Bioinformatics, 33(13), 2017, 2020–2028
doi: 10.1093/bioinformatics/btx107
Advance Access Publication Date: 22 February 2017
Original Paper

Bioimage informatics

fastER: a user-friendly tool for ultrafast and robust cell segmentation in large-scale microscopy

Oliver Hilsenbeck[1,*], Michael Schwarzfischer[2], Dirk Loeffler[1], Sotiris Dimopoulos[1], Simon Hastreiter[1], Carsten Marr[2], Fabian J. Theis[2,3] and Timm Schroeder[1,*]

[1]Department of Biosystems Science and Engineering, ETH Zurich, Basel, Switzerland, [2]Institute of Computational Biology, Helmholtz Zentrum München, Neuherberg, Germany and [3]Department of Mathematics, Technische Universität München, Garching, Germany

*To whom correspondence should be addressed.

Associate Editor: Robert Murphy

Received on June 13, 2016; revised on January 15, 2017; editorial decision on February 18, 2017; accepted on February 21, 2017

Abstract

Motivation: Quantitative large-scale cell microscopy is widely used in biological and medical research. Such experiments produce huge amounts of image data and thus require automated analysis. However, automated detection of cell outlines (cell segmentation) is typically challenging due to, e.g. high cell densities, cell-to-cell variability and low signal-to-noise ratios.

Results: Here, we evaluate accuracy and speed of various state-of-the-art approaches for cell segmentation in light microscopy images using challenging real and synthetic image data. The results vary between datasets and show that the tested tools are either not robust enough or computationally expensive, thus limiting their application to large-scale experiments. We therefore developed fastER, a trainable tool that is orders of magnitude faster while producing state-of-the-art segmentation quality. It supports various cell types and image acquisition modalities, but is easy-to-use even for non-experts: it has no parameters and can be adapted to specific image sets by interactively labelling cells for training. As a proof of concept, we segment and count cells in over 200 000 brightfield images (1388 $\times$ 1040 pixels each) from a six day time-lapse microscopy experiment; identification of over 46 000 000 single cells requires only about two and a half hours on a desktop computer.

Availability and Implementation: C ++ code, binaries and data at https://www.bsse.ethz.ch/csd/software/faster.html.

Contact: oliver.hilsenbeck@bsse.ethz.ch or timm.schroeder@bsse.ethz.ch

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Quantitative large-scale cell microscopy has become an indispensable tool for biomedical research: imaging based high-content screening is widely used, e.g. for drug discovery (Zanella *et al.*, 2010), and time-lapse imaging (Schroeder, 2008) provides insights into numerous biological processes (Cohen *et al.*, 2010; Eilken *et al.*, 2009; Etzrodt *et al.*, 2014; Filipczyk *et al.*, 2015; Rieger *et al.*, 2009). Accurate detection of cell outlines in the acquired image data (cell segmentation) is crucial for automated cell tracking (Magnusson *et al.*, 2015) and to quantify cellular features like morphologies or fluorescent signals from intracellular markers (Li *et al.*, 2013). The enormous amounts of data typically generated in such

experiments (up to millions of images per experiment) require efficient automated analysis (Myers, 2012). Automated cell segmentation, however, is often challenging, because problems like high cell densities, cell-to-cell variability, complex cellular shapes, varying image illumination and low signal-to-noise ratios (SNRs) cannot always be avoided (Peng, 2008). This holds true especially when imaging mammalian cells (Skylaki *et al.*, 2016), which typically change their individual morphologies with high frequency (Schroeder, 2011).

Many methods for automatic cell segmentation have been published (Eliceiri *et al.*, 2012), but most are limited to specific cell and microscopy types (Meijering, 2012) or not robust enough for large-scale microscopy (Buggenthin *et al.*, 2013). Here, we evaluate accuracy and speed of several state-of-the-art approaches for cell segmentation. The results show that the segmentation quality by each method varies between datasets, that no method clearly outperforms the others and that lacking efficiency remains the major challenge for their application to large scale experiments. We therefore developed the software fastER (**fast** segmentation with **e**xtremal **r**egions), a new fast and trainable tool for cell segmentation that extracts texture and shape features from candidate regions, estimates the likelihood of each to be a cell with a support vector machine (SVM) and calculates an optimal set of non-overlapping candidate regions using a divide and conquer approach.

fastER uses *extremal regions* (Matas *et al.*, 2002), which are represented as a tree of nested regions (see Methods), as candidates. Approaches based on merge trees have been used successfully before for bioimage analysis (Funke *et al.*, 2015; Liu *et al.*, 2012; Schiegg *et al.*, 2014). Most notably, *CellDetect* (Arteta *et al.*, 2012) uses extremal region trees, a structured SVM (Tsochantaridis *et al.*, 2004) and dynamic programming to extract an optimal set of non-overlapping regions. This approach has been shown to produce state-of-the-art detection accuracy even for challenging image data and is also used in subsequent works (Arteta *et al.*, 2013, 2016). We modify this approach on the algorithmic level, and as a result, fastER achieves similar detection, but higher segmentation accuracy in two out of three datasets used for evaluation. Additionally, we integrate our implementation into the linear time framework by Nistér and Stewénius (2008) for the calculation of *maximally stable extremal regions* (Matas *et al.*, 2002). Overall, fastER is orders of magnitude faster than the other tools evaluated here, while producing similar segmentation quality.

2 Materials and methods

2.1 Definitions

A 2D-greyscale image can be defined as a mapping $I : D \subset Z^2 \rightarrow S$, where D is the image domain and S is the range of possible greylevels, e.g. $S = [0, .., 255]$ for 8-bit greyscale images. *Connectivity* defines neighbourhoods between pixels: 4-connectivity specifies that each pixel is adjacent only to its four horizontal and vertical neighbours, while 8-connectivity specifies that each pixel is also adjacent to its four diagonal neighbours. Formally, connectivity defines an adjacency relation $A \subset D \times D$: two pixels p and q are adjacent to each other if and only if $(p, q) \in A$, which can also be written as pAq.

With this, *regions* (=*connected components*), *extremal regions* and *component trees* can be defined (Fig. 1). A region $x \subseteq D$ is a set of pixels fulfilling that for each pair of pixels $p, q \in x$ a path of adjacent pixels exists in x that connects p and q. Let $N(x)$ denote the set of pixels that are not in region x but adjacent to it: $N(x) = \{q \in D \backslash x : \exists p \in x : pAq\}$, where $\backslash$ denotes the set difference. Then, region x is *extremal* if and only if:

$$\forall\ q \in N(x) : I(q) > t_x, \tag{1}$$

where t_x denotes the maximal pixel intensity in x, i.e. $t_x = \max(\{I(p) : p \in x\})$. Equation 1 specifies that the intensities of all pixels that are adjacent to x (but not contained in x) have to be greater than t_x. From Equation 1 it follows that any extremal region x can be specified by a single *seed pixel*, i.e. any pixel $p \in x$ with $I(p) = t_x$: x is the set of pixels that can be reached from p through a path of adjacent pixels without exceeding the pixel intensity $I(p)$. Extremal regions can be used to define *component trees* (Najman and Couprie, 2006): a component tree is a directed tree representation of a greyscale image with one node for each extremal region, and a directed edge from the node corresponding to an extremal region a to the node corresponding to an extremal region b exists if and only if b is a true subset of a (i.e. $b \subset a$) and no other extremal region exists that contains b and also is a true subset of a.

2.2 Segmentation algorithm

We can now formally describe the segmentation algorithm used by fastER (Fig. 2A), including the modifications to the approach used by CellDetect. Given an image to segment, our aim is to extract a set of non-overlapping regions that is most likely to correspond to single cells. In fluorescence microscopy—and, when optimized image acquisition protocols are used (Supplementary Fig. S1), also in transmitted light microscopy—cells appear as regions that are darker or brighter than background. This fits the definition of extremal regions (Equation 1), and we therefore consider all extremal regions in the image as candidate regions for cell segmentation (to detect regions that are brighter than background, the image has to be inverted first). Additionally, each extremal region x is split in two distinct sub-regions x_1 and x_2 (i.e. $x_1 \cup x_2 = x \wedge x_1 \cap x_2 = \varnothing$) if x has exactly two child nodes c_1 and c_2 in the component tree: x_1 and x_2 then contain the extremal regions corresponding to the child nodes c_1 and c_2, respectively, and each remaining pixel of x is added to the closest sub-region. Additional constraints are required to ensure that the sub-regions remain connected (see Supplementary Note S1.1 for details). The two sub-regions x_1 and x_2 are then also used as candidate regions to increase segmentation accuracy, e.g. regions b_1 and b_2 in Figure 2A. This is an extension to CellDetect, which considers only maximally stable extremal regions as candidates, i.e.

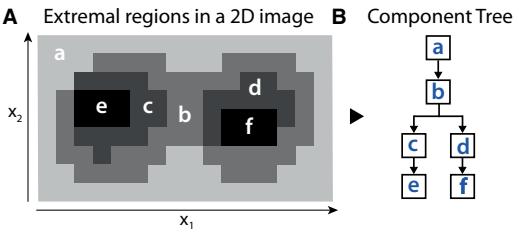

Fig. 1. Definitions. A *region* is a set of connected pixels, and an *extremal region* is a region with maximal size but containing only pixels whose intensities are below or equal to a specific threshold (see text for formal definitions). (**A**) A 2D-image with six extremal regions (*a-f*). Every extremal region can contain other extremal regions, which are by definition subsets of their parent extremal regions (e.g. *c* contains *e*). (**B**) Thus, any greyscale image can be represented as a directed tree (*component tree*), where each node corresponds to an extremal region. The top node always corresponds to the extremal region which contains all pixels and all other extremal regions of the image (*a* in the example)

a subset of all extremal regions. The latter, however, can impair segmentation accuracy, e.g. for cells with low contrast (compare Supplementary Fig. S1).

To estimate the likelihood (or score) $S(x)$ that a candidate region x corresponds to a single cell, a feature vector is extracted from it (Table 1). It is used as input for a support vector machine (SVM) with a Gaussian kernel (Burges, 1998), which is trained in a separate step (see Training algorithm). A positive score $S(x) > 0$ indicates that the candidate region x corresponds to a single cell, whereas a negative score $S(x) \leq 0$ indicates that it corresponds to background or multiple cells. We use LIBSVM (Chang and Lin, 2011) in our implementation of fastER. Any machine learning method or mathematical model that allows calculation of scores from feature vectors could be used, but the SVM allows comparably fast calculation of scores, and the Gaussian kernel avoids assumptions about the distribution of features that may not hold (e.g. linear separability). Unlike CellDetect, which uses 92 features, fastER uses only nine features (Table 1) that describe texture, gradient and shape of candidate regions and are informative for their classification. This small set of features enables high segmentation quality, but also high training and segmentation efficiency (see Results).

Given the candidate regions contained in an image and their scores, our aim is to select a high-scoring, but non-overlapping subset of them. For this purpose, we use a recursive formula $F(x)$. Given the component tree of the image, $F(x)$ selects a set of non-overlapping candidate regions contained in extremal region x with a divide and conquer approach. To segment an image, $F(x)$ is applied on the whole image, which is an extremal region by definition (equation 1). Let $S(x)$ denote the score of candidate region x, $\bar{S}(X)$ the average score of the set of candidate regions X (we furthermore define that $\bar{S}(\varnothing) = 0$) and $C(x)$ the set of extremal regions

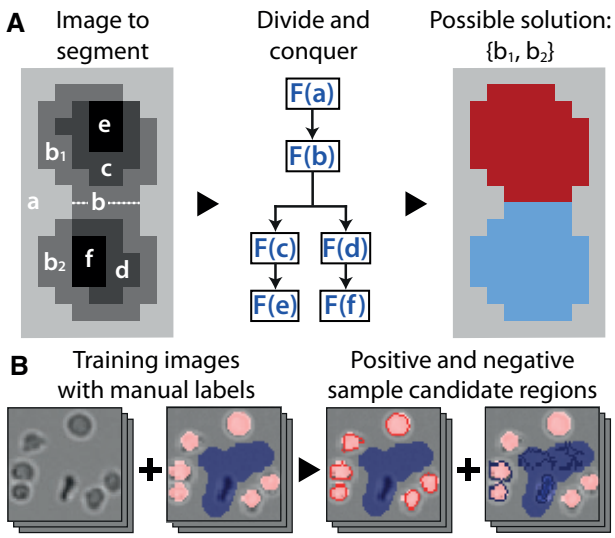

Fig. 2. Method overview. (**A**) For segmentation, all extremal regions (*a* to *f*) and, under certain conditions, split extremal regions ($\{b_1, b_2\}$) are used as candidate regions. From each, a feature vector (Table 1) is extracted and processed by a support vector machine (SVM) to estimate its likelihood to be a cell. Based on these likelihoods, an optimal set of non-overlapping regions is calculated with a divide and conquer approach (e.g. $\{b_1, b_2\}$). (**B**) For training, the user marks cells (bright) and background regions (dark) in one or more training images. Candidate regions best fitting to the markings are then extracted from the images (bright and dark outlines) and used as positive (i.e. corresponding to single cells) or negative (i.e. corresponding to background or multiple cells) samples for training of the SVM (Color version of this figure is available at Bioinformatics online.)

corresponding to child nodes of x in the component tree. With this, $F(x)$ can be defined as follows:

$$F(x) = \underset{X \in \Delta(x)}{\operatorname{argmax}} \bar{S}(X), \tag{2}$$

where $\Delta(x)$ is defined as follows if x has sub-regions x_1 and x_2 with positive scores (i.e. $S(x_1) > 0 \wedge S(x_2) > 0$):

$$\Delta(x) = \{\{x\}, \{x_1, x_2\}, \bigcup_{c \in C(x)} F(c)\}, \tag{3}$$

and otherwise as:

$$\Delta(x) = \{\{x\}, \bigcup_{c \in C(x)} F(c)\} \tag{4}$$

When applied on an extremal region x, $F(x)$ thus selects the set of candidate regions with the highest average score among up to three possibilities: x itself; the sub-regions x_1 and x_2 created by splitting x (if the conditions to split x are met, and if $S(x_1) > 0$ and $S(x_2) > 0$); and the regions obtained by applying $F(x)$ to each extremal region corresponding to a child node of x in the component tree (which is either a set of regions with positive scores or an empty set, e.g. if x has no nested extremal regions). In Figure 2A, for example, $F(b)$ could resolve to $F(b) = \operatorname{argmax}_{X \in \Delta(b)} \bar{S}(X)$ with $\Delta(b) = \{\{b\}, \{b_1, b_2\}, F(c) \cup F(d)\}$. Thus, the formula $F(x)$ extracts a set of non-overlapping candidate regions with approximately maximal average score under the constraint that for any extremal region $y \subseteq x$ with $S(y) > 0$, it holds that either y itself, a parent extremal region containing y or at least one child region contained in y is included in the solution. This also holds for any pair of sub-regions y_1 and y_2 with $S(y_1) > 0$ and $S(y_2) > 0$. Without this constraint, only one region with the highest score would be returned in most cases. There can be cases, where the solution extracted by fastER does not have globally maximal average score under the additional constraint—this prevents undesirable segmentation results (see Supplementary Note S1.3 for an example). This is an important difference to CellDetect, which extracts a set of non-overlapping regions with globally maximal sum of scores. The latter, however, favours solutions with many regions and can thus increase the risk of over-segmentation (Supplementary Fig. S2).

2.3 Efficient calculation

In the previous section, we defined our segmentation algorithm with a recursive formula based on the component tree (Equation 2), which can be calculated in quasi-linear time, i.e. $\mathcal{O}(n \log(n))$, where n is the number of pixels in the image (Najman and Couprie, 2006). There is, however, a faster way: Nistér and Stewénius (2008) have proposed an algorithm to calculate maximally stable extremal regions that is linear in the number of pixels (i.e. $\mathcal{O}(nm)$, where n is the number of pixels and m the number of greylevels (Carlinet and Geraud, 2014)) and has proven to be fast in practice. It processes each node of the component tree beginning at a leaf node and ending at the root node containing the whole image (see Nistér and Stewénius, 2008 for implementation details). When the algorithm processes a node, all its pixels are known and all its child nodes have been processed already. Therefore, it is perfectly suited to apply our recursive segmentation formula (Equation 2), and we now summarize the changes that are required to adapt it for this purpose (see Supplementary Note S1.4 for complete details).

The algorithm processes all nodes of the component tree without ever fully calculating it: at most one branch of it is constructed at a time. To enable this, each component has to be merged into its parent component after it has been processed. We could thus adapt the algorithm for

Table 1. Features used to classify candidate regions

Feature	Description
Size	Number of pixels in the candidate region
Major axis length	Major axis length of an ellipse with the same second order moments as the candidate region
Minor axis length	Minor axis length of an ellipse with the same second order moments as the candidate region
Avg. intensity	Mean pixel intensity in the candidate region
Heterogeneity	Standard deviation of the pixel intensities in the candidate region
Avg. gradient	Mean absolute pixel intensity difference to pixels adjacent to the candidate region
Std. dev. of gradient	The standard deviation of the absolute pixel intensity differences to pixels adjacent to the candidate region
Eccentricity	Number of pixels adjacent to the candidate region divided by the number of pixels adjacent to a perfect circle with the same area
Avg. background	Average intensity of pixels adjacent to the candidate region

Note: For formal definitions, see Equations 6–14 or Supplementary Note S1.2.

our segmentation by storing a list of contained pixels in each component, and, after the component has been processed, merging this list into the parent component's pixel list. Additionally, it would then be required for every component to iterate over all its pixels in order to calculate the features for classification of the corresponding candidate regions. Both operations would be computationally expensive. Therefore, we derived a set of variables with constant size that is stored in each component and is sufficient to calculate the features, so that a list of pixels is not required (these variables are therefore *sufficient statistics*). Given a region $x \subseteq D$, let $G(x)$ denote all pairs of pixels in x and $D \backslash x$ that are adjacent: $G(x) = \{(p,q) : p \in x, q \in D \backslash x, pAq\}$. With this, the sufficient statistics $s_i(x)$, with $i = 0, .., 11$, for candidate region x are defined as follows:

$$\begin{aligned} s_0 &= |x|,\ s_1 = \sum_{p \in x} I(p),\ s_2 = \sum_{p \in x} I(p)^2,\ s_3 = \sum_{p \in x} p_x, \\ s_4 &= \sum_{p \in x} p_y,\ s_5 = \sum_{p \in x} p_x^2,\ s_6 = \sum_{p \in x} p_y^2,\ s_7 = \sum_{p \in x} p_x p_y, \\ s_8 &= |G(x)|,\ s_9 = \sum_{(p,q) \in G(x)} |I(p) - I(q)|, \\ s_{10} &= \sum_{(p,q) \in G(x)} (I(p) - I(q))^2,\ s_{11} = \sum_{(p,q) \in G(x)} I(q), \end{aligned} \tag{5}$$

where p_x and p_y denote the x- and y-coordinate of pixel p, respectively. From the sufficient statistics, the features (Table 1) can be calculated in $\mathcal{O}(1)$:

$$\text{Size} = s_0 \tag{6}$$

$$C_{1,1} = \frac{1}{s_0 - 1}\left(s_5 - \frac{s_3^2}{s_0}\right)$$

$$C_{1,2} = C_{2,1} = \frac{1}{s_0 - 1}\left(s_7 - \frac{s_3 s_4}{s_0}\right)$$

$$C_{2,2} = \frac{1}{s_0 - 1}\left(s_6 - \frac{s_4^2}{s_0}\right)$$

$$\text{Major axis length} = \text{Major eigenvalue of } C \tag{7}$$

$$\text{Minor axis length} = \text{Minor eigenvalue of } C \tag{8}$$

$$\text{Avg. intensity} = \frac{s_1}{s_0} \tag{9}$$

$$\text{Heterogeneity} = \frac{1}{s_0 - 1}\left(s_2 - \frac{s_1^2}{s_0}\right) \tag{10}$$

$$\text{Avg. gradient} = \frac{s_9}{s_8} \tag{11}$$

$$\text{Std. dev. of gradient} = \frac{1}{s_8 - 1}\left(s_{10} - \frac{s_9^2}{s_8}\right) \tag{12}$$

$$\text{Eccentricity} = \frac{s_8}{8\sqrt{s_0/\pi}} \tag{13}$$

$$\text{Avg. background} = \frac{s_{11}}{s_8} \tag{14}$$

Note that C is the 2×2 covariance matrix of the x- and y-coordinates of the pixels in x. Also note that the definition of Eccentricity (Equation 13) is specific to 4-connectivity: in that case, if a region x is a perfect circle, $G(x)$ contains approximately $4d = 8r = 8\sqrt{A/\pi}$ elements, where d, r and A denote the diameter, radius and size of x, respectively.

In each component, sufficient statistics are stored for the extremal region x corresponding to the component and, if available, for its two sub-regions x_1 and x_2. They are updated when a pixel is added to the component, and to merge a component into its parent component, their sufficient statistics are simply added together (see Supplementary Note S1.4 for details); both operations are in $\mathcal{O}(1)$. Additionally to the sufficient statistics, the list of regions obtained by applying the recursive segmentation formula (Equation 2) is stored in each component together with their sum of scores. This has little impact on runtime, because every extremal region can be specified by a single seed pixel as described previously, so it is not required to store a list of all contained pixels for each extracted region (this also holds for sub-regions). The concept of sufficient statistics has previously been used in computer vision (Cipolla *et al.*, 2013). Our contribution is the definition of such statistics (including the update rules required to maintain them) for the extraction of all features used by fastER from extremal regions. This is not trivial for features involving the neighbouring pixels of candidate regions (Equations 11–14) and the corresponding sufficient statistics s_8 to s_{11} (compare Supplementary Equations S4–S7). To the best of our knowledge, this has not been described before for extremal regions.

With this, we can efficiently enumerate the nodes of the component tree and the corresponding candidate regions, extract their features, calculate their scores, and apply our recursive segmentation formula (Equation 2). The complexity of our implementation is $\mathcal{O}(nm)$, where n is the number of pixels and m the number of possible greylevels in the image (see Supplementary Note S1.4 for runtime analysis and Results for evaluation of practical runtime).

2.4 Training algorithm

In this section, we describe how to adapt our segmentation algorithm to specific datasets without manual specification of parameters. For this purpose, we developed an algorithm to extract positive and negative sample extremal regions from an image that fit best to cell and background labels added by the user to train fastER

(Fig. 2B). From these, feature vectors (Table 1) are extracted and used to train the SVM. For a training image I, let P_i (with $i = 1, .., n$) denote the set of positive labels (i.e. a set of regions labelled as corresponding to single cells) and N the set of pixels marked as negative. As negative training samples, we use all extremal regions in I that overlap with more than 20% of pixels with N, or with multiple positive regions. To extract positive training samples, we consider for each positive label P_i a set of candidate regions $C_i(I)$:

$$\begin{aligned} C_i(I) \quad &= \{x \in ER(I) : \ (x \cap P_i \neq \varnothing) \ \wedge \\ &\neg(\exists \ j \neq i : x \cap P_j \neq \varnothing) \ \wedge \\ &(x \cap N = \varnothing)\}, \end{aligned} \tag{15}$$

where $ER(I)$ denotes the set of all extremal regions in I. Thus, $C_i(I)$ is the set of extremal regions in I that overlap with P_i, but with no other positive label and with no negatively labelled pixel. For each positive label P_i, we then count for each corresponding candidate region x in $C_i(I)$ true positives tp_x (i.e. pixels in x and in P_i), false positives fp_x (i.e. pixels in x, but not in P_i) and false negatives fn_x (i.e. pixels in P_i, but not in x). Based on this, we calculate for every candidate region x sensitivity, specificity and F-score:

$$se_x = \frac{tp_x}{tp_x + fn_x}, \ sp_x = \frac{tp_x}{tp_x + fp_x}, \ F_x = \frac{2 * se_x * sp_x}{se_x + sp_x} \tag{16}$$

Finally, we select for each positive label P_i the candidate region with the highest F-score, and the union of these regions is used as positive training samples $S_P(I)$:

$$S_P(I) = \bigcup_{1 \leq i \leq n} \operatorname*{argmax}_{x \in C_i(I)}(F_x) \tag{17}$$

For efficient calculation, the training algorithm can also be integrated into the framework described in Nistér and Stewénius (2008).

3 Results

Figure 3 illustrates that fastER supports various cell and image acquisition types and is robust against common challenges for automated cell segmentation in large-scale microscopy. For quantitative comparison of segmentation speed and quality, we chose five state of the art methods: *U-Net* (Ronneberger *et al.*, 2015; Çiçek *et al.*, 2016), a recently published deep learning method using a convolutional neural network (CNN) that has outperformed other approaches on very challenging light microscopy images of cells, is comparably fast, and can be trained from sparse annotations; *Ilastik* (Sommer *et al.*, 2011), which is a general purpose segmentation tool that also uses machine learning and supports interactive training; *CellProfiler* (Carpenter *et al.*, 2006), which supports various algorithms and is one of the most widely used tools for cell segmentation; *CellX* (Dimopoulos *et al.*, 2014; Mayer *et al.*, 2013), which implements a recently published method that produces highly accurate segmentation results when applied on suitable images; and CellDetect because of its algorithmic similarity with fastER.

For evaluation, we used two sets of real and one set of synthetic cell microscopy images: dataset *PC* (Fig. 4A) from Arteta *et al.* (2012) contains 22 phase contrast images (each 400×400 pixels) of HeLa cells; dataset *BF* (Fig. 4B) from a long-term time-lapse microscopy experiment contains 17 brightfield images (each 1388×1040 pixels) of murine blood stem and progenitor cells; and dataset *FL* (Fig. 4C) contains 20 synthetic fluorescence images (each 1200×1200 pixels) that were generated with SimuCell (Rajaram *et al.*, 2012) (see Supplementary Note S2.1). Datasets PC and BF exhibit high cell densities, varying cellular morphologies and uneven image illumination. While strong gradients of image illumination as in dataset BF should ideally be prevented during image acquisition, they often cannot be avoided without excluding typical microscopy and incubation hardware or discarding large parts of imaging data. The dataset is a good benchmark for robustness against such problems. The cells in dataset FL have different sizes as well as low and varying signal-to-noise ratios (SNRs), which are typical for images of primary cells. We chose to use synthetic fluorescence images, because the precise manual segmentation required for ground truth generation can be difficult for cells with low SNRs (consider e.g. Fig. 3E).

3.1 Evaluation of segmentation speed

To evaluate practical runtime, we selected three images with low, medium and high cell densities, respectively, from dataset BF and measured for each method the average segmentation time per image. U-Net requires GPU acceleration for optimal performance and was run on an Nvidia GeForce GTX 1080 (8 GB) graphics card. The remaining methods were run on an Intel Core i5-3380M CPU (2.90 GHZ) ensuring that only one CPU core could be used by each implementation to make the results comparable. This was repeated 50 times for each method to average out random fluctuations. We used the same parameters for each method as for the evaluation of segmentation quality (see next section). Time for image preprocessing (i.e. denoising for fastER and illumination correction for CellProfiler) was included, but time for post-processing (filling of holes and size filter) was only included for fastER, because it proved negligible for the other methods. The results (Table 2) show that fastER outperformed the other methods. U-Net is relatively fast as well when compared with the single-core CPU speeds of the other methods. However, it is also clearly outperformed by fastER, especially when batch-processing images using all cores of a more powerful CPU. It is important to point out that this evaluation only considered segmentation, but not training speed.

3.2 Evaluation of segmentation quality

Each dataset was divided into a training and a test set, and to evaluate a method on a dataset, it was first tuned using only the corresponding training set (see Supplementary Note S2.2 for complete details). fastER first denoises images with bilateral filtering (Tomasi and Manduchi, 1998) using parameters automatically derived from the training labels. For U-Net, pixel weight maps required to learn separation of adjacent cells were generated. The data deformation and value augmentation layers included with 3D U-Net were added to the 2D U-Net network, and their parameters were adjusted for each dataset. In dataset BF, the mean was subtracted from each image, and dropout layers were disabled, as this seemed to improve results strongly for this dataset. For Ilastik, a foreground and a background class were created. All features with all sizes were selected for datasets PC and FL. Features not improving segmentation quality were excluded for dataset BF to increase segmentation speed since this dataset was also used for runtime evaluation. Results were exported using the *Simple segmentation* option. For CellX and CellDetect, images of dataset FL were first denoised with Gaussian blurring. For CellProfiler, after illumination correction in dataset BF and Gaussian blurring in dataset FL, seeds were detected with *IdentifyPrimaryObjects*, and the results were refined with *IdentifySecondaryObjects* using Watershedding (Vincent and Soille, 1991). The segmentation results of all methods were post-processed by filling holes (if not already done) and subsequent size filtering

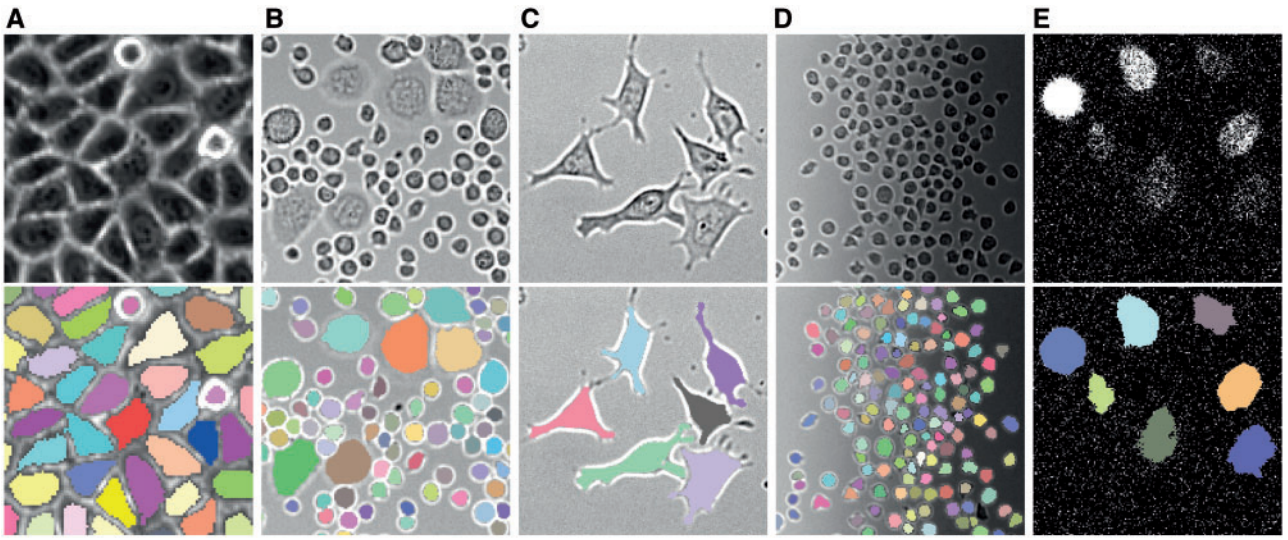

Fig. 3. fastER is robust against challenges for automated image segmentation that are typical in large-scale cell microscopy and supports various cell and image acquisition types. The upper row shows image patches from different cell microscopy experiments and the lower row the corresponding segmentation results obtained with fastER (see Table 3 for quantitative evaluation of segmentation quality). Each image patch illustrates one typical problem (contrast of all images was enhanced for display only). (**A**) Phase-contrast image of HeLa cells from (Arteta *et al.*, 2012) with high cell densities. (**B**) Brightfield image of differentiating murine blood progenitor cells with varying cellular morphologies. (**C**) Brightfield image of murine embryonic stem cells with complex cellular shapes. (**D**) Brightfield image of murine blood progenitor cells with varying image illumination. (**E**) Fluorescence image of murine embryonic stem cells expressing $\text{Nanog}^{\text{KATUSHKA}}$ fusion proteins (Filipczyk *et al.*, 2013) with low and varying signal-to-noise ratios (Color version of this figure is available at *Bioinformatics* online.)

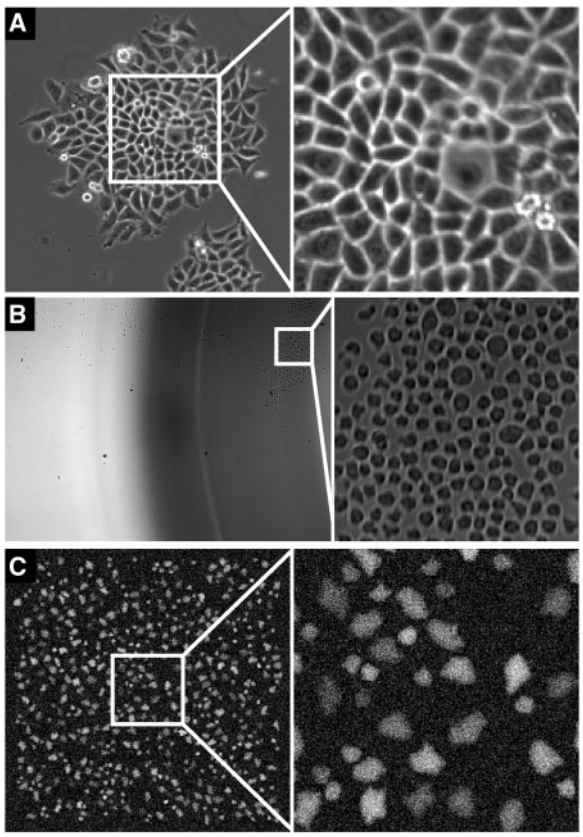

Fig. 4. Example images from the datasets used for evaluation of segmentation quality (Table 3). (**A**) Dataset *PC* from Arteta *et al.* (2012) contains 22 phase contrast images of HeLa cells. (**B**) Dataset *BF* from a long-term time lapse microscopy experiment contains 17 brightfield images of blood stem and progenitor cells. (**C**) Dataset *FL* contains 20 synthetic fluorescence images. Contrast of all images was enhanced for display only

with mostly identical parameters (these were only adjusted for individual methods if the detection accuracy would have been impaired otherwise). The parameters of CellX and CellProfiler were tuned extensively (Ilastik and CellDetect required only labelled training images). The results of each method were then compared with a ground truth segmentation (which was manually created for datasets PC and BF). The additional preprocessing steps were added to the other methods specifically for each dataset to optimize their results, if possible, and thus enable fair evaluation of their segmentation quality. However, this extensive tuning of segmentation methods requires expert knowledge and is thus not feasible for many typical users. Additionally, tuning a large numbers of parameters (as required e.g. for CellProfiler) is very time consuming, but we did not consider the time required for this in the evaluation.

To quantify detection accuracy, we counted for each method and dataset true positives (i.e. detected cells containing the centroid of exactly one of the ground truth cells), false positives (i.e. detected cells containing no centroids of the ground truth cells) and false negatives (i.e. ground truth cells whose centroids were not contained in any detected cell). From this, we calculated precision, recall and F-score, which summarizes the detection accuracy (formulas shown in equation 16). Since this does not take into account merge errors, we counted them separately: if a detected cell contained $n > 1$ centroids of reference cells, the number of merge errors was increased by $n - 1$. To quantify segmentation accuracy, we calculated the Jaccard index for each detected cell A that contained exactly one centroid of a ground truth cell B: $J(A, B) = |A \cap B|/|A \cup B|$.

It quantifies pixel-wise overlap between a detected cell and the corresponding reference cell and is always between 0 and 1, where 0 indicates no overlap and 1 perfect match. Outlines were only annotated for a random subset of ground truth cells (i.e. for 465 out of 1142 and 1188 out of 1464 cells in the test sets of datasets PC and BF, respectively) and only centroids for the remaining cells. All cells in the test sets were then used to calculate F-scores and merge errors, but in dataset PC and BF only completely labelled cells were used to

Table 2. Evaluation of segmentation speed

Method	On CPU[a] or GPU[b]	Avg. time per image [s]	Avg. time per image relative to fastER
fastER	CPU	0.363 ± 0.03	×1
U-Net[c]	**GPU**	7.13 ± 0.02	×19.6
CellProfiler	CPU	11.8 ± 0.6	×32.4
CellDetect	CPU	47.2 ± 0.8	×129.9
ilastik	CPU	52.5 ± 2.0	×144.4
CellX	CPU	94.9 ± 2.3	×261.1

Note: The average time required by each method to segment a single image from dataset BF (Fig. 4B) was measured (see text for details).

[a]Using one core of an Intel Core i5 CPU (2.90 GHZ) for all methods.

[b]Using an Nvidia GeForce GTX 1080 (8 GB).

[c]Note, however, that the CPU and GPU times measured here are hardly comparable: e.g. batch-processing images of the same size with fastER using all four cores of a Core i7-4790K (4 GHZ) is about 150 times faster than with U-Net running on the GeForce GTX 1080.

Table 3. Evaluation of segmentation quality

Dataset	Method	F-score[a]	Jaccard index[a]	Merge rate[a]
PC n = 1142	fastER	97.2	**74.9±11.2**	2.01
	ilastik	97.2	73.5 ± 10.9	1.93
	CellX	86.1	49.4 ± 11.4	3.33
	CellDetect	97.1	67.4 ± 11.7	2.01
	CellProfiler	76.6	69.5 ± 14.0	**0.61**
	U-Net	**97.4**	74.4 ± 9.1	2.19
BF n = 1464	fastER	97.5	81.6 ± 9.2	0.48
	ilastik	**97.6**	75.2 ± 8.7	0.41
	CellX	87.9	46.3 ± 23.8	**0.00**
	CellDetect	95.8	**83.5±6.9**	2.32
	CellProfiler	78.2	63.0 ± 14.0	0.41
	U-Net	97.0	82.7 ± 6.9	1.91
FL n = 5000	fastER	99.8	87.7 ± 9.7	0.14
	ilastik	**99.9**	88.8 ± 5.1	0.04
	CellX	92.8	78.7 ± 11.7	**0.00**
	CellDetect	99.8	71.3 ± 12.6	**0.00**
	CellProfiler	99.5	82.1 ± 10.5	0.34
	U-Net	**99.9**	**92.1±3.8**	0.32

Note: Detection accuracy (F-score), segmentation accuracy (Jaccard index) and merge errors (relative to the total number of cells) were quantified for 3 challenging datasets (n denotes the total number of cells in each dataset; see text for details). The best values in each dataset are shown in bold.

[a]Multiplied by 100 for better readability.

calculate Jaccard indexes (the test set of synthetic dataset FL contained 5000 cells with complete ground truth).

In Table 3, F-score, average Jaccard index with standard deviation and merge errors are reported for each method and dataset. For dataset PC, fastER, ilastik and U-Net achieved the highest F-Scores and Jaccard indexes, producing nearly identical segmentation quality. CellProfiler produced fewer merge errors than all other methods, but performed poorly regarding the other measures. For dataset BF, ilastik achieved the highest F-score, which was, however, again very similar to the F-scores of fastER and U-Net. CellDetect achieved the highest Jaccard index, but also a slightly lower F-score and more merge errors than any other method. U-Net achieved the second highest Jaccard index, but produced also a comparably high number of merge errors. For dataset FL, U-Net and ilastik achieved the highest F-score, but the value was again very similar to that of fastER. U-Net achieved the highest Jaccard index, but produced again a comparably high number of merge errors.

The detection accuracy of fastER was highly comparable with the best in all datasets. CellDetect and U-Net achieved higher segmentation accuracies in datasets BF and PC, respectively, but at the expense of more merge errors. Ilastik produced comparably high segmentation quality in all datasets, but no method clearly outperformed the others. Which method to choose for optimal results thus depends on the used cell and image acquisition type, but also on the purpose of cell segmentation. Automated cell tracking, for example, is typically stronger affected by merge errors than by low segmentation accuracy.

3.3 Population dynamics of in-vitro blood formation

As proof of concept, we analyzed the population dynamics of in-vitro blood formation using long-term time-lapse microscopy. Primary murine blood stem and progenitor cells (purified by CD150+ CD34- cKIT+ Sca-1+ Lin- CD48-/low phenotype from mouse bone marrow) were cultivated in differentiation conditions for six days and imaged with brightfield microscopy about every three minutes. This resulted in over 200 000 images (1388 × 1040 pixels each), in which cells were then segmented and counted using fastER (see Supplementary Note S3 for details). The results (Fig. 5) reveal that the number of cells remained approximately constant for about one day and grew exponentially afterwards. In total, more than 46 000 000 single cells were segmented and counted, but the analysis required only 2 h and 37 min on a regular desktop computer (Intel Core i7-4790K @ 4.00 GHZ, 16 GB RAM, Windows 7 64-bit).

4 Discussion

Reliable and fast automated cell segmentation in high-throughput imaging data is crucial for many areas of biological and medical research (Etzrodt *et al.*, 2014; Hoppe *et al.*, 2014). Due to inherent problems (e.g. high cell density, cell-to-cell heterogeneity, low and variable signal-to-noise ratios, illumination gradients, changing imaging modalities, etc.) of typical cell imaging data, unsupervised segmentation solutions working for many current and future imaging experiments without extensive parameter tuning do not exist, and will likely remain impossible. Adaptable algorithms that allow efficient and user-friendly training on new datasets are therefore crucial to enable cell segmentation not only in individual specific datasets (Meijering, 2012).

We therefore developed fastER, which can be adapted efficiently to different imaging datasets and then allows fast and accurate automated segmentation in large volumes of data. Quantitative evaluation confirmed that fastER is orders of magnitude faster than existing methods, while producing state of the art segmentation quality even for challenging image data. fastER thus enables analysis of large-scale experiments without access to high performance computing facilities, which are not available in most typical laboratories. Importantly, it also enables on the fly analysis of running experiments. fastER does not use the structured SVM framework (Tsochantaridis *et al.*, 2004) employed by CellDetect and much less features. It can therefore be trained much faster (e.g. for dataset BF, training required only about 0.6 s for fastER compared to 13 min for CellDetect on the computer used for the proof of concept). It can thus display segmentation results in a real-time live preview when the user adds, changes or removes a training label (Supplementary Movie S1). This gives crucial immediate feedback on whether the method is applicable (i.e. when fitting candidate regions can be found), where to add new labels (i.e. where segmentation errors

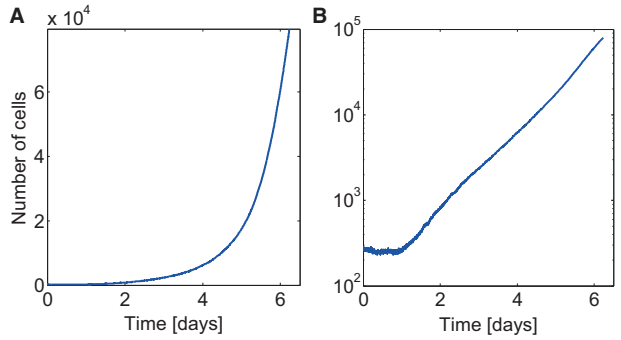

Fig. 5. Population dynamics of in-vitro blood formation. Differentiating and proliferating blood and progenitor cells were cultivated in vitro for six days and imaged every three minutes with brightfield microscopy. The cells were then segmented and counted with fastER. The results, shown with regular (**A**) and logarithmic scale (**B**), reveal that the number of cells remained approximately constant for about one day and grew exponentially afterwards

occur), and when to stop training (i.e. when segmentation quality does not improve any more when adding new labels). The included training algorithm efficiently extracts training samples from rough user annotations. Importantly, and in contrast to many other methods, no other parameters or processing steps need to be specified manually. fastER therefore is efficient and easy-to-use also for non-experts. U-Net is also applicable to a wide range of analysis problems, but requires a much larger amount of precisely annotated training data, which is not available for many typical experiments. In addition, training takes several hours even with a high-end GPU, thus making iterative labelling and parameter tuning extremely inefficient. This is a major usability problem, because training is crucial to obtain optimal results for a specific dataset.

fastER supports images of various cell types from different imaging approaches, and is robust against common challenges for automated cell segmentation in large-scale microscopy. fastER works best with transmitted light images acquired slightly out of focus (Supplementary Fig. S1), but this does not increase acquisition time or photo-toxicity. As with other segmentation approaches, limitations remain, e.g. when high cell densities and blurring in fluorescence microscopy impair accurate segmentation with extremal regions. However, through interactive training with immediate feedback, users can quickly determine whether fastER is applicable to a given dataset.

We have implemented fastER in C++ for optimal speed and usability, but segmentation results are stored in standard formats and can thus easily be opened with existing image analysis software for further processing. fastER supports the experiment structure used by *tTt* and *qTfy* (Hilsenbeck *et al.*, 2016), but other experiments can be opened as well without requiring conversion. It includes a batch-processing mode that uses all available CPU cores for segmentation. We open-source fastER and provide binaries for Windows and Linux (it can also be compiled for OS X). Early versions of fastER are successfully used in several research projects of our and collaborating laboratories (Filipczyk *et al.*, 2015; Hoppe *et al.*, 2016), where it has become a core software component and will thus be maintained and extended for many years to come.

Funding

This work was supported by the SNF to TS. TS and OH acknowledge financial support for this project from SystemsX.ch.

Conflict of Interest: none declared.

References

Arteta,C. *et al.* (2012) Learning to detect cells using non-overlapping extremal regions. In: Ayache,N. *et al.* (eds) *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2012*. Springer, Berlin, Heidelberg, pp. 348–356.

Arteta,C. *et al.* (2013). Learning to detect partially overlapping instances. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3230–3237.

Arteta,C. *et al.* (2016) Detecting overlapping instances in microscopy images using extremal region trees. *Med. Image Anal.*, **27**, 3–16.

Buggenthin,F. *et al.* (2013) An automatic method for robust and fast cell detection in bright field images from high-throughput microscopy. *BMC Bioinformatics*, **14**, 297.

Burges,C.J.C. (1998) A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, **2**, 121–167.

Carlinet,E. and Geraud,T. (2014) A comparative review of component tree computation algorithms. *IEEE Trans. Image Process.*, **23**, 3885–3895.

Carpenter,A.E. *et al.* (2006) CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biol.*, **7**, R100.

Chang,C.-C. and Lin,C.-J. (2011) Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, **2**, 1–27.

Çiçek,Ö. *et al.* (2016) 3d u-net: Learning dense volumetric segmentation from sparse annotation. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 424–432. Springer.

Cipolla,R. *et al.* (2013) *Machine Learning for Computer Vision*. Springer, Berlin, Heidelberg.

Cohen,A.R. *et al.* (2010) Computational prediction of neural progenitor cell fates. *Nat. Methods*, **7**, 213–218.

Dimopoulos,S. *et al.* (2014) Accurate cell segmentation in microscopy images using membrane patterns. *Bioinformatics*, **30**, 2644–2651.

Eilken,H.M. *et al.* (2009) Continuous single-cell imaging of blood generation from haemogenic endothelium. *Nature*, **457**, 896–900.

Eliceiri,K.W. *et al.* (2012) Biological imaging software tools. *Nat. Methods*, **9**, 697–710.

Etzrodt,M. *et al.* (2014) Quantitative single-cell approaches to stem cell research. *Cell Stem Cell*, **15**, 546–558.

Filipczyk,A. *et al.* (2013) Biallelic expression of nanog protein in mouse embryonic stem cells. *Cell Stem Cell*, **13**, 12–13.

Filipczyk,A. *et al.* (2015) Network plasticity of pluripotency transcription factors in embryonic stem cells. *Nat. Cell Biol.*, **17**, 1235–1246.

Funke,J. *et al.* (2015). Learning to segment: training hierarchical segmentation under a topological loss. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 268–275. Springer.

Hilsenbeck,O. *et al.* (2016) Software tools for single-cell tracking and quantification of cellular and molecular properties. *Nat. Biotechnol.*, **34**, 703–706.

Hoppe,P.S. *et al.* (2014) Single-cell technologies sharpen up mammalian stem cell research. *Nat. Cell Biol.*, **16**, 919–927.

Hoppe,P.S. *et al.* (2016) Early myeloid lineage choice is not induced by random PU.1/GATA1 protein ratios. *Nature*, **535**, 299–302.

Li,F. *et al.* (2013) Chapter 17: bioimage informatics for systems pharmacology. *PLoS Comput. Biol.*, **9**, e1003043.

Liu,T. *et al.* (2012). Watershed merge tree classification for electron microscopy image segmentation. In: *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 133–137. IEEE.

Magnusson,K.E.G. *et al.* (2015) Global linking of cell tracks using the Viterbi algorithm. *IEEE Trans. Med. Imaging*, **34**, 911–929.

Matas,J. *et al.* (2002). Robust wide baseline stereo from maximally stable extremal regions. In: *Procedings of the British Machine Vision Conference 2002*, pp. 384–393. British Machine Vision Association.

Mayer,C. *et al.* (2013) Using cellX to quantify intracellular events. *Curr. Protoc. Mol. Biol.*, **14**, 1–21.

Meijering,E. (2012) Cell segmentation: 50 years down the road [Life Sciences]. *IEEE Signal Process. Mag.*, **29**, 140–145.

Myers,G. (2012) Why bioimage informatics matters. *Nat. Methods*, **9**, 659–660.

Najman,L. and Couprie,M. (2006) Building the component tree in quasi-linear time. *IEEE Trans. Image Process.*, **15**, 3531–3539.

Nistér,D. and Stewénius,H. (2008). Linear time maximally stable extremal regions. In: Forsyth,D. *et al.* (eds) *Computer Vision – ECCV 2008*. Springer, Berlin, Heidelberg, pp. 183–196.

Peng,H. (2008) Bioimage informatics: a new area of engineering biology. *Bioinformatics*, **24**, 1827–1836.

Rajaram,S. *et al.* (2012) SimuCell: a flexible framework for creating synthetic microscopy images. *Nat. Methods*, **9**, 634–635.

Rieger,M.A. *et al.* (2009) Hematopoietic cytokines can instruct lineage choice. *Science*, **325**, 217–218.

Ronneberger,O. *et al.* (2015). U-net: convolutional networks for biomedical image segmentation. In: Navab,N. *et al.* (eds) *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer, Cham, pp. 234–241.

Schiegg,M. *et al.* (2014) Graphical model for joint segmentation and tracking of multiple dividing cells. *Bioinformatics*, **31**, 948–956.

Schroeder,T. (2008) Imaging stem-cell-driven regeneration in mammals. *Nature*, **453**, 345–351.

Schroeder,T. (2011) Long-term single-cell imaging of mammalian stem cells. *Nat. Methods*, **8**, S30–S35.

Skylaki,S. *et al.* (2016) Challenges in long-term imaging and quantification of single-cell dynamics. *Nat. Biotechnol.*, **34**, 1137–1144.

Sommer,C. *et al.* (2011). Ilastik: Interactive learning and segmentation toolkit. In: *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 230–233. IEEE.

Tomasi,C. and Manduchi,R. (1998). Bilateral filtering for gray and color images. In: *Sixth International Conference on Computer Vision*, pp. 839–846. IEEE.

Tsochantaridis,I. *et al.* (2004). Support vector machine learning for interdependent and structured output spaces. In: *Proceedings of the twenty-first international conference on Machine learning*, p. 104.

Vincent,L. and Soille,P. (1991) Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.*, **13**, 583–598.

Zanella,F. *et al.* (2010) High content screening: seeing is believing. *Trends Biotechnol.*, **28**, 237–245.