Algorithms for cytoplasm segmentation of fluorescence labelled cells

Carolina Wählby [a,*], Joakim Lindblad [a], Mikael Vondrus [a], Ewert Bengtsson [a] and Lennart Björkesten [b]
[a] *Centre for Image Analysis at Uppsala University, Uppsala, Sweden*
[b] *Amersham Pharmacia Biotech, Uppsala, Sweden*

Received 2 August 2001
Accepted 24 June 2002

Abstract. Automatic cell segmentation has various applications in cytometry, and while the nucleus is often very distinct and easy to identify, the cytoplasm provides a lot more challenge. A new combination of image analysis algorithms for segmentation of cells imaged by fluorescence microscopy is presented. The algorithm consists of an image pre-processing step, a general segmentation and merging step followed by a segmentation quality measurement. The quality measurement consists of a statistical analysis of a number of shape descriptive features. Objects that have features that differ to that of correctly segmented single cells can be further processed by a splitting step. By statistical analysis we therefore get a feedback system for separation of clustered cells. After the segmentation is completed, the quality of the final segmentation is evaluated. By training the algorithm on a representative set of training images, the algorithm is made fully automatic for subsequent images created under similar conditions. Automatic cytoplasm segmentation was tested on CHO-cells stained with calcein. The fully automatic method showed between 89% and 97% correct segmentation as compared to manual segmentation.

1. Introduction

Flow cytometry is a reliable, reproducible and quantitative method for studies of the phenotypes that compose a heterogeneous cell population. Although largely applied to the analysis of single cells, flow cytometry has some drawbacks. The mapping of functional activities is limited to cells in suspension, and therefore removed from the tissue structure. There is an uncertainty due to artifacts such as cellular debris and clusters of cells in the cell suspension, and it is difficult to go back and have a closer look at signals which deviate from the normal after the analysis is completed, unless the full experiment is run again.

Another important source of information about cells is provided by fluorescence staining in combination with fluorescence microscopy, also called image cytometry. Visual evaluation of fluorescence microscopy images is tedious and the inter- and intra-observer variability is often high. Digital image analysis of images produced by a digital camera attached to a fluorescence microscope allows for fast automated high throughput detection and analysis of spatial, spectral and temporal distribution of emitted fluorescence from single cells. The basis for all automatic image analysis needed in high content cell screening applications is cell segmentation. Before any cell specific spatial, spectral and temporal features can be extracted, the individual cells have to be separated, i.e., segmented, from the image background and from each other. While the nuclei of the cells are often quite distinct and easy to detect, e.g., based on its regular shape [12,16], the cytoplasm provides a lot more challenge, especially if no nuclear counter stain is available. Regularly shaped cytoplasms can be segmented using methods similar to those for nuclear segmentation [17], but irregular shapes require different methods.

Segmentation is one of the most intensely studied problems in image analysis and still no robust general-purpose methods exist. Therefore various application-adapted methods which take advantage of the a priori knowledge about the images, are needed. In this study we have developed a sequence of processing steps that lead to an automatic cytoplasm segmentation of fluorescence microscopy cell images. Through a statisti-

[*]Corresponding author: Carolina Wählby, Centre for Image Analysis Lägerhyddv. 17 SE-75237 Uppsala, Sweden. Tel.: +46 18 471 3469; Fax: +46 18 553447; E-mail: carolina@cb.uu.se.

cal analysis of descriptive features and a feedback system for separation of clustered objects the results of the segmentation are improved. The general methodology is applicable also for other segmentation problems. The obvious next step, not within the scope of this study, is to perform the analysis of cell features in the spatial, temporal and spectral domains using the segmentation result.

2. Cell segmentation strategies

When extracting features of individual cells in an image, the first task is to find and define the individual cells, i.e., cell segmentation. The method for cytoplasm segmentation of fluorescence labeled cells presented in this paper consists of a number of processing steps as shown in Fig. 1. Errors such as over- and under-segmentation produced by the initial processing steps can be corrected after the automatic quality control and feedback step. A quality control routine after the final step provides a measure of the reliability of the segmentation result.

2.1. Image pre-processing

Large scale intensity variations and shading effects in the image caused by uneven illumination and other variations over the field of view are undesirable, as they may introduce problems for the segmentation step as well as position dependence for features such as integrated pixel intensity. Given that the uneven background is due to uneven illumination and the intensity

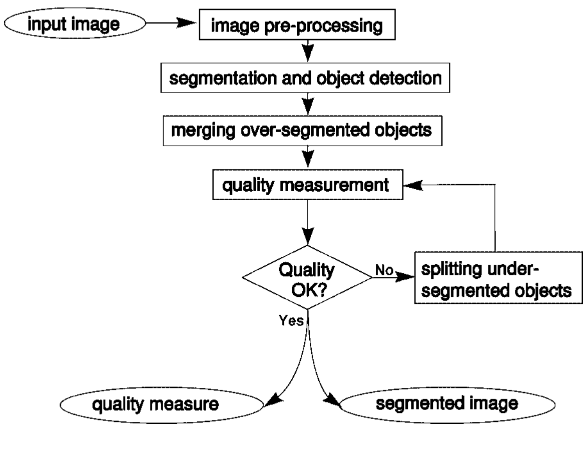

input image
image pre-processing
segmentation and object detection
merging over-segmented objects
quality measurement
Quality OK?
No
splitting under-segmented objects
Yes
quality measure
segmented image

Fig. 1. An overview of the steps in the presented cytoplasm segmentation algorithm. Only objects that are considered not cell-like (according to the quality measurement) are sent to the splitting step. The different steps are explained in the text.

of the fluorescence in each cell is a linear function of the light illuminating the cell, the usually accepted illumination correction is: $(I_{\text{raw}} - I_{\text{dark}})/(I_{\text{blank}} - I_{\text{dark}})$, where I_{dark} is an image without the illumination and I_{blank} is an image from a blank part of the slide with illumination. Whether it is feasible to acquire good images of the shading situation directly in conjunction with the imaging procedure, or not, is very much dependent on the imaging environment.

Data-driven approaches are usually simpler from a practical viewpoint, but may however introduce artifacts if the algorithm used cannot reliably solve the estimation problem. The shading correction methodology that has been used in this paper is entirely data-driven, and has shown to be both reliable and fast. A robust iterative method, described in [4] and [10], is used to find a good approximation of the background of the image. This background approximation does not give I_{blank} and I_{dark} separately, and the illumination correction described above must therefore be simplified either into a pure additive model where the background is subtracted or a pure multiplicative model where the background is removed by division. Similar results were achieved in the two cases, and we chose to use the additive model.

The algorithm works by iteratively making better and better estimates of the background of the image. The background is assumed to be smooth and slowly varying (if we have abrupt changes in the illumination, there is something wrong with the imaging device). B-spline surfaces [8] is a class of parametric surfaces which are easy to model and have many nice properties [5]. Cubic B-spline functions are ensured by definition to be continuous in the second derivative and therefore forced to be smooth. They also have the particularly useful property of minimizing the bending energy in 1D [18]. A cubic B-spline surface should therefore be suitable as a model for the background shading. A surface patch S of the background, is modeled by a tensor product of spline functions. I.e., a surface point $S(u,v)$ will be written as $S(u,v) = \sum_{kl} B_k(u)B_l(v)x_{kl}$. Where B_k are the B-spline blending polynomials and the x_{kl} are the control points of the surface. The number of control points for the B-spline surface defines how flexible it is. If the background varies a lot, many control points are needed, but in most cases only a few are enough. We used 5×5 evenly spread control points, which demonstrated to work well in all of the tested images. An example of the shading correction can be seen in Fig. 2.

To get a first estimate of the background, the spline surface is initially fitted to the whole image. The dis-

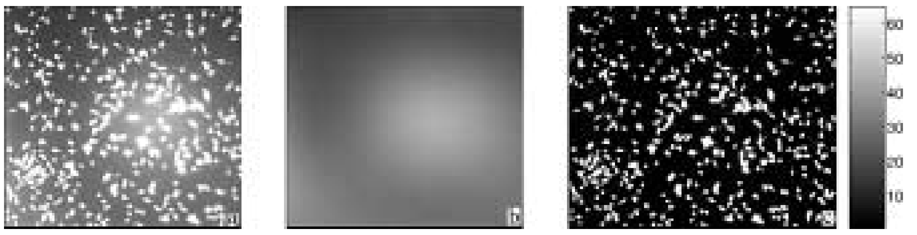


Fig. 2. Original image before background correction (a), the fitted spline surface of the background (b) and the image after background subtraction (c). Note that the intensity scale (same for all images) has been set to enhance the contrast of the darker parts of the images.

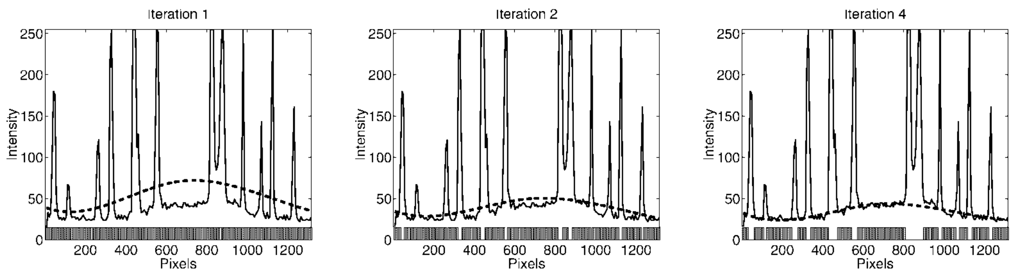


Fig. 3. The intensity profile across one image with fitted background (dashed) after the first, second and fourth (final) iteration of the background approximation algorithm. At the bottom of each graph is indicated which pixels from the original image that have been used for fitting the spline surface to the background.

tance between the spline surface and the background is minimized by least squares regression. This first estimate will give a too bright image of the background, as it also includes the brighter cells into the background. This first background is subtracted from the original image to get a first estimate of a background compensated image. The standard deviation of the pixel values in this image is calculated, and all pixels that are brighter than a constant number (in this case 1.7) of standard deviations are considered to belong to the foreground and are masked away. The value of 1.7 standard deviations can be motivated as the point of maximum change of trend in a Gaussian noise distribution. The algorithm is not too sensitive to this value, and it can be varied between 1 and 2 with marginal impact on the result, see [10].

The second iteration starts again with the original image, but this time the spline surface is only fitted to the pixels that have not already been masked away as foreground pixels. Therefore this second estimate will be a little bit better than the first estimate of the background. Once again this new background is subtracted from the original image and more foreground pixels are found and masked away. This iterative procedure continues until the average change in pixel value between two successively calculated backgrounds is less than half the original quantization step of the image. Convergence is fast and the stop criterium is usually reached after 4–10 iterations. An intensity profile across the image in Fig. 2 together with the approximated background after 1, 2 and 4 iterations can be seen in Fig. 3.

Visual inspection shows that this algorithm performs well on all tested images, and no comparison of the result with separately acquired illumination images of the scenes has been made. However, if the image contains information of a spatial frequency similar to the one of the background, the algorithm may not perform well. The evaluation performed in [10] indicates that the performance of the algorithm stays fairly robust under the imaging conditions of this study.

2.2. *Initial segmentation*

A method inspired by the watershed algorithm [13,20] was used for initial separation of clusters of cells and segmentation of the image into cells and background in one step. A 2D grey-level image can be thought of as a topographic relief where the cells with high intensity are peaks separated by lower intensity valleys. A grey-level image of a cluster of cells is shown in Fig. 4(a) and an intensity profile along a line in Fig. 4(a) is shown in Fig. 4(b). The cells can easily be separated from the flat, pre-processed background by a single intensity threshold, as can be seen in Fig. 4(c) and (d). It is however not possible to separate

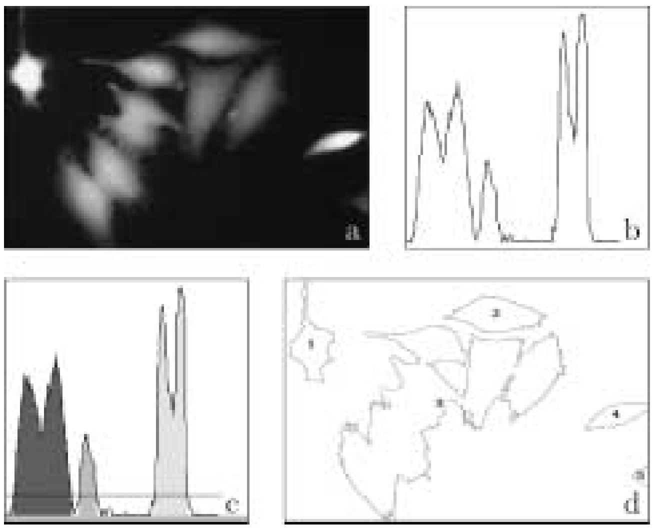


Fig. 4. Initial object detection. A grey-scale image of a cluster of cells is shown in (a). The variation in grey-scale intensity is illustrated by measuring the intensity along a line across (a), as shown in (b). If segmentation by a single threshold followed by labeling of connected components is applied to (a), it is not possible to separate and find all objects at once, as shown in (c) and (d). The different shades of grey represent separate objects in (c). Each number in (d) represents a single object found after thresholding of (a) and labeling of connected components.

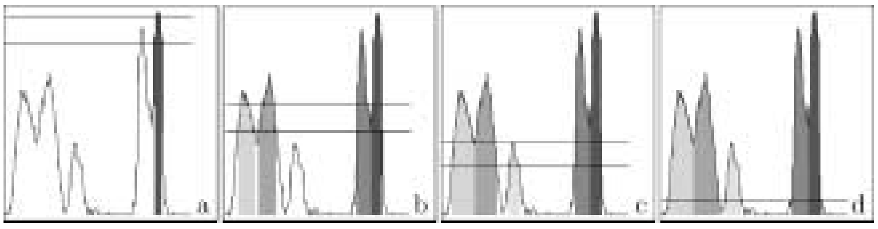


Fig. 5. Steps in the initial segmentation process. The two horizontal lines represent the upper and the lower threshold. New objects are found every time an unlabeled object reaches above the upper threshold. All pixels connected to a labeled object reaching above the lower threshold are given the same label. All the objects in the image are labeled as the thresholds slide down to the object-background threshold.

clusters of cells from each other using only one threshold since the intensity valleys between the peaks vary in depth. There are also “false” valleys due to noise and structures within the cells. These valleys are usually small and do not mark the border between two adjacent cells.

The segmentation algorithm used here is a watershed algorithm with double thresholds. It starts from the highest intensities, or peaks, in the image and places an upper threshold here. Labels are then assigned to all pixels that are 8-connected to a peak and have an intensity above a lower threshold. The upper and the lower threshold are then decreased by one, and if an unlabeled pixel with an intensity equal to the upper threshold is found, it is given a new label. The same label is thereafter given to all 8-connected pixels that are unlabeled and have an intensity greater than the lower threshold. The upper and the lower threshold are decreased by one and the process is repeated. The procedure is illustrated in Fig. 5(a–d). The two parallel lines represent the position of the upper and the lower threshold at the current step. Labels that have been assigned to the cells are represented by different shades of grey.

The upper and the lower thresholds are separated by a constant distance at all times except when the lower threshold reaches a pre-defined threshold for the background. The value of this background threshold should preferably be derived from the background compensation algorithm, but this is left as future work. In this paper it was set manually, once for each set of training images. The distance will then decrease for each iteration until also the upper threshold reaches the threshold for the background. The distance between the two thresholds decides the minimum intensity valley that can separate two adjacent cells. This reduces over-

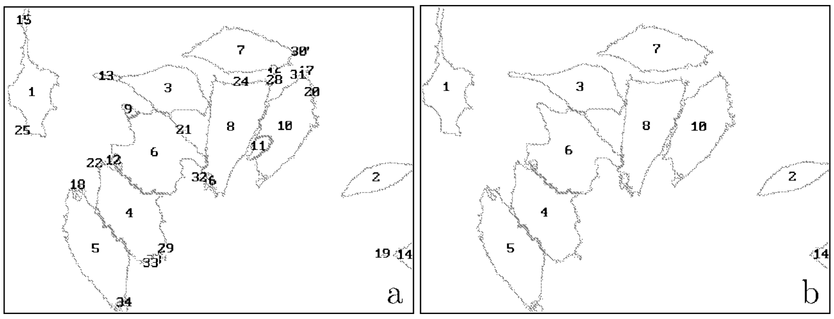


Fig. 6. Result of the initial segmentation process is shown in (a). Some objects are over-segmented, i.e., split in to several smaller parts, and a number of small noise objects are detected. The result after the merging step is shown in (b).

segmentation due to false valleys that would appear with the standard watershed algorithm. The distance between the upper and the lower threshold is dependent on the imaging conditions and is decided based on inspection of the training images. The fact that the distance between the two thresholds shrinks down to zero when the lower threshold reaches the background threshold makes the algorithm approach the standard watershed algorithm, resulting in a lot of small dark noise objects close to, but not touching, the cell boundary. These small noise objects are easily discarded by the merging step (see below). The result of the initial segmentation step when applied to Fig. 4(a) can be seen in Fig. 6(a). This initial segmentation algorithm usually results in both over- and under-segmentation as well as a lot of small noise objects. Note that this version of the watershed algorithm works directly on the grey-scale image, and does not make use of the distance transform for segmentation as described in [15].

2.3. Merging over-segmented cells

The initial segmentation algorithm often results in over-segmentation of cells as well as detection of small noise objects, as can be seen in Fig. 6(a). Over-segmentation is reduced by merging small objects with their neighboring objects. Selecting objects that should be merged or discarded by looking at cells with small area showed not to be the best solution, as the size of a cell varies as it goes through the cell cycle. Mitotic cells are usually very small but show high staining intensity while inter-phase cells are larger and lower in intensity. Therefore a better "size" measure is the integrated pixel intensity over the object, as this value stays fairly constant throughout the cell cycle. The minimum integrated pixel intensity is decided based on inspection of the training images.

Once a small object is found, its neighborhood is examined. If a single touching neighboring object is found, this label is put in a merging list. If no touching objects are found, a zero is placed in the merging list. If several touching objects are found, the label of the neighbor with the highest summed intensity of touching border pixels is put in the merging list. When all small objects have been examined the merging list is reduced to avoid exchange of labels between groups of small objects. Objects are then merged according to the merging list. A zero in the merging list means that the object should be discarded as noise. The result after merging, and removal of small noise objects, can be seen in Fig. 6(b).

2.4. Quality measure

As some of the cells are not correctly segmented after the initial segmentation and merging steps, the next step in the algorithm is to split under-segmented objects consisting of more than one cell. It is desirable to pick out these objects and send them to a splitting algorithm, to try to find a way to separate the cells in a correct manner. To send all the segmented cells to the splitting algorithm is not the best solution though. As the splitting algorithm is not 100% accurate, this may lead to splitting already correctly segmented cells and doing more harm than good. Also, it is of course more time consuming to pass all objects to the splitting step, than to only analyze objects that look odd in some way.

Erroneously segmented objects consisting of more than one true cell should exhibit some different properties than correctly segmented cells. Unfortunately, there exists no single feature which separates the correctly segmented cells from the incorrectly segmented clusters. We therefore need a multivariate approach, including many features into the classifier, to be able to separate the two classes. To include all features one could think of into the statistical measure is not a good thing, as this will most certainly lead to over training on the training population, and lower the performance on successive images. To get good generalization properties in the classifier, it is desirable to keep the number of features used as low as possible.

To be able to perform the selection of features in an automatic fashion, a way of judging the quality of the resultant classifier is needed. One way to get a unified measure of the discriminating property of a scalar measure is to draw a so called ROC curve (receiver operating characteristic) [18,19]. Here one does not select any single threshold, but rather use all possible thresholds. An ROC curve is a plot where for each possible threshold, the percentage of true positives is plotted against the percentage of false positives. If the classifier is efficient, the ratio of true positives to false positives should be large, and an efficient classification rule is correspondingly characterized by a large area under the ROC curve, this is called the accuracy (A) of the classifier. A bad classifier has an accuracy A near 0.5, which corresponds to random selection, whereas a perfect classifier has $A = 1$, i.e., no false positives at all.

Having a way to tell how good a classifier is, we can start to look for the optimal one. To decide what is cell-like and what is not, a statistical Mahalanobis distance in a multi-parameter feature space is calculated for each segmented object. This distance measure tells how much each object deviates from the center of a known population consisting of only correctly segmented cells. The distance measure is based on a selected set of features measured for a large number of correctly (i.e., manually) segmented cells in a training image. A mean value vector $\bar{\boldsymbol{x}}$ and a covariance matrix $\boldsymbol{S}$ is calculated from the features of the cells in the training image. For every object i to classify, a vector of feature values $\boldsymbol{x}_i$ is measured from the image and the squared generalized distance d^2 [6] is calculated as

$$d^2 = (\boldsymbol{x}_i - \bar{\boldsymbol{x}})' \boldsymbol{S}^{-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}), \quad i = 1, 2, \ldots, n. \tag{1}$$

The object is classified as cell-like if its statistical distance from the mean is less than a constant distance d_c. If the distance is greater than d_c, the object is considered not cell-like (and may, e.g., consist of more than one cell). Once a good classifier has been created for a specific image type, it can be used for classification of objects in a large number of similar images.

To verify the quality of the classifier, we must have both a training set and a test set of objects, along with the known true classification for each object. We can then pick a set of features, train the distance measure (i.e., construct the mean vector and covariance matrix) on the training set, and check how well it performs on the test set by calculating the accuracy A from the ROC plot. Unfortunately combinatorics is against us here, and we cannot afford to test all combinations of features as it would take far too long time. Instead we have taken the fairly standard approach of using a stepwise scheme of removing and inserting features into the classifier, to hopefully find a near optimal solution.

First, all plausible features (see Appendix) are included in the classifier. This will most certainly result in over-training, as it gives too many degrees of freedom for the classifier. One features at a time is therefore removed temporarily, and the accuracy of the classifier is tested on the training set. The feature that contributed the least to the accuracy is then removed. This is done over and over again until there is only one feature left. To be sure not to accidentally remove the best feature from the beginning, before removing another feature, it is always checked if the inclusion of one of the previously removed features will give an accuracy strictly higher than what we had before with the same number of features. I.e., we remove and put back features alternating, but when we put features back we always make sure that the accuracy goes up. Therefore the process will not go on forever, and finally we will have only one feature left.

For each number of features included in the classifier, we now know what features to use, and how well the classifier performs. This list is then backtracked to find an optimum with good performance, and a moderate number of features included in the classifier. This stepwise feature selection procedure is described in detail in [11].

2.5. *Features for segmentation quality measurement*

Starting from a set of 30 features, 11 were removed due to linear or near linear dependencies, as this would give a singular matrix in the Mahalanobis distance. The remaining 19 features are listed in the appendix. See Section 3 (Results) for the individual selections of features for each of the images tested.

When a good classifier is found, still a threshold must be set to decide which objects have a high probability of being correctly segmented. The segmentation quality is measured twice. First, when deciding which of the objects to send the splitting algorithm, and second, when deciding the over all segmentation quality.

At the first step, a rather low threshold can be set, i.e., it is acceptable if some of the already correctly segmented cells are sent to the splitting step. This is due to the fact that the splitting step will only split the object if the splitting will result in two new objects which are more cell like than the original one, see Section 2.6.

At the second quality measure, the particular application has to be considered when deciding how important it is that all correctly segmented cells are included in the further analysis. Including more correctly segmented cells will come at the cost of some incorrectly segmented cells being included in the analysis too, as the two populations almost always overlap. On the other hand, if the application requires all cells to be correctly segmented, a high threshold must be set, and thus, many correctly segmented cells will be excluded together with the incorrectly segmented cells.

2.6. Splitting of under-segmented objects

As some of the cells are not correctly segmented after the initial segmentation and merging step, the next step in the algorithm is to split under-segmented cells. Only objects that have a low statistical quality score, i.e., objects that have a high probability of consisting of clusters of cells, are sent to the splitting step.

Separation of touching or overlapping objects can be done in many different ways. One approach is to study the shape, and in particular the concavities of the object, to try to extract information about how and if the object should be split. In [1] the chain code of the contour of the object is analyzed in the search for endpoints of potential splitting lines. Concavities are thereafter connected pairwise by splitting lines and the grey-level information along each line is used as a parameter in the search for the best splitting line. Another way to split clusters of objects, described in [14], is to analyze distance profiles from the contour of the object to a bounding box. Using a weighted sum of different concavity features, a set of possible splitting lines is built up.

The algorithm described in this paper also uses the shape of the clusters when locating potential splitting lines. Concavities are found using the convex hull of the binary image of the segmented objects from the previous step in the algorithm. The convex hull can be thought of as putting a rubber band around the object making it convex. See Fig. 7(b), where the object together with the shaded areas make up the convex hull. We have used a discrete approximation of the convex hull. Using a 5×5 neighborhood the concavities are filled giving an approximation to the convex hull consisting of polygons having at most 16 sides [2]. Subtracting the original object from the convex hull gives the convex deficiency (concavity regions) of the cluster. The deepest points of every concavity are used as endpoints for splitting lines. The deepest point in a concavity is defined as the point which is inside the convex hull of the object but not inside the object itself, and lying as far from the border of the convex hull as possible, see Fig. 7(c). Having a threshold for this depth gives a way to control how deep into the concavity a point must lie to serve as an endpoint. Should the object contain less than two concavities the object is not subject for any further splitting analysis. For every object a set of endpoints is obtained. Points from different concavities are then combined forming potential splitting lines. For every line the statistical quality score for the two new objects are calculated and compared with the quality score of the object before the split. A criteria for a line to be a candidate for a splitting line is that the line results in more cell-like objects, i.e., a better quality score (as described above).

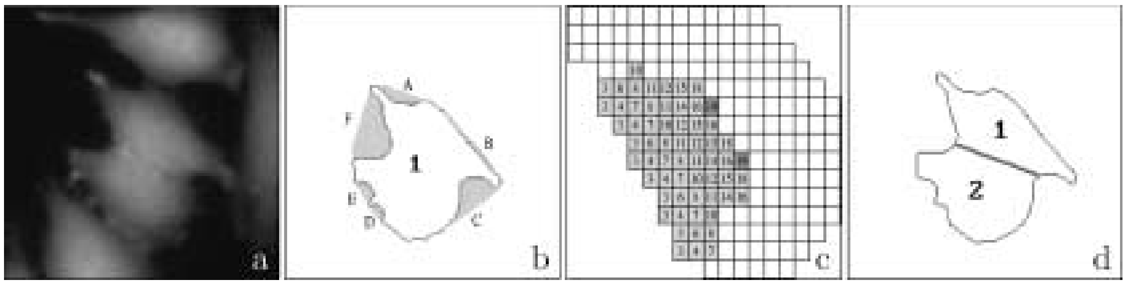


Fig. 7. The different steps in finding splitting lines. Original grey-scale image (a). The segmentation result where two cells have incorrectly been segmented as a single object. The convex deficiency of the object is shown as the shadowed areas around the borders of the object (b). This object contained 6 concavities labeled A to F. A close up of the concavity E is shown in (c). The numbers represent chamfer (3,4) distances to the border. Concavities C and F contain the final endpoints used for splitting the object. The splitting line has its endpoints where the concavities have their maximum distance value, i.e., the deepest point (d).

Using the mean quality score of the two new objects is no guarantee of improvement, since one object might get a very good score and the other a worse score while the mean is still better than the original. A better approach is to use the worst quality score of the two objects. A line resulting in two objects where the worst quality score of the two is better than that of the original object, is a true candidate for a dividing line and is added to a set of possible splitting lines. The line candidate having the best quality score is used as the final splitting line.

2.7. *Final quality measure*

To assess the quality of the final result, a quality measure of each of the segmented cells is reported along with the segmented image. By classifying the final segmentation results as cell-like or not cell-like using the classifier described in Section 2.4, it is possible to continue the analysis of cell specific features using only those cells that are likely to be correctly segmented, i.e., if the application allows for it.

2.8. *Methodology*

The image segmentation algorithm as a whole needs not only an image as input, but also a number of other parameters such as thresholds for minimum object integrated pixel intensity and distance between upper and lower thresholds for the initial segmentation step. If the imaging conditions do not change much between images of the same type, it is sufficient to tune the parameters once for a training image, and then use the same set of parameters for all the remaining images in a fully automatic manner. Features of manually segmented cells in the training image are used for calculation of the mean vector and the covariance matrix needed for the quality measure. To create a good classifier, it is necessary to select a training image showing a representative subset of the cell population.

If several types of cells (with different stain intensity, growth, etc.) exist in the image, it may be necessary to have more than just two classes, i.e., cell/not cell, for the method to work. All objects that do not fit into any of the different classes of cell types would then be sent to the splitting step. The algorithm as a whole assumes that the training image set is representative for the images to segment. If this condition is not fulfilled, it should be apparent in the over all quality measure, which is an indicator of the performance of the algorithm.

3. Results

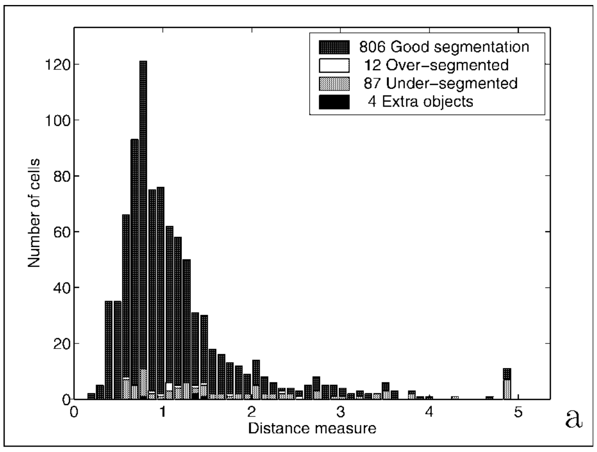
We have tested the algorithms on two sets of images. The images show CHO-cells cultivated for 24 hours and stained with calcein dietoxy methyl ester, an indicator of living cells. The stained cells were detected by fluorescence microscopy and a CCD-camera equipped with a narrow green filter. A 10× objective and a 20× objective was used for the first and second set of images, respectively. One image at each magnification was used as a training image for parameter optimization. These parameters were then used in the segmentation of the similar test images. All objects cut by the image border were removed from analysis to simplify matters.

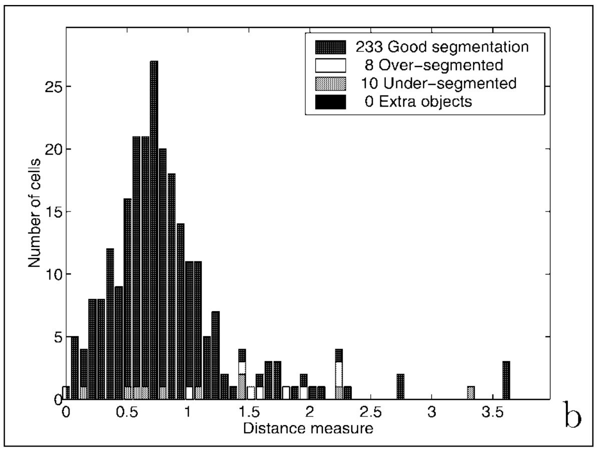
For each type of image (i.e., magnification and type of cells and stain, etc.), a set of features were automatically selected from a set of 19 linearly independent features (described in the Appendix), using the method described in Section 2.4. By inspection of the performance on the corresponding test set, four features were found to be a good tradeoff between performance and generalization in all of the investigated cases. These four selected features were used to calculate a quality measure based on statistical distance. The four features selected by the algorithm showed to be dependent on the training image used, see Table 1, and of course the individual weighting of each of the features into the distance measure is dependent on the training set.

The result of the segmentation was evaluated by automatic comparison with a manual segmentation of the same image. The center of each manually segmented object was found, and the label at the corresponding

Table 1
Features selected for different training sets (ordered after importance)

Set 1 10×	Set 2 10×	Set 3 20×	Set 4 20×
2nd transversal moment	Integrated intensity	Convex perimeter	Edge intensity range
Edge intensity range	Mass displacement	Edge intensity range	Compactness index
Integrated intensity	2nd transversal moment	Max length	Mean intensity
Intensity range	Intensity range	Mass displacement	2nd transversal moment





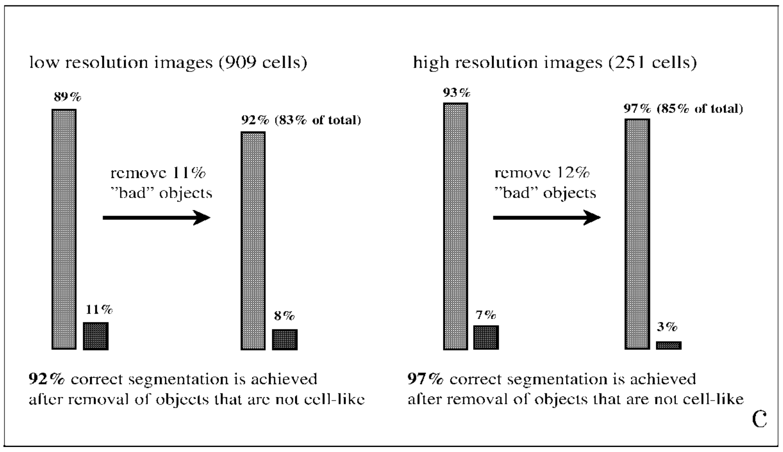


Fig. 8. Histogram of the statistical distance, i.e., inverse of the quality score, of the segmented cells in the low (a) and high (b) resolution images along with a bar plot of the final result of the segmentation algorithm (c).

position in the automatically segmented image was put in a table. If the same label appears more than once in the table, the corresponding objects are under-segmented. In the same way, the center of each automatically segmented object was found, and the label at the corresponding position in the manually segmented image was put in a table. If the same label appears more than once in this table, the corresponding object is over-segmented. If no label at all is found, we have found an extra object, i.e., noise or debris. This method of evaluating the segmentation result is fast and efficient, but does not say anything about how well the outline of the objects overlap.

Success rates for the segmentation procedure is of course very much dependent on the specific imaging situation, but still it gives an indication on the possibilities of the method. For the low resolution images (10$\times$ objective), 89% of a total of 909 cells were correctly segmented, as seen in Fig. 8(a and c). However, if the application allows for rejection of some cells, removing the least cell-like objects can improve the result. E.g., rejecting the 11% of the cells in the whole population which are the least cell-like according to the quality measure, resulted in 92% of the remaining cells being correctly segmented. For the high resolution images (20$\times$ objective), 93% of a total of 251 cells were correctly segmented before the splitting algorithm was applied. The splitting resulted in over-segmentation of many of the previously correctly segmented cells, and was therefore not performed on the high resolution image. This is a clear indication that the described splitting algorithm is not the most robust, especially when dealing with the not so smooth cell borders of the high resolution images. Intelligent smoothing of the high resolution images, and/or further tuning of the algorithm may improve this situation. This has not been evaluated in this paper. Rejecting the 12% least cell-like objects of the higher resolution image resulted in as much as 97% correctly segmented cells in the remaining population. See Fig. 8(b and c).

4. Discussion and comments

The methodology applied throughout the paper does not impose any restrictions on the classifier used, other than that it should give a single scalar output for each of the objects analyzed. It would be interesting to try other, more advanced classifiers than just a statistical distance measure to see if any improvement can be found, for example, by using statistical techniques that take the non-normal distributions of some of the features into account. This, together with additional features that better describe what is cell-like, would give a quality measurement that better separates the incorrectly segmented cells from the correct ones.

The described segmentation algorithm rests on the assumption that all cells are similar, or belong to one out of a limited number of classes of similar cells. This may appear as a problem when the goal of the analysis is to find variations in staining patterns, structures, intensities, etc. However, if it is possible to stain for several antigens in parallel, segmentation can be made in the image of the stain showing the general cytoplasm, while a different stain is used for the structure of interest (but not general shape). For example, a regular cytoplasm stain can be imaged using one fluorescent dye and corresponding filters for the camera, and the staining of interest can be imaged simultaneously using a different dye and filter set. The segmentation procedure is then run on the image of the cytoplasm stain, and the segmentation result is applied to the image of the stain of interest and used as a template for feature extraction (see, e.g., [21]).

The cells that we try to segment are only stained with a cytoplasmic stain, i.e., no nuclear counter-stain is available. If a nuclear counter-stain is available, the nuclei can be segmented first, e.g., by methods described in [12,16], and then be used as seeds in the segmentation of the cytoplasm. This significantly simplifies the detection of individual cytoplasms. However, an additional stain means additional impact on the cells, as well as additional costs and time. There is also a limit to the number of stains that can be used in parallel, i.e., adding a separate nuclear stain reduces the number of stains that can be used for response studies.

Performance of the splitting algorithm may be improved by using splines or some other curve drawing method instead of always applying straight splitting lines. The intensity distribution across the clustered cells could be used to allow splitting of objects not having concavities. At present, the splitting algorithm results in over-segmentation of high resolution images. This can partly be explained by the fact that the number of cells to train the algorithm on was smaller than for the low resolution images.

Another reason for failure of the segmentation is due to saturation of the images, i.e., the upper limit of the camera well depth is reached, and information on grey-value variation is lost. This can be avoided by simply adjusting the exposure time.

5. Conclusions

It is possible to achieve 89–97% correct segmentation accuracy for this fully automatic method for cytoplasm segmentation of fluorescence labeled cells. By including statistical analysis and a feedback system, segmentation errors can be located and in many cases corrected. This has been shown to give a significant improvement to the overall performance of the segmentation algorithm.

Acknowledgements

This project was sponsored by Amersham Pharmacia Biotech, Uppsala, Sweden and the Swedish Foundation for Strategic Research (SSF) through the VISIT (VISual Information Technology) program.

Appendix

The set of 19 features that were used as a basis for the stepwise feature selection procedure. Linearly dependent features, such as Convex deficiency = Convex area − Area, are modeled internally by the Mahalanobis distance and do not contribute with any further information. Additionally, the inclusion of linearly dependent features results in a singular matrix, leading to numerical instability of the Mahalanobis distance.

1. Area = The number of pixels belonging to the object, provides a measure of the object size.
2. Perimeter = The sum of steps taken when walking around the edge pixels of the object. Horizontal and vertical steps are weighted by $a = 0.948$ and diagonal steps are weighted by $b = 1.343$ [3,7,9].

3. Compactness index = Perimeter^2/($4\pi *$Area). A measure of compactness.
4. Convex area = The area of the convex hull of the object. The convex hull is the smallest polyhedron that will cover the object [2].
5. Convex perimeter = The perimeter of the convex hull.
6. Relative convex area = Convex area/Area.
7. Relative convex perimeter = Convex perimeter/Perimeter.
8. Length = The longer side of smallest circumscribed rectangle.
9. Width = The shorter side of smallest circumscribed rectangle.
10. Elongatedness = Length/Width.
11. 2nd longitudinal moment = The variance of pixel position along the main axis.
12. 2nd transversal moment = The variance of pixel position orthogonal to the main axis.
13. Integrated intensity = The sum of the grey level intensities of all pixels belonging to the object.
14. Mean object intensity = Average pixel value.
15. Standard deviation of the object intensity.
16. Intensity range = Maximum − minimum object intensity.
17. Edge intensity range = Maximum − minimum intensity along the border of the object.
18. Normalized edge intensity range = Edge intensity range/Intensity range.
19. Mass displacement = Distance between the center of mass given by the grey-level image of the object and the center of mass given by a binary mask of the object.

References

[1] E. Bengtsson, O. Eriksson, J. Holmquist, T. Jarkrans, B. Nordin and B. Stenkvist, Segmentation of cervical cells: Detection of overlapping cell nuclei, *Computer Graphics and Image Processing* **16** (1981), 392–394.

[2] G. Borgefors and G. Sanniti di Baja, Analyzing nonconvex 2D and 3D patterns, *Computer Vision and Image Understanding* **63** (1996), 145–157.

[3] L. Dorst and A.W.M. Smeulders, Length estimators for digitized contours, *Computer Vision, Graphics and Image Processing* **40** (1987), 311–333.

[4] S. Gilles, M. Brady, J. Declerck, J. Thirion and N. Ayache, Bias field correction of breast MR images, in: *Proceedings of the Fourth International Conference on Visualization in Biomedical Computing (VBC)*, Springer, Hamburg, Germany, 1996, pp. 153–158.

[5] R.W. Hamming, *Numerical Methods for Scientists and Engineers*, 2nd edn, McGraw-Hill, 1973.

[6] R.A. Johnson and D.W. Wichern, *Applied Multivariate Statistical Analysis*, 4th edn, Prentice-Hall, 1998.

[7] Z. Kulpa, Area and perimeter measurement of blobs in discrete binary pictures, *Computer Graphics and Image Processing* **6** (1977), 434–454.

[8] P. Lancaster and K. Šalkauskas, *Curve and Surface Fitting, an Introduction*, Academic Press, London, 1986.

[9] J. Lindblad, Perimeter and area estimates for digitized objects, in: *Proceedings of the SSAB (Swedish Society for Automated Image Analysis) Symposium on Image Analysis*, Norrköping, Sweden, March 2001, pp. 113–117.

[10] J. Lindblad and E. Bengtsson, A comparison of methods for estimation of intensity nonuniformities in 2D and 3D microscope images of fluorescence stained cells, in: *Proceedings of the 12th Scandinavian Conference on Image Analysis (SCIA)*, Bergen, Norway, June 2001, pp. 264–271.

[11] J. Lindblad, C. Wählby, M. Vondrus, E. Bengtsson and L. Björkesten, Statistical quality control for segmentation of fluorescence labelled cells, in: *Proceedings of the 5th Korea–Germany Joint Workshop on Advanced Medical Image Processing*, Seoul, Korea, May 2001.

[12] N. Malpica, C.O. de Solorzano, J.J. Vaquero, A. Santos, I. Vallcorba, J.M. Garcia-Sagredo and F. del Pozo, Applying watershed algorithms to the segmentation of clustered nuclei, *Cytometry* **28**(4) (1997), 289–297.

[13] H. Netten, L.J. van Vliet, H. Vrolijk, W.C.R. Sloos and I.T. Young, Fluorescent dot counting in interphase cell nuclei, *Bioimaging* **4** (1996), 93–106.

[14] U. Pal, K. Rodenacker and B.B. Chaudhuri, Automatic cell segmentation in cyto- and histometry using dominant contour feature points, *Analyt. Cell. Pathol.* **17** (1998), 243–250.

[15] P. Ranefall, B. Nordin, L. Egevad and E. Bengtsson, A new method for segmentation of color images applied to immunohistochemically stained cell nuclei, *Analyt. Cell. Pathol.* **15** (1997), 145–156.

[16] P. Ranefall, K. Wester and E. Bengtsson, Automatic quantification of immunohistochemically stained cell nuclei using unsupervised image analysis, *Analyt. Cell. Pathol.* **16** (1998), 29–43.

[17] C.O. de Solorzano, R. Malladi, S.A. Lelievre and S.J. Lockett, Segmentation of nuclei and cells using membrane related protein markers, *Journal of Microscopy* **201**(3) (2001), 404–415.

[18] M. Sonka and J.M. Fitzpatrick, *Handbook of Medical Imaging*, Vol. 2, SPIE Press, Washington, 2000.

[19] J.A. Swets, R.M. Dawes and J. Monahan, Better decisions through science, *Scientific American* **283**(4) (2000), 70–75.

[20] L. Vincent and P. Soille, Watersheds in digital spaces: An efficient algorithm based on immersion simulations, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(6) (1991), 583–597.

[21] C. Wählby, F. Erlandsson, E. Bengtsson and A. Zetterberg, Sequential immunofluorescence staining and image analysis for detection of large numbers of antigens in individual cell nuclei, *Cytometry* **47**(1) (2002), 32–41.