Image analysis algorithms for cell contour recognition in budding yeast

Autor Kvarnstrom M.
http://www.opticsinfobase.org/□□

Abstract: Quantification of protein abundance and subcellular localization dynamics from fluorescence microscopy images is of high contemporary interest in cell and molecular biology. For large-scale studies of cell populations and for time-lapse studies, such quantitative analysis can not be performed effectively without some kind of automated image analysis tool. Here, we present fast algorithms for automatic cell contour recognition in bright field images, optimized to the model organism budding yeast (*Saccharomyces cerevisiae*). The cell contours can be used to effectively quantify cell morphology parameters as well as protein abundance and subcellular localization from overlaid fluorescence data.

1. Introduction

Fluorescence microscopy is an exceptionally powerful technique to study protein processes *in vivo*. However, due to the qualitative and subjective nature of human interpretation of images, quantitative data analysis is not straightforward. In addition, for statistical or high throughput measurements, manual analysis becomes highly time consuming. Automated image analysis opens up for advanced studies in the diversity of cell populations, i.e. characterization of phenotype heterogeneity, or single cell studies *in vivo*. This type of studies have recently attracted significant attention, and have provided key insights in for example, gene expression noise and cell cycle regulation.

In this report, novel algorithms designed for automatic analysis of bright field and fluorescence images of budding yeast (*Saccharomyces cerevisiae*) are presented. Budding yeast is one of the most important model organisms for cell and molecular biology, with advantages of being easy to handle and modify genetically. In this context, it also benefits from the library of chromosomally tagged Green Fluorescence Protein (GFP) fusion proteins that became publicly available in 2003. The main principle behind our approach is to use bright field images for cell recognition and then transfer the extracted contours to the corresponding fluorescence images for protein analysis. Hence, our method does not rely on staining of the cell membrane and it is therefore suitable for *in vivo* studies.

Automatic identification and recognition of cells in microscopy images has been reported before, but rather few authors have addressed the entire process from image to determination of precise cell contours of individual cells. Chen *et al.* used a machine learning approach to automatically find approximate cell boundaries in a graphical model representation. de Carvalho *et al.* used the watershed transform applied on a topology based on a combination of gray-scale images and various hierarchical and geometric properties of the cells. Niemist¨o also used a watershed approach for separating yeast cells prior to localization of small fluorescent organelles. Rue and Husby successfully used an advanced deformable template model

for the near-circular cells of interest in order to identify fluorescently labeled contours of cells even when the borders are in part not visible. The drawback with their scheme is its huge computational complexity and the computations had to be carried out using a Markov chain Monte Carlo (MCMC) algorithm. Mardia *et al.* used a similar approach to detect partially occluded objects. Software for automatic analysis of fluorescence microscopy images of budding yeast has been presented by other groups. Miroshita and co-workers developed an image analysis program for cell morphology characterization of gene-deletion strains. However, their algorithm requires the cell membrane to be stained with a fluorescent dye. The open-source software Cell-ID, on the other hand, uses bright field images for cell recognition. The actual image analysis methods used have not yet been published.

We here present algorithms for the entire process of image analysis, from thresholding and segmentation of images, to refined precise single cell contours. The main novel contribution in this work is the method for refined cell contour extraction.We utilize a shape model similar to the deformable template model used by Rue and Husby, with constraints between three consecutive points on the cell contour. Our model, on the other hand, enables computations of much lower computational complexity and, hence, higher speed. Figure 1 illustrates the main steps of our method. The initial steps involve segmentation using a new adaptive thresholding technique, for finding suitable candidate cell centers, see Figs. 1(a)-(c). The final step is the cell contour extraction, where a more refined cell contour is searched for. The final resulting contours are shown in Fig. 1(d).

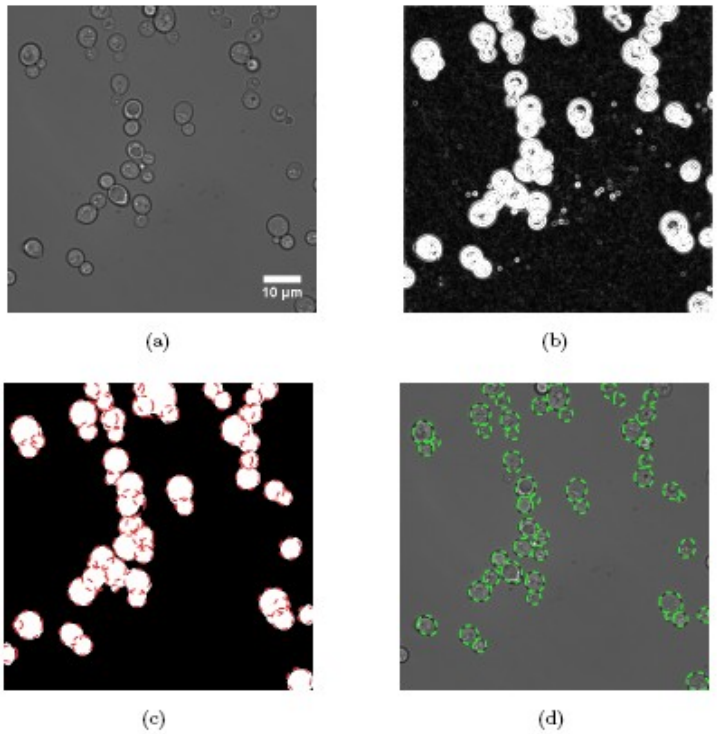


Fig. 1. Illustration of the main steps of the cell recognition method: (a) the original bright field image showing a population of budding yeast cells; (b) the gradient, or edge map,image, where bright pixels represent a large gradient magnitude; (c) the segmented image with identified circles in

red. The centers of these circles will serve as candidate cell centers; (d) the result after cell contour extraction.

The cell contour extraction algorithm computes the optimal solution in polynomial time, using a dynamic programming scheme. The high speed of the algorithm makes it suitable for studies where large data sets are involved, such as time-lapse or high-throughput studies of cell populations. Our method is shown to be fast and, for the kind of images considered, we found that 96% of the cells were satisfactorily recognized. The algorithms are also found to be robust and adaptive, in the sense that they work without human intervention and are able to identify cells with high precision even in the presence of several neighboring cells, at different illumination levels or if the cells are slightly out of focus.

The paper is organized as follows: first the experimental and biological setup, used for producing the images in this paper, is presented. The image analysis methods are described in the subsequent section which is the main part and focus of the paper. We then evaluate our image analysis method, and exemplify its use in a biological application. Details regarding the cell contour algorithm have been moved to the Appendix.

2. Methods

2.1. Image acquisition

Images were collected using an inverted epi-fluorescence microscope (Nikon TE2000E) equipped with a back illuminated, electron multiplying gain CCD camera (Ixon DV887- DCSBV, Andor Technology). Bright field images, used to describe the image analysis method, were recorded using either a 60X objective (Plan Apo, NA=1.40) or a 100X objective (Plan Apo, NA=1.45). Images used in the rest of the paper were recorded with the 60X objective.

For the biological study, images were recorded in bright field and fluorescence mode every 10 minutes for 6 hours. To ensure that all nuclei were recorded, the cells were imaged in three image planes with a separation of approximately 2 μm. A mercury lamp was used for fluorescence excitation and a filter set (excitation: HQ470/40x, emission: HQ515/30m, dichroic: 505DCLP, Chroma Corporation, USA) were used for GFP imaging. The fluorescence images were recorded with an integration time of 500 ms using an excitation intensity of approximately 0.1 W/cm2 at the sample.

2.2. Strains and cultivation

TheMCM4 strain was obtained from the GFP-library. The cells were kept at agar plates with YPD medium containing 1% (w/v) yeast extract (Merck), 2% (w/v) Bacto-peptone (Merck), and 2% (w/v) glucose (Merck). For microscopy studies, cells were taken from the plate and placed in YNB medium that was composed of 0.17%

(w/v) yeast nitrogen base without amino acids (Difco), 0.5% (w/v) ammonium sulfate (Merck), 2% (w/v) glucose and aminoacid supplement lacking histidine (BIO101 Inc.). The pH of all media was adjusted to 6.0. The cells were pre-grown using an orbital shaker at 30◦C. At early log phase, 10 µl of the suspension was placed between two cover slides separated by the secure seal spacer S24737 from Molecular Probes, creating an enclosed well. The cells were kept at approximately 20 ◦C during the experiments.

3. Cell contour recognition algorithm

3.1. Adaptive thresholding and segmentation

The segmented image is computed via thresholding on the gradient image *I mag* , which is computed from the original bright field image *I*. The reasons for using *I mag* are twofold. First of all, the distribution of gradient pixel values is uni-modal, which makes it easier to find a suitable threshold, see Fig. 2.

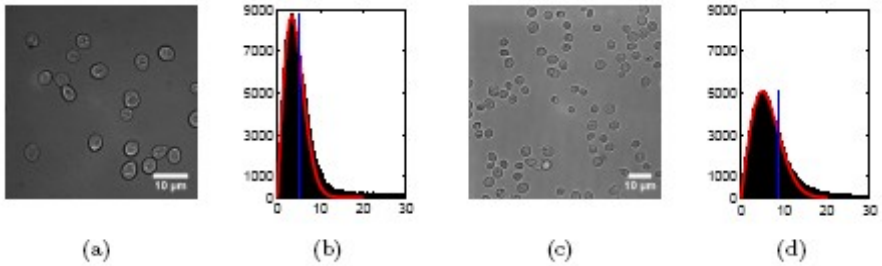


Fig. 2. Illustration of how differences in illumination level and cell densities in the bright field images, (a) and (c), affect the gradient image histograms, shown in (b) and (d). The red curves represent the corresponding Rayleigh distributions that have been fitted to values in the histogram that lay below the median value, here displayed as blue vertical lines. The horizontal axis has been truncated at 30 for ease of display.

Operating on *Imag* also has the benefit of making the segmentation invariant to differences in the overall illumination level of the image. In practice, this invariance also holds for a slowly varying background illumination level. Regions in the image where cell contours are located typically exhibit a large gradient magnitude. The opposite is true for background regions. This relationship between *I* and *I mag* can be seen by comparing Figs. 1(a) and (b).

The gradient image is computed using Prewitt's method. The original bright field image *I* is filtered with two masks, one for each direction, where each mask enhances differences in the corresponding direction. This results in two new images, *I x* and *Iy*, each being a measure of the pixelwise variation in the *x* and *y* directions, respectively. After this, we let $I_{mag} = (I_x^2 + I_y^2)^{1/2}$ for each pixel location. Pixels with gradient values above a certain threshold β will now be assigned as foreground regions. In Fig. 1(c) we show the thresholded gradient image after filling of holes and running a morphological opening.

The crucial part in the segmentation step is to automatically find the threshold β. It is particularly important to adapt to the illumination level and the density of cells in the image. Depending on the experimental setup, the preferred threshold level may vary substantially. Adaptive thresholding techniques found in the literature, e.g. Otsu's nonparametric method or the parametric method of fitting the histogram to a mixture of Gaussians, typically rely on bimodality of the histogram or that pixel values have a specific distribution. From the histograms in Fig. 2, we see that bi-modality is not the case here. Furthermore, the distribution for large gradient pixel values is hard to categorize. Although not displayed in the figure, the apparent uniformity for large values continuous up to values above 200 where it finally starts to decline.

The maximum value of these particular histograms are 588 and 770. The method we present here does not make any assumptions on the distribution of gradient pixel values of large magnitude, the only assumption made is that the distribution of noise in background regions are approximately normal distributed.

Gradient values below the median, shown as vertical lines in Figs. 2(b) and (d), are fitted to a Rayleigh distribution function,

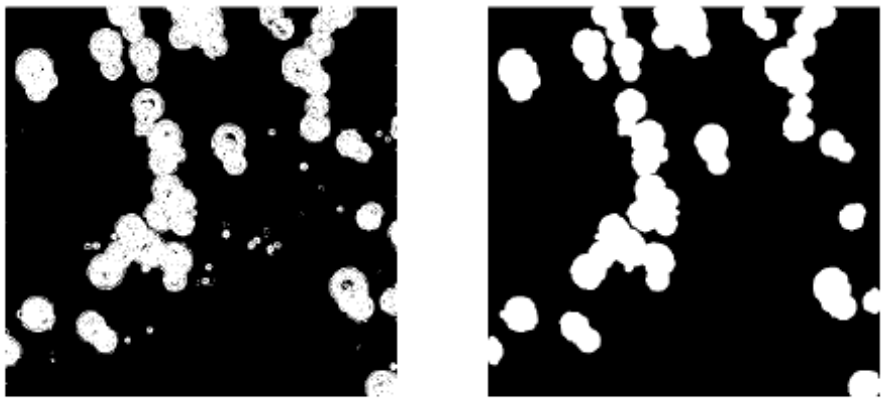
$$f_R(x) = \frac{x}{\sigma^2}\exp\left\{-\frac{x^2}{2\sigma^2}\right\}, \quad (1)$$

where σ is the parameter that is varied until the best fit is found and x is the pixel value. The rationale behind this procedure is that gradient pixel values of $I\,mag$ will be Rayleigh distributed under the assumptions that the values of Ix and Iy are identical, independent, and normally (Gaussian) distributed. Even though this is a crude assumption, we see from the fitted curves in Figs. 2(b) and (d), that Eq. (1) gives an excellent description of the gradient value distribution below the median.

The parameter σ constitutes a good measure of the pixel intensity variation for low gradient pixel values, i.e. for pixels in the background regions. Therefore, pixels which have gradient values far larger than σ should be rejected as background and instead be considered as foreground.

Here we let the threshold relate to σ through $\beta = 7.5\sigma$. This corresponds to declaring pixels with gradient values more than 6 standard deviations larger than the mean of the estimated distribution of background pixels as foreground pixels. It should be mentioned that it is not necessary to specifically use the median as the delimiter for fitting the Rayleigh distribution.

To obtain the final segmented image, the holes (black regions encircled by white regions) in the thresholded image are filled. Furthermore, in order to remove small structures due to noise, we apply a morphological opening with a circular structure element of radius 4 pixels. In Fig. 3, these two steps are illustrated for the image in Fig. 1.



(a) (b)

Fig. 3. Final steps of the segmentation; (a) thresholded gradient image, where white pixels are above β; (b) the final segmented image after filling of holes and after applying a morphological opening to the image in (a).

3.2. Finding candidate cell centers

Since each connected component in the segmented image represents a possible cluster of cells, these clusters need to be disconnected into individual cells. This is accomplished by searching for circle segments along the outer contours of objects in the segmented image, using a variation of the circular Hough transform. The principle is illustrated in Fig. 4.

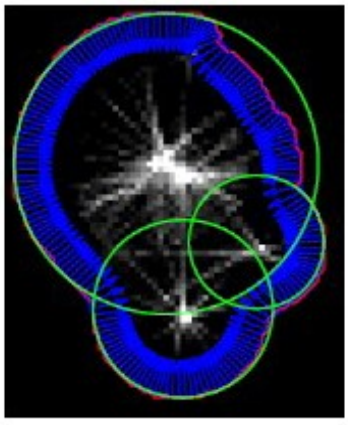


Fig. 4. Illustration of the method for finding candidate cell centers. The outer boundary of the cell cluster and the approximate normals are shown in red and blue, respectively. The accumulation matrix is displayed as a grey scale image in the background. The identified circles are displayed in green.

While traversing the outer boundary of a connected region in the segmented image, an approximate normal to the boundary is computed at each point. Along each normal, we step-wise go inwards and register the distances to the current boundary point. In this way, an accumulation matrix is formed where each element holds the number of times it has been ”hit” by a normal from the boundary and keeps track of the corresponding distances. When each point along a cell cluster boundary has been

visited, we look for local maxima in this accumulation matrix. The resulting circles in Fig. 1(c) are displayed in green. The centers of the circles will then act as candidate cell centers for the final contour extraction.

A common approach for finding suitable object centers from segmented images in similar applications is to look for local maxima in the distance transform of the segmented image. However, the distance transform only works when cells are sparsely clustered, up to approximately three or four cells in each cluster. Although somewhat more computationally intensive, the method described above works for heavier clustering and is hence more generally applicable. Nevertheless, both approaches depend on that all cells must have at least a part of the boundary exposed to the background region in the segmented image.

3.3. Cell contour extraction using dynamic programming

Reasonable yeast cell contours are fairly smooth and, at least to a certain degree, convex in shape. In order to find the ”most suitable” cell contour, such shape constraints should be balanced with the data in the image, for example gradients and directional derivatives. In the literature, a shape model that is adapted according to surrounding edges is often referred to as a deformable template, see e.g. [18, 11, 10]. The reason behind using a shape model is that it will guide the solution to pick the edges which correspond best to the model, when image information is either missing or when it contains inconsistent and ambiguous edges. Examples are loss of contrast at a specific part of a cell boundary or nearby occluding cells. Concerning the image information part of the cell extraction, it turns out that the dark diffraction fringe that is evident around yeast cells imaged in bright field, is located at approximately the same distance to the true cell contour, irrespective of the distance from the focal plane. In this paper we utilize this feature to locate cell contours, but this can easily be modified depending on the application.

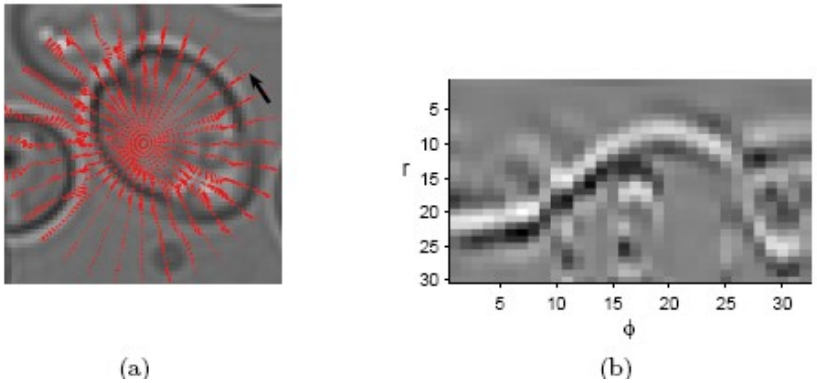
In principle, each closed sequence of connected pixels surrounding a candidate cell center is a candidate for a cell contour, and therefore the space of possible contours is immense. In order to regularize the problem, we introduce a reference system in the form of a polar plot relative to each candidate center. Figure 5(a) displays the directional derivatives $I\,x$ and Iy at $M = 32$ rays emanating from a candidate cell center, where each ray is sampled at $N = 30$ equally spaced radial points. Here we let suitable contour points to be those that are located in between positions with large derivatives directed in opposite directions. This conforms with the notion that the true cell boundary lies at the dark fringe around the cell. Formally, $f(r,\varphi)$ denotes the contour point criterion function for a point at distance r and angle φ to be a contour point. The angle φ is measured in the counter-clockwise direction, starting at the black arrow in Fig. 5(a).

Let

$$f(r_i,\phi_m) = \left(I_x(r_{i+1},\phi_m) - I_x(r_{i-1},\phi_m)\right)\cos\phi_m + \left(I_y(r_{i+1},\phi_m) - I_y(r_{i-1},\phi_m)\right)\sin\phi_m \qquad (2)$$

for $I \in \{1,2, \ldots ,N\}$ and $m \in \{1,2, \ldots ,M\}$. $Ix\,(r,\varphi\,)$ and $Iy\,(r,\varphi\,)$ are interpolated values (where needed) at distance r and angle φ from the candidate cell center in the derivative images $I\,x$ and Iy , respectively. The cosine and sine operations correspond to a projection along the current ray.

This will result in less impact of edges that are not oriented in the same direction as the ray. In Fig. 5(b), the criterion function $f\,(r,\varphi\,)$ is displayed for the candidate center in Fig. 5(a).



(a) (b)

Fig. 5. (a) Directional derivatives along M = 32 rays, each sampled at N = 30 radial distances emanating from a candidate cell center; (b) Polar plot of the criterion function f in Eq. (2) at the points along the rays in (a). The angles φ are measured in the counterclockwise direction, starting from the black arrow in (a). The rows in (b) correspond to the radial distances r in (a) and the columns in (b) correspond to the the angles of the rays φ . Bright pixels in (b) represent good contour points as measured by the criterion function.

Bright pixels in Fig. 5(b) represent high values of f and hence ”good” positions. The idea is now to pick a path from left to right in this polar plot, visiting each column once. To find the optimal path, we use dynamic programming. Note that simply taking the maximum point r for each ray in direction φ will in general not produce a continuous and closed contour.

When picking a path from left to right in the polar plot, restrictions on radial transitions will affect the variety of shapes of the final contour. In other words, the restrictions will define our deformable template model. When using dynamic programming, it is only possible to put local restrictions on these transitions; they cannot for example depend on the entire path up to the present angle φ . Global shape constraints, such as convexity, are therefore not possible to enforce directly. It is not even possible to guarantee that the final contour will be closed, i.e. that it starts and ends in the same radial position. In the special cases of enforcing convexity and closedness, there are however methods to circumvent this problem.

Here we use a method that penalizes transitions in the polar plot that correspond to right turns in the original image coordinates as we encircle the candidate center in a counter-clockwise direction. If we in addition can guarantee that the resulting contour will be closed, this local convexity condition will produce a globally convex contour, i.e. the contour of a shape that is geometrically convex. Closedness can in turn be enforced implicitly by running the algorithm separately for

each of the *N* possible starting points. Nevertheless, we use a heuristic algorithm that runs the dynamic programming scheme for three consecutive laps and uses the middle part of the final path as the solution. Details regarding the dynamic programming scheme and the transition conditions are presented in the Appendix.

In Fig. 6, we demonstrate the contour extraction method. In the upper example, we used a simple transition rule that only allows straight or diagonal transitions between consecutive columns in the polar plot. In the lower example, we used a condition that penalizes transitions that corresponds to making right-turns, also called the local convexity condition. With the latter condition, the algorithm manages to retrieve the contour even though the candidate center is heavily off-centered. This is the typical behavior of the algorithm. By using the local convexity rule, the deformable template is flexible enough to handle heavily off-centered candidate cell centers while still being able to enforce the convexity of the resulting contour. Hence, besides being more theoretically appealing than the transition rule exemplified in the upper example of Fig. 6 which is commonly used in contour extraction using dynamic programming, the local convexity rule also has increased robustness towards poor positioning of candidate cell centers. In both examples, we used the heuristic three laps-methods for obtaining closed contours, where the middle part (between the two vertical red lines) is used as optimal path and displayed in the corresponding images on the right. Note that only using one lap in the polar plot of Fig. 6, the upper example would not have resulted in a closed contour, since the optimal path deviates upwards from the correct one as it approaches the boundary on the right.

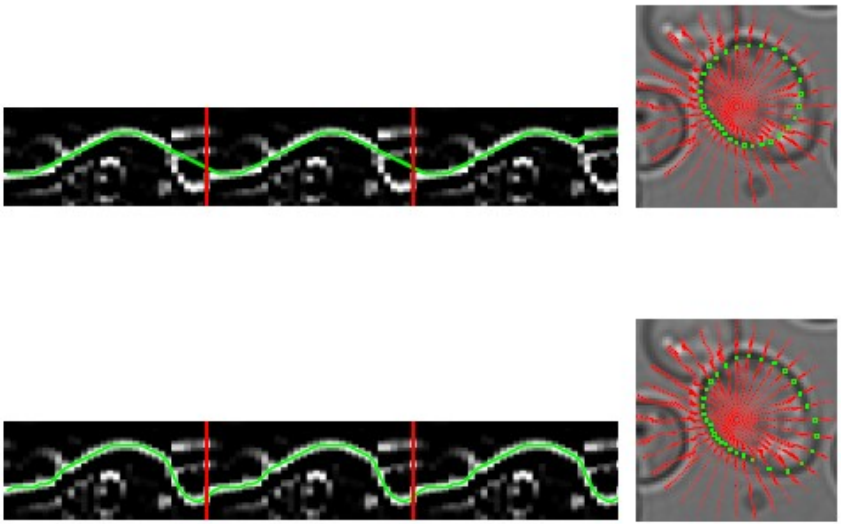


Fig. 6. Illustration of the refined contour extraction algorithm using two different transition rules. On the left are two optimal paths in three consecutive copies of the polar plot. The upper transition rule only allows for straight or diagonal transitions in the polar plot, whereas the transition rule in the lower example is the local convexity condition that penalizes transitions that corresponds to turning right in the image as the candidate center is encircled. The solution to the middle copy of the polar plot is used as resulting contour. The resulting contours to the examples on the left are displayed in the sub-images on the right as green squares.

4. Performance

The cell contour recognition algorithms were run on 1163 cells in 25 independent images, each containing between 24 to 65 cells. Typical images exhibited a relatively low degree of cell clustering and cell density and are well represented by the examples in Figs. 1 and 2. The density and clustering of the images in the study are comparable to the density and clustering in the example images of the references. Visual examination of the defined cell contours showed that 96% were correctly defined, and the algorithms were proven to be robust both in regards to differences in cell sizes as well as the degree of off focus positioning. The remaining 4% of the cells were either missed, incorrectly defined, or were false hits. These classes are exemplified in Figs. 7(a)-(d). Cells are missed mainly as a consequence of being part of a cluster. Even if they are not completely surrounded by cells, the outer border of cells located inside clusters could be too short for a candidate cell center to be defined using the method illustrated in Fig. 4. This is case for the missed cells in Fig. 7(d). Incorrectly defined contours are generally only generated for buds. Here, it could be argued that buds should not be included as a separate cell at all. However, if they should, a more narrow range of possible radial distances in the refined contour extraction step should be used for buds. In either case, a last step could fairly easily be included, discarding unreasonable contours, e.g. those that are too small or have a too large overlap with other cells. In this way, the number of incorrect contours and false hits could be reduced.

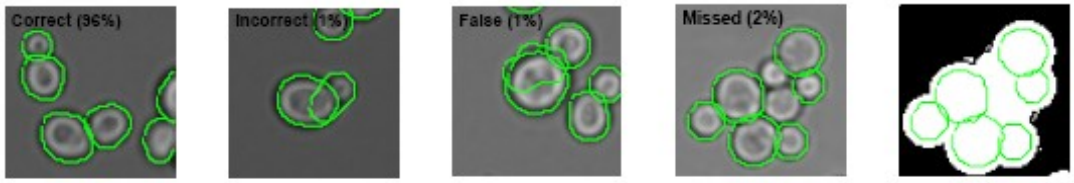


Fig. 7. Performance of cell contour recognition. Figures (a)-(d) illustrate the contour classes used in the success rate estimation. Of >1000 cells analyzed, 96% were defined correctly, 1% incorrectly, 1% were false hits and 2% were missed. The two missed cells in (d) were lost because they only have a minor part of their border free to fit to a circle, as shown in (e).

To estimate the speed of the algorithms, cpu-times were measured with the algorithms running on a PC with an Intel Duo-Core CPU at 1.66GHz with 2GB RAM, in Windows XP and Matlab R2006a. Time-critical algorithms were implemented in C and linked to Matlab using the mex-utility. Cpu-times were measured by alternate use of the ”tic” and ”toc” commands in Matlab. For three example images, containing 49, 96 and 173 cells, the cpu-times for cell recognition from bright field image to refined cell contours were measured to be approximately 1.4, 2.0 and 2.8 seconds, respectively. The convex contour extraction part, using the heuristic three-laps method, takes about 10 ms per candidate center to compute for $M = 32$ and $N = 30$.