(\mathbf{D} - \mathbf{W}))\mathbf{K} \end{bmatrix}$.

For each cut point C, its optimal objective function value is computed using that $\boldsymbol{\alpha}$. We can then easily find $C_{optimal}$, which attains the minimum objective function value among all C's. After find $C_{optimal}$, again since $x'_i \in [0, \ldots, 255]$, we need not actually compute $f(x')$ to determine whether x' is an outlier. We only need to compare its value against $C_{optimal}$. Thus, $W(x_s, y_s, z_s)$ is classified as "undersized window" if $B(x_s, y_s, z_s) > C_{optimal}$.

Finally, we utilize a sampling strategy to make this procedure more efficient. In each 3D image, m windows are randomly sampled from all $W(x_s, y_s, z_s)$'s to calculate $\hat{C}_{optimal}$. For each class of images which are stained by the same fluorescence dye, bootstrap is used to choose a sufficiently large m [3]. More specifically, the standard error of $\hat{C}_{optimal}$, which can be computed by bootstrap, is used to decide whether m is sufficient. Initially, we set $m = 1000$, and gradually increase m until the standard error of $\hat{C}_{optimal}$ is small enough.

2.3 Recomputing Background Noise in "Undersized Windows"

After detecting all "undersized windows", we re-estimate their values based on the mean of their surrounding correctly estimated windows. Specifically, the surrounding is defined by whether the distance between the centers of the "undersized window" and the correctly estimated window is smaller than r. Initially, $r = 1$; then we gradually increase r until there are surrounding correctly estimated windows (Fig. 2(c)); finally, the background noise for each voxel is computed by linear interpolation based on the neighboring sampled voxels (Fig. 2(d)), and the estimated foreground signal is computed by $F(x, y, z) = I(x, y, z) - B(x, y, z)$.

3 Experiments and Results

To perform quantitative performance analysis, we collected four sample 3D images using two-photon microscopy on four different 3D mouse brains. Each mouse brain was made optically transparent by the method in [10]. To test the applicability of our method to different types of fluorescence dyes, Samples 1 and 3 were stained by DAPI which marks all cell nuclei in the samples. In these two samples, the background noise is moderate, but the sizes of foreground objects vary greatly (Fig. 3). Samples 2 and 4 were stained by GFAP which marks all astrocytes in the samples. In these two samples, the foreground objects are relatively sparse, but the background noise is quite severe (Fig. 3). Three representative slices were selected from each of these four 3D images for evaluation. To represent the changes across different depths, they were selected from the top, middle, and bottom of the 3D images. Human experts then labeled all the foreground objects in these representative slices. After that, the ground truth images were generated by computing the dot product between the binary human-labeled images and the original images. In this way, in the ground truth images, all the foreground objects keep their original intensities while all background intensities become 0.

Two known methods are selected for comparison with our method. The first one is a recently published shading correction method based on matrix ranking minimization [8]. The second one is the rolling ball algorithm, which is a well established method in ImageJ [2]. In the first method, for each 2D slice, its background is estimated by the neighboring d slices, where d is manually selected so that it achieves the best performance. In the second method, the ball size is selected to be equal to our window size $w = 5$. In our method, the percentile $p = 10\%$ and the parameters in the one-class learning model, $k = 7$, $\gamma_1 = 1$, $\gamma_2 = 1$, $\varepsilon = 1$, and $m = 4000$, are fixed across different samples.

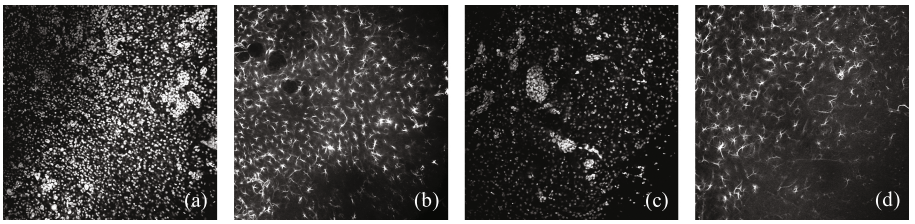

Fig. 3. (a)(b)(c)(d): Some example slices from 3D image Samples 1, 2, 3, and 4.

Table 1. Root mean squared errors (RMSE).

	Sample 1	Sample 2	Sample 3	Sample 4	Avg. Improvement
Our method	**9.21**	**8.48**	**4.26**	**7.74**	**52.53%**
Low rank [8]	18.01	9.50	5.68	7.81	34.56%
Rolling ball [2]	20.07	11.03	7.24	9.88	22.37%
Original Image	17.53	17.15	9.23	18.55	

As in [6,8], the quality of each method is measured using the root mean squared error (RMSE) between the processed slice F and the ground truth G, with RMSE $= \sqrt{\frac{1}{n}\sum_{i,j}(F_{ij} - G_{ij})^2}$. Table 1 summarizes the results. A smaller RMSE means more accurate background correction. On Table 1, one can see that when foreground objects are sparse (Samples 2 and 4), our method and [8] have comparable performance and are both much better than the rolling ball algorithm [2]. However, when foreground objects are more complicated (Samples 1 and 3), both [8] and the rolling ball algorithm are considerably worse than our method. In some cases, they can damage foreground objects (Fig. 4) and thus are even worse than the original images.

Table 2. Average processing time per voxel (in μs).

	Sample 1	Sample 2	Sample 3	Sample 4	Average
Our method	$0.66\mu s$	$0.76\mu s$	$0.98\mu s$	$0.81\mu s$	$0.80\mu s$
Low rank [8]	$130.3\mu s$	$128.2\mu s$	$133.7\mu s$	$126.3\mu s$	$129.6\mu s$
Rolling ball [2]	$0.54\mu s$	$0.46\mu s$	$0.42\mu s$	$0.46\mu s$	$0.47\mu s$

The efficiency of each method is measured by the average processing time per voxel, as shown in Table 2. Both [8] and our method were implemented in MATLAB. All these methods were tested on the same workstation. By Table 2, to process a 3D image for a whole mouse brain (with 10^{11} voxels), our method will take about 22 hours while the method in [8] will take 150 days. This shows the crucial need of efficiency for processing such large 3D images.

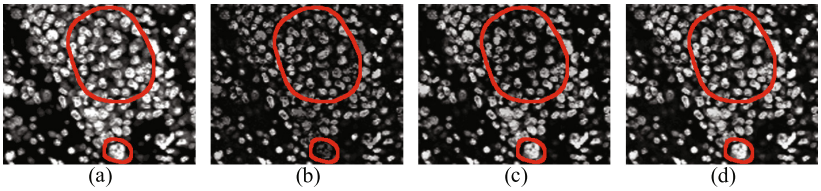

Fig. 4. A cropped window in a slice from Sample 1. (a) The labeled ground truth; (b) the result of the rolling ball algorithm; (c) the result of [8]; (d) our result. The rolling ball algorithm damages all foreground objects that are larger than its ball size. [8] is able to deal with sparse objects; but, it still damages the center of big clusters.

4 Conclusions

We present a fast background removal algorithm based on percentile filtering. The crucial "undersized window" problem in spatial filtering is tackled by unsupervised one-class learning. Extensive experiments on real 3D datasets show our method has superior performance and efficiency comparing with the current state-of-the-art background correction method and the rolling ball algorithm.

Acknowledgment. This research was supported in part by NSF under Grant CCF-1217906, and by NIH Pathway to Independence Award 5R00CA158066-04 and NIH 1R01CA194697-01.

References

1. Chen, T.W., Lin, B.J., Brunner, E., Schild, D.: In situ background estimation in quantitative fluorescence imaging. Biophysical Journal 90(7), 2534–2547 (2006)
2. Collins, T.J.: ImageJ for microscopy. Biotechniques 43(1), 25–30 (2007)
3. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. CRC Press (1994)
4. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 2nd edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2001)
5. Likar, B., Maintz, J.A., Viergever, M.A., Pernus, F.: Retrospective shading correction based on entropy minimization. Journal of Microscopy 197(3), 285–295 (2000)
6. Lindblad, J., Bengtsson, E.: A comparison of methods for estimation of intensity non uniformities in 2D and 3D microscope images of fluorescence stained cells. In: Proc. of the Scandinavian Conf. on Image Analysis, pp. 264–271 (2001)
7. Liu, W., Hua, G., Smith, J.R.: Unsupervised one-class learning for automatic outlier removal. In: 2014 IEEE Conference on CVPR, pp. 3826–3833 (2014)
8. Peng, T., Wang, L., Bayer, C., Conjeti, S., Baust, M., Navab, N.: Shading correction for whole slide image using low rank and sparse decomposition. In: Golland, P., Hata, N., Barillot, C., Hornegger, J., Howe, R. (eds.) MICCAI 2014, Part I. LNCS, vol. 8673, pp. 33–40. Springer, Heidelberg (2014)
9. Tainaka, K., Kubota, S.I., Suyama, T.Q., Susaki, E.A., Perrin, D., Ukai-Tadenuma, M., Ukai, H., Ueda, H.R.: Whole-body imaging with single-cell resolution by tissue decolorization. Cell 159(4), 911–924 (2014)
10. Tomer, R., Ye, L., Hsueh, B., Deisseroth, K.: Advanced CLARITY for rapid and high-resolution imaging of intact tissues. Nature Protocols 9(7), 1682–1697 (2014)