Chapter 2
Rapid Prototyping of Image Analysis Applications

Cris L. Luengo Hendriks, Patrik Malm, and Ewert Bengtsson

2.1 Introduction

When developing a program to automate an image analysis task, one does not start with a blank slate. Far from it. Many useful algorithms have been described in the literature, and implemented countless times. When developing an image analysis program, experience points the programmer to one or several of these algorithms. The programmer then needs to try out various possible combinations of algorithms before finding a satisfactory solution. Having to implement these algorithms just to see if they work for this one particular application does not make much sense. This is the reason programmers and researches build up libraries of routines that they have implemented in the past, and draw on these libraries to be able to quickly string together a few algorithms and see how they work on the current application. Several image analysis packages exist, both commercial and free, and they can be used as a basis for building up such a library. None of these packages will contain all the necessary algorithms, but they should provide at least the most basic ones. This chapter introduces you to one such package, DIPimage, and demonstrates how one can proceed to quickly develop a solution to automate a routine medical task. As an illustrative example we use some of the approaches taken over the years to solve the long-standing classical medical image analysis problem of assessing a Pap smear. To make best use of this chapter, you should have MATLAB and DIPimage running on your computer, and try out the command sequences given.

C.L. Luengo Hendriks (✉)
Centre for Image Analysis, Swedish University of Agricultural Sciences,
Box 337, SE-751 05 Uppsala, Sweden
e-mail: cris@cb.uu.se

G. Dougherty (ed.), *Medical Image Processing: Techniques and Applications*, Biological and Medical Physics, Biomedical Engineering, DOI 10.1007/978-1-4419-9779-1_2,

2.2 MATLAB and DIPimage

2.2.1 The Basics

DIPimage is built on MATLAB (The MathWorks, Natick, MA, USA), which provides a powerful and intuitive programming language, publication-quality graphing, and a very extensive set of algorithms and tools. DIPimage adds to this a large collection of image processing and analysis algorithms, easy computation with images, and interactive graphical tools to examine images. It is designed for ease of use, using MATLAB's simple command syntax and several graphical user interfaces, and is accessible to novices but fast and powerful enough for the most advanced research projects. It is available free of charge for academic and other noncommercial purposes from its website: http://www.diplib.org/.

DIPimage extends the MATLAB language with a new data type. Natively, MATLAB knows about arrays of numbers, characters, structures or cells (the latter can contain any other data type). With this toolbox installed, images are added to this list. Even though images can be seen simply as an array of numbers, there are several advantages to this new type: indexing works differently than in an array, the toolbox can alter the way operations are performed depending on the pixel representation, and images can be automatically displayed. This latter point is significant, for it greatly enhances accessibility to novices and significantly increases the interactivity in the design phase of an image analysis application.

In MATLAB, assigning the value `1` to a variable `a` is accomplished with:

```
a = 1;
```

Additionally, if the semicolon is left off this statement, MATLAB will reply by displaying the new value of the variable:

```
a = 1
a =
     1
```

Similarly, when leaving the semicolon off a DIPimage statement that assigns an image into a variable, MATLAB will reply by displaying that image in a figure window. For example, the next statement reads in the Pap smear image in file "papsmear.tif"[1] and assigns it to variable `a`.

```
a = readim('papsmear.tif')
Displayed in figure 10
```

Depending on the chosen configuration, the image will be displayed to a new window or an existing window. To suppress automatic display, all that is thus needed is to add a semicolon at the end of all statements.

[1] You can obtain this file from http://www.cb.uu.se/~cris/Images/papsmear.tif

DIPimage features a graphical user interface (GUI) that gives access to the most commonly used functions in the toolbox (if the GUI does not appear by default in your installation of DIPimage, run the command `dipimage` in the MATLAB command window). The GUI's menus list all available functions. Selecting one of these functions changes the area below the menus to allow parameter selection and execution of the function. For example, the function used above, `readim`, is available under the "File" menu. Selecting it brings up a control to select the file to read and choose a name for the variable that will hold the image. Pressing the "Execute" button will read the selected file and put its contents into the chosen variable. Additionally, the result of the command is displayed.

2.2.2 Interactive Examination of an Image

The figure windows in which the images are automatically displayed have four menus. The second and third ones allow the user to change the way the image is displayed. Note that some menu options are only present when applicable. For example, the "Mappings" menu has options to choose the slicing direction in 3D and 4D images, which are not visible with 1D or 2D images; two- or higher-dimensional, gray-value images have options to select a color map, which are hidden for color images. The fourth menu, "Actions," contains all the interactive tools that a user can use to examine the image in the display. The "Action" enabled by default is "Pixel testing," which allows the user to hold the left mouse button down to get a reading of the values of the pixel under the cursor. The title bar of the figure window shows the coordinates and either the gray value or the RGB values of the pixel. Holding down the right mouse button allows the user to measure distances. The "Zoom" and "Pan" modes allow closer examination of large images. For 3D and 4D images, several additional options exist. "Step through slices" is to use the mouse to change the slice of the image shown (it is also possible to do this with the keyboard, without changing the mode). Most interestingly, "Link displays" can be used to link various windows displaying 3D or 4D images. These windows will then all show the same slice at all times. This is very useful when, for example, comparing the output of various filters. We encourage the reader to explore these options and read more about them in the user manual [1].

2.2.3 Filtering and Measuring

A large selection of filters and analysis tools are available through the DIPimage GUI. Many more are accessible only from the command line, and are consequently hidden. Typing

```
help dipimage
```

gives an overview of (almost) all functions in DIPimage. For more information on any one function, use the `help` command with the function's name. We will stick

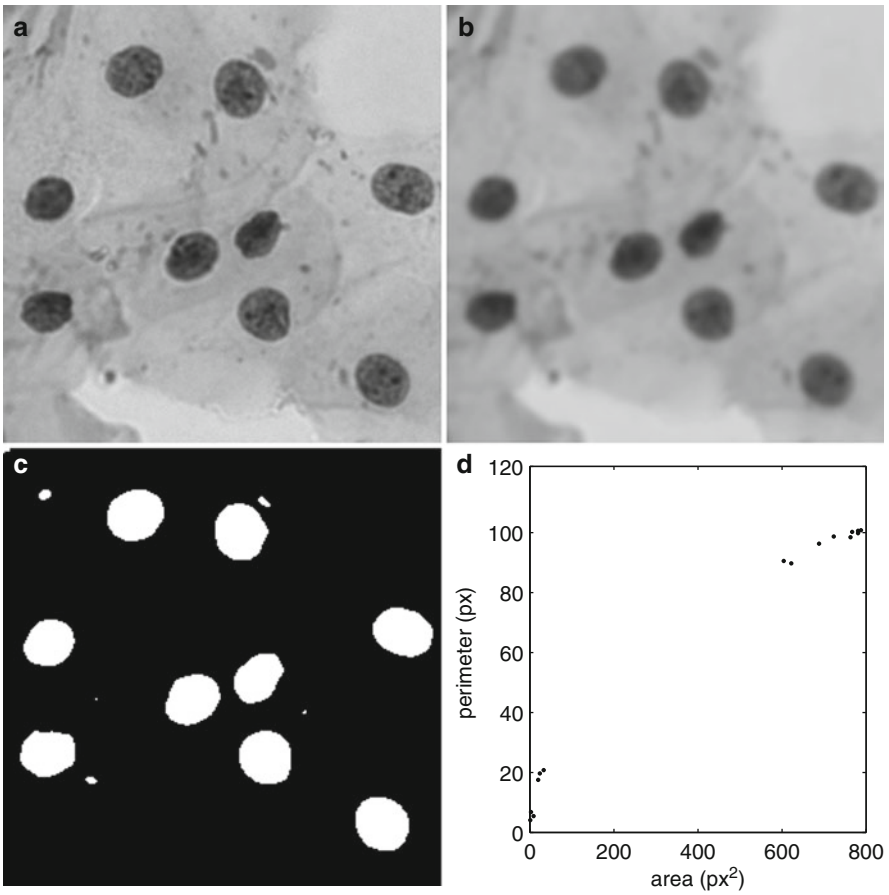

Fig. 2.1 A first look at DIPimage: (**a**) input image, (**b**) result of gaussf, (**c**) result of threshold, and (**d**) plot of measured area vs. perimeter

to the functions in the GUI for now. Let us assume that we still have the Pap smear image loaded in variable `a`. We select the "Gaussian filter" from the "Filters" menu. For the first parameter we select variable a, for the second parameter we enter 2, and for output image we type `b`. After clicking "Execute," we see

```
b = gaussf(a,2,'best')
Displayed in figure 11
```

on the command line, and the filtered image is shown in a window (Fig. 2.1b). Typing the above command would have produced the same result, without the need for the GUI. Because `'best'` is the default value for the third parameter, the same result would also have been accomplished typing only

```
b = gaussf(a,2)
```

Next we will select the "Threshold" function from the "Segmentation" menu, enter `-b` for input image and `c` for output image, and execute the command. We now have a binarized image, where the nuclei are marked as objects (red) and the rest as background (Fig. 2.1c). If we had left the minus sign out of the input to the threshold, the output would have been inverted. Finally, we select the "Measure" function from the "Analysis" menu, use `c` as the first input image, select several measurements by clicking on the "Select…" button (for example: "size," "center" and "perimeter," hold the control key down while clicking the options to select more than one), and execute the command. On the command window we will now see the result of the measurements. We can generate a plot of surface area ("size") vs. perimeter (Fig. 2.1d) by typing

```
figure, plot(msr.size, msr.perimeter,'.')
```

Note several small objects are found that are obviously not nuclei. Based on the size measure these can be discarded. We will see more of this type of logic in Sect. 2.4.

2.2.4 Scripting

If you collect a sequence of commands in a plain text file, and save that file with a ".m" extension, you have created a script. This script can be run simply by typing its name (without the extension) at the MATLAB command prompt. For example, if we create a file "analyse.m" with the following content:

```
a = readim('papsmear.tif');
b = smooth(a,2);
c = threshold(-b);
msr = measure(c,[],{'Size','Center','Perimeter'});
figure, plot(msr.size,msr.perimeter,'.')
```

then we can execute the whole analysis in this section by just typing

```
analyse
```

It is fairly easy to collect a sequence of commands to solve an application in such a file, execute the script to see how well it works, and modify the script incrementally. Because the GUI prints the executed command to the command line, it is possible to copy and paste the command to the script. The script works as both a record of the sequence of commands performed, and a way to reuse solutions. Often, when trying to solve a problem, one will start with the working solution to an old problem. Furthermore, it is easier to write programs that require loops and complex logic in a text editor than directly at the command prompt.

If you are a programmer, then such a script is obvious. However, compared to many other languages that require a compilation step, the advantage with MATLAB is that you can select one or a few commands and execute them independently of the rest of the script. You can execute the program line by line, and if the result of one line is not as expected, modify that line and execute it again, without having to run the whole script anew. This leads to huge time savings while developing new algorithms, especially if the input images are large and the analysis takes a lot of time.

2.3 Cervical Cancer and the Pap Smear

Cervical cancer is one of the most common cancers for women, killing about a quarter million women world-wide every year. In the 1940s, Papanicolaou discovered that vaginal smears can be used to detect the disease at an early, curable stage [2]. Such smears have since then commonly been referred to as Pap smears. Screening for cervical cancer has drastically reduced the death rate for this disease in the parts of the world where it has been applied [3]. Mass screens are possible because obtaining the samples is relatively simple and painless, and the equipment needed is inexpensive.

The Pap smear is obtained by collecting cells from the cervix surface (typically using a spatula), and spreading them thinly (by smearing) on a microscope slide (Fig. 2.2). The sample is then stained and analyzed under the microscope by a cytotechnologist. This person needs to scan the whole slide looking for abnormal cells, which is a tedious task because a few thousand microscopic fields of view need to be scrutinized, looking for the potentially few abnormal cells among the

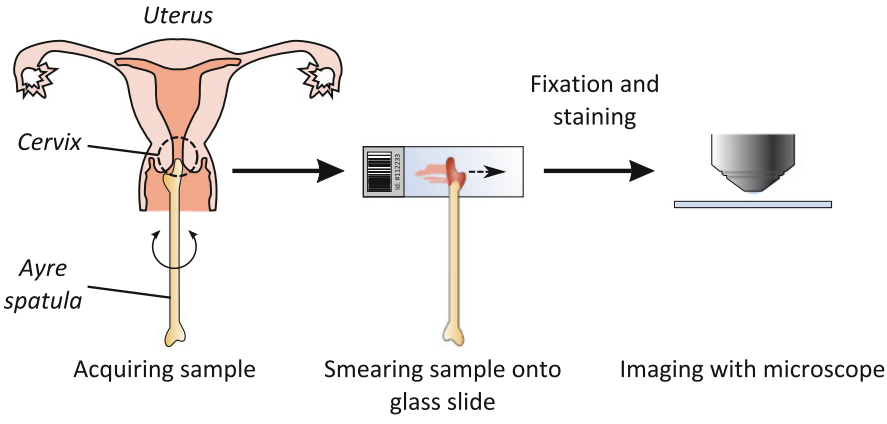

Fig. 2.2 A Pap smear is obtained by thinly smearing collected cells onto a glass microscope slide

several hundred thousand cells that a slide typically contains. This work is made even more difficult due to numerous artifacts: overlapping cells, mucus, blood, etc. The desire to automate the screening has always been great, for all the obvious reasons: trained personnel are expensive, they get tired, their evaluation criteria change over time, etc. The large number of images to analyze for a single slide, together with the artifacts, has made automation a very difficult problem that has occupied numerous image analysis researchers over the decades.

2.4 An Interactive, Partial History of Automated Cervical Cytology

This section presents an "interactive history," meaning that the description of methods is augmented with bits of code that you, the reader, can try out for yourself. This both makes the descriptions easier to follow, and illustrates the use of DIPimage to quickly and easily implement a method from the literature. This section is only a partial history, meaning that we show the highlights but do not attempt to cover everything; we simplify methods to their essence, and focus only on the image analysis techniques, ignoring imaging, classification, etc. For a somewhat more extensive description of the history of this field see for instance the paper by Bengtsson [4].

2.4.1 The 1950s

The first attempt at automation of Pap smear assessment was based on the observation that cancer cells are typically bigger, with a greater amount of stained material, than normal cells. Some studies showed that all samples from a patient with cancer had at least some cells with a diameter greater than 12 µm, while no normal cells were that large. And thus a system, the cytoanalyzer, was developed that thresholded the image at a fixed level (Fig. 2.3a), and measured the area (counted the pixels) and the integrated optical density (summed gray values) for each connected component [5]. To replicate this is rather straightforward, and very similar to what we did in Sect. 2.2.3:

```
a = readim('papsmear.tif');
b = a<128;
msr = measure(b,a,{'Size','Sum'});
```

As you can see, this only works for very carefully prepared samples. Places where multiple cytoplasms overlap result in improper segmentation, creating false large regions that would be identified as cancerous. Furthermore, the threshold value of 128 that we selected for this image is not necessarily valid for other images. This

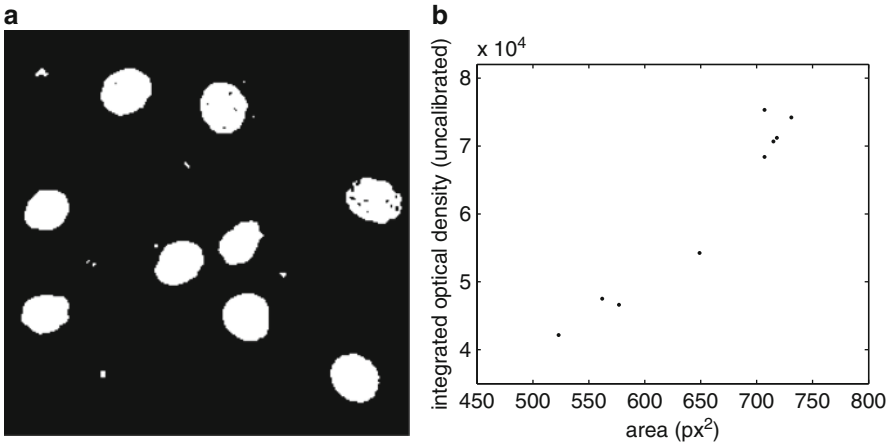

Fig. 2.3 (**a**) Segmented Pap-smear image and (**b**) plot of measured area vs. integrated optical density

requires strong control over the sample preparation and imaging to make sure the intensities across images are constant.

The pixel size in this machine was about 2 µm, meaning that it looked for segmented regions with a diameter above 6 pixels. For our image this would be about 45 pixels. The resulting data was analyzed as 2D scatter plots and if signals fell in the appropriate region of the plot the specimen was called abnormal (Fig. 2.3b):

```
figure, plot(msr.size,msr.sum,'.')
```

All the processing was done in hardwired, analog, video processing circuits. The machine could have worked if the specimens only contained well-preserved, single, free-lying cells. But the true signal was swamped by false signals from small clumps of cells, blood cells, and other debris [6].

2.4.2 The 1960s

One of the limitations of the cytoanalyzer was the fixed thresholding; it was very sensitive to proper staining and proper system setup. Judith Prewitt (known for her local gradient operator) did careful studies of digitized cell images and came up with the idea of looking at the histogram of the cell image [7]. Although this work was focused on the identification of red blood cells, the method found application in all other kinds of (cell) image analysis, including Pap smear assessment.

Assuming three regions with different intensity (nucleus, cytoplasm, and background), we would expect three peaks in the histogram (Fig. 2.4a). For simple shapes, we expect fewer pixels on the border between the regions than in the

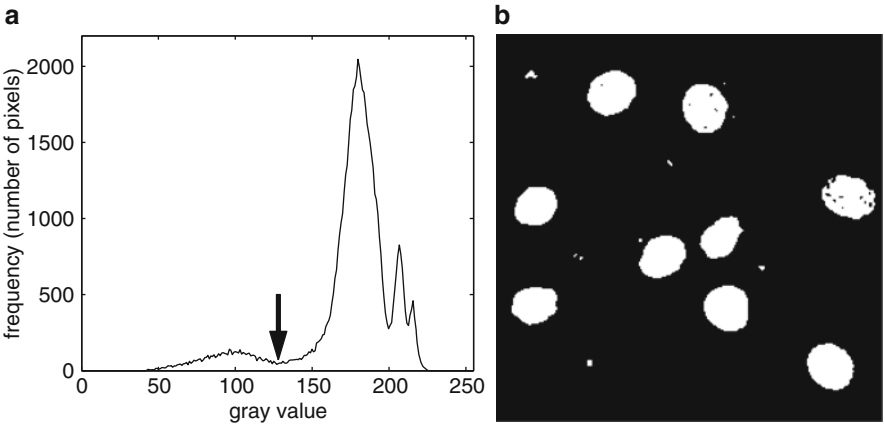

Fig. 2.4 (**a**) Histogram of Pap-smear image, with calculated threshold and (**b**) segmented image

regions themselves, meaning that there would be two local minima in between these three peaks, corresponding to the gray values of the pixels forming the borders. The two gray values corresponding to these two local minima are therefore good candidates for thresholding the image, thereby classifying each pixel into one of the three classes (Fig. 2.4b). Detecting these two local minima requires simplifying the histogram slightly, for example by a low-pass filter, to remove all the local minima caused by noise:

```
a = readim('papsmear.tif');
h = diphist(a);      % obtain a histogram
h = gaussf(h,3);     % smooth the histogram
t = minima(h);       % detect the local minima
t(h==0) = 0;         % mask out the minima at the tails
t = find(t)          % get coordinates of minima
a < t(1)             % threshold the image at the first
                       local minimum
```

Basically, this method substitutes the fixed threshold of the cytoanalyzer with a smoothing parameter for the histogram. If this smoothing value is taken too small, we find many more than two local minima; if it is too large, we do not find any minima. However, the results are less sensitive to the exact value of this parameter, because a whole range of smoothing values allows the detection of the two minima, and the resulting threshold levels are not affected too much by the smoothing. And, of course, it is possible to write a simple algorithm that finds a smoothing value such that there are exactly two local minima:

```
h = diphist(a);      % obtain a histogram
t = [0,0,0];         % initialize threshold array
while length(t)>2    % repeat until we have 2 local
                       minima
```

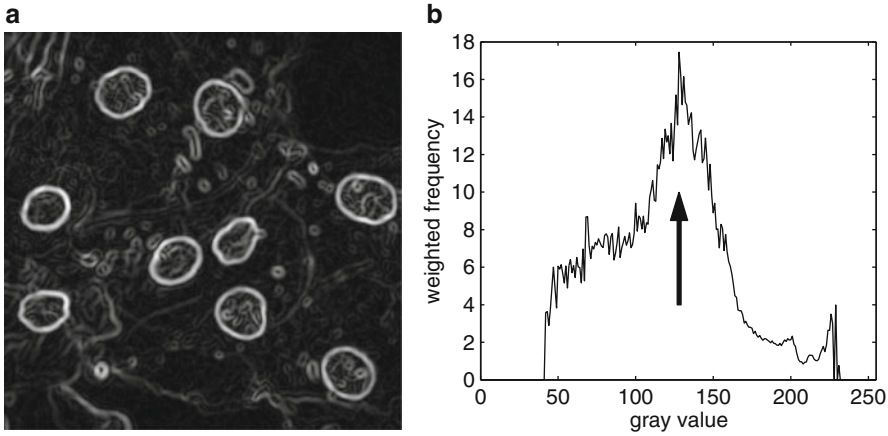

Fig. 2.5 (**a**) Gradient magnitude of Pap-smear image and (**b**) differential histogram computed from the Pap-smear image and its gradient magnitude

```
    h = gaussf(h,1); % (the code inside the loop is
                       identical to that used above)
    t = minima(h);
    t(h==0)=0;
    t = find(t);
end
```

The loop then repeats the original code, smoothing the histogram more and more, until at most two local minima are found.

2.4.3 The 1970s

In the late 1960s, a group at Toshiba, in Japan, started working on developing a Pap smear screening machine they called CYBEST. They used a differential histogram approach for the automated thresholding [8, 9]. That is, they computed a histogram weighted by a measure of edge strength; pixels on edges contribute more strongly to this histogram than pixels in flat areas. Peaks in this histogram indicate gray values that occur often on edges (Fig. 2.5). Though CYBEST used a different scheme to compute edge strength, we will simply use the Gaussian gradient magnitude (`gradmag`).

```
a = readim('papsmear.tif');
b = gradmag(a);
h = zeros(255,1);    % initialize array
for i = 1:255
    t = a==i;    % t is a binary mask
    n = sum(t); % n counts number of pixels with value i
```

```
    if n>0
    h(i) = sum(b(t))/n;    % average gradient at pixels
                             with value i
    end
end
h = gaussf(h,2);    % smooth differential histogram
[~,t] = max(h);     % find location of maximum
a < t    % threshold
```

A second peak in this histogram gives a threshold to distinguish cytoplasm from background, much like in Prewitt's method.

This group studied which features were useful for analyzing the slides and ended up using four features [10]: nuclear area, nuclear density, cytoplasmic area, and nuclear/cytoplasmic ratio. These measures can be easily obtained with the `measure` function as shown before. They also realized that nuclear shape and chromatin pattern were useful parameters but were not able to reliably measure these features automatically, mainly because the automatic focusing was unable to consistently produce images with all the cell nuclei in perfect focus. Nuclear shape was determined as the square of the boundary length divided by the surface area. Determining the boundary length is even more sensitive to a correct segmentation than surface area. The `measure` function can measure the boundary length (`'perimeter'`), as well as the *shape factor* (`'p2a'`). The shape factor, computed by $perimeter^2/(4\pi area)$, is identical to CYBEST's nuclear shape measure, except it is normalized to be 1 for a perfect circle. The chromatin pattern measure that was proposed by this group and implemented in CYBEST Model 4 is simply the number of blobs within the nuclear region [11]. For example (using the `a` and `t` from above):

```
m = gaussf(a) < t;    % detect nuclei
m = label(m)==3;      % pick one nucleus
m = (a < mean(a(m))) & m;    % detect regions within
                               nucleus
max(label(m))    % count number of regions
```

Here, we just used the average gray value within the nucleus as the threshold, and counted the connected components (Fig. 2.6). The procedure used in CYBEST was more complex, but not well described in the literature.

The CYBEST system was developed in four generations over two decades, and tested extensively, even in full scale clinical trials, but was not commercially successful.

2.4.4 The 1980s

In the late 1970s and 1980s, several groups in Europe were working on developing systems similar to CYBEST, all based on the so-called "rare event model," that is,

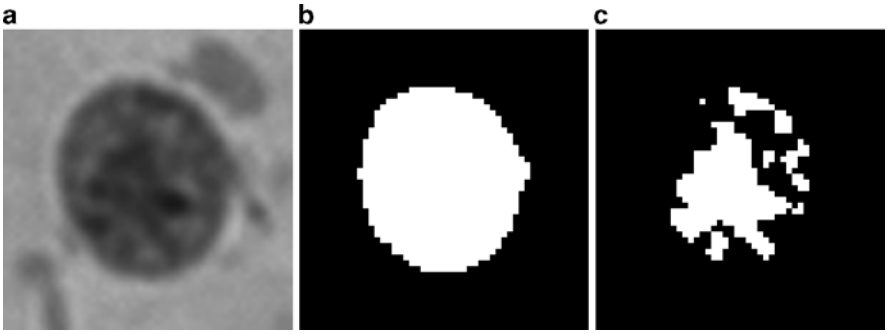

Fig. 2.6 A simple chromatin pattern measure: (**a**) the nucleus, (**b**) the nucleus mask, and (**c**) the high-chromatin region mask within that nucleus

looking for the few large, dark, diagnostic cells among the few hundred thousand normal cells. And these systems had to do this sufficiently fast, while avoiding being swamped by false alarms due to misclassifications of overlapping cells and small clumps of various kinds.

As a side effect of the experimentation on feature extraction and classification, carried out as part of this research effort, a new concept emerged. Several groups working in the field observed that even "normal" cells on smears from patients with cancer had statistically significant shifts in their features towards the abnormal cells. Even though these shifts were not strong enough to be useful on the individual cell level, it made it possible to detect abnormal specimens through a statistical analysis of the feature distributions of a small population, a few hundred cells, provided these features were extracted very accurately. This phenomenon came to be known as MAC, malignancy associated changes [12]. The effect was clearly most prominent in the chromatin pattern in the cell nuclei. The CYBEST group had earlier noted that it was very difficult to extract features describing the chromatin pattern reliably in an automated system. A group at the British Colombia Cancer Research Centre in Vancouver took up this idea and developed some very careful cell segmentation and chromatin feature extraction algorithms.

To accurately measure the chromatin pattern, one first needs an accurate delineation of the nucleus. Instead of using a single, global threshold to determine the nuclear boundary, a group in British Colombia used the gradient magnitude to accurately place the object boundary [13]. They start with a rough segmentation, and selected a band around the boundary of the object in which the real boundary must be (Fig. 2.7a):

```
a = readim('papsmear.tif');
b = gaussf(a,2)<128;    % quick-and-dirty threshold
c = b-berosion(b,1);    % border pixels
c = bdilation(c,3);  % broader region around border
```

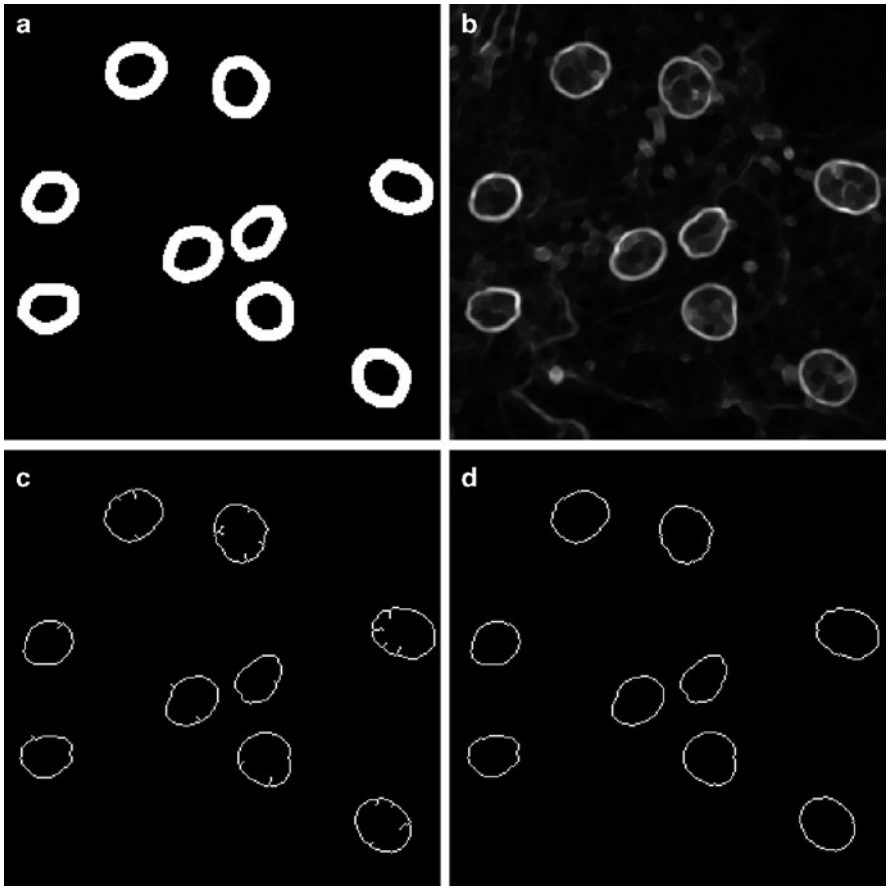

Fig. 2.7 Accurate delineation of the nucleus: (**a**) boundary regions, (**b**) gradient magnitude, (**c**) upper skeleton, and (**d**) accurate boundaries

Now comes the interesting part: a conditional erosion that is topology preserving (like the binary skeleton), but processes pixels in order of the gray value of the gradient magnitude (Fig. 2.7b), low gray values first. This implies that the skeleton will lie on the ridges of the gradient magnitude image, rather than on the medial axis of the binary shape. This operation is identical to the upper skeleton or upper thinning, the gray-value equivalent of the skeleton operation [14], except that the latter does not prune terminal branches nor isolated pixels (Fig. 2.7c). We can prune these elements with two additional commands (Fig. 2.7d):

```
g = gradmag(a);
g = closing(g,5);   % reducing number of local minima
                      in g
d = dip_upperskeleton2d(g*c);
     % compute upper skeleton in border region only
```

```
d = bskeleton(d,0,'looseendsaway');
     % prune terminal branches
d = d -getsinglepixel(d);    % prune isolated pixels
```

It is interesting to note, the upper skeleton is related to the watershed in that both find the ridges of the gray value image. We could have used the function `watershed` to obtain the exact same result.

We now have an accurate delineation of the contour. The following commands create seeds from the initial segmentation, and grow them to fill the detected contours:

```
e = b & ~c;
e = bpropagation(e,~d,0,1,0);
```

Because the pixels comprising the contours are exactly on the object edge, we need an additional step to assign each of these pixels to either the background or the foreground. In the paper, the authors suggest two methods based on the gray value of the pixel, but do not say which one is better. We will use option 1: compare the border pixel's value to the average for all the nuclei and the average for all the background, and assign it to whichever class it is closest:

```
gv_nuc = mean(a(e));    % average nuclear gray value
gv_bgr = mean(a(~(d|e)));    % average background gray
                               value
t = a < (gv_nuc+gv_bgr)/2;   % threshold halfway between
                               the two averages
e(d) = t(d);    % reassign border pixels only
```

The other option is to compare each border pixel with background and foreground pixels in the neighborhood, and will likely yield a slightly better result for most cells.

A very large group of features were proposed to describe each nucleus. Based on the outline alone, one can use the mean and maximum radius, sphericity, eccentricity, compactness, elongation, etc., as well as Fourier descriptors [15]. Simple statistics of gray values within one nucleus are maximum, minimum, mean, variance, skewness, and kurtosis. Texture features included contour analysis and region count after thresholding the nucleus into areas of high, medium and low chromatin content; statistics on the co-occurrence matrix [16] and run lengths [17]; and the fractal dimension[18]. The fractal dimension is computed from the fractal area measured at different resolutions, and gives an indication of how the image behavior changes with scale. The fractal area is calculated with:

```
fs = 1 + abs(dip_finitedifference(a,0,'m110')) + ...
    abs(dip_finitedifference(a,1,'m110'));
m = label(e)==4;    % pick one nucleus
sum(fs(m))    % sum values of fs within nucleus
```

The function `dip_finitedifference` calculates the difference between neighboring pixels, and is equivalent to MATLAB's function `diff`, except it returns an image of the same size as the input.

Multivariate statistical methods were finally used to select the best combination of features to determine whether the cell population was normal or from a slide influenced by cancer.

2.4.5 The 1990s

In the 1990s, research finally lead to successful commercial systems being introduced: AutoPap [19] and PAPNET [20]. They built on much of the earlier research, but two concepts were extensively used in both of these commercial systems: mathematical morphology and neural networks. In short, what the PAPNET systems did was detect the location of nuclei of interest, extract a square region of fixed size around this object, and use that as input to a neural network that classified the object as debris/benign/malignant [21]. Using such a system, these machines avoided the issues of difficulty in segmentation, careful delineation, and accurate measurement. Instead, the neural network does all the work. It is trained with a large collection of nuclei that are manually classified, and is then able to assign new nuclei to one of the classes it was trained for. However, the neural network is a "black box" of sorts, in that it is not possible to know what features of the nucleus it is looking at to make the decision [22]. Slightly simplified, the method to extract fixed-sized image regions containing a nucleus is as follows:

```
a = readim('papsmear.tif');
b = gaussf(a);  % slight smoothing of the image
b = closing(b,50)-b;    % top-hat, max. diameter is 50
                          pixels
c = threshold(b);
```

The closing minus the input image is a top-hat, a morphological operation that eliminates large dark regions. The result, `c`, is a mask where all the large objects (nuclei clusters, for example) have been removed. But we also want to make sure we only look at objects that are dark in the original image:

```
c = c & ~threshold(a);
```

`c` now contains only objects that are dark and small (Fig. 2.8a). The next step is to remove the smallest objects and any object without a well-defined edge:

```
d = gradmag(a,3);
d = threshold(d);    % detect strong edges
d = brmedgeobjs(~d);    % find inner regions
```

The first step finds regions of large gradient magnitude. In the second step we invert that mask and remove the part that is connected to the image border.

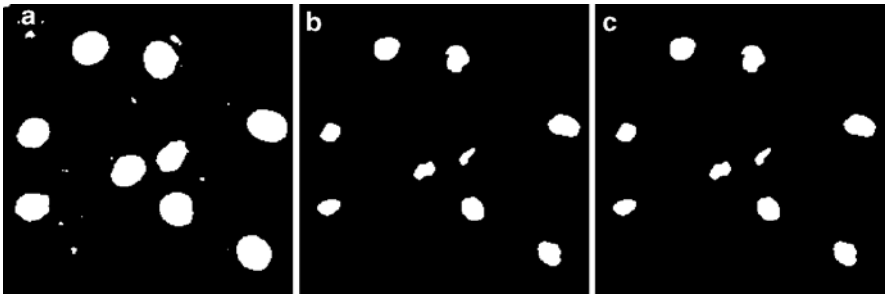

Fig. 2.8 Detecting the location of possible nuclei: (**a**) all dark, small regions, (**b**) all regions surrounded by strong edges, and (**c**) the combination of the two

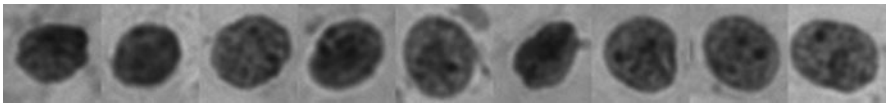

Fig. 2.9 Extracted regions around potential nuclei. These subimages are the input to a neural network

The regions that remain are all surrounded by strong edges (Fig. 2.8b). The combination of the two partial results,

```
e = c & d;
```

contains markers for all the medium-sized objects with a strong edge (Fig. 2.8c). These are the objects we want to pass on to the neural network for classification. We shrink these markers to single-pixel dots and extract an area around each dot (Fig. 2.9):

```
e = bskeleton(e,0,'looseendsaway');
    % reduce each region to a single dot
coords = findcoord(e);    %get the coordinates for each
                            dot
N = size(coords,1); % number of dots
reg = cell(N); % we'll store the little regions in here
for ii=1:N
    x = coords(ii,1);
    y = coords(ii,2);
    reg{ii} = a(x-20:x+20,y-20:y+20);
        % the indexing cuts a region from the image
end
reg = cat(1,reg{:}) % glue regions together for display
```

That last command just glues all the small regions together for display. Note that, for simplicity, we did not check the `coords` array to avoid out-of-bounds indexing when extracting the regions. One would probably want to exclude regions that fall partially outside the image.

The PAPNET system recorded the 64 most "malignant looking" regions of the slide for human inspection and verification, in a similar way to what we did just for our single image field.

2.4.6 The 2000s

There was a reduced academic research activity in the field after the commercial developments took over in the 1990s. But there was clearly room for improvements, so some groups continued basic research. This period is marked by the departure from the "black box" solutions, and a return to accurate measurements of specific cellular and nuclear features. One particularly active group was located in Brisbane, Australia [23]. They applied several more modern concepts to the Pap smears and demonstrated that improved results could be achieved. For instance, they took the dynamic contour concept (better known as the "snake," see Chapter 4) and applied it to cell segmentation [24]. Their snake algorithm is rather complex, since they used a method to find the optimal solution to the equation, rather than the iterative approach usually associated with snakes, which can get stuck in local minima. Using "normal" snakes, one can refine nuclear boundary thus:

```
a = readim('papsmear.tif')
b = bopening(threshold(-a),5);  % quick-and-dirty
    %segmentation
c = label(b);     % label the nuclei
N = max(c);       % number of nuclei
s = cell(N,1);    % this will hold all snakes
vf = vfc(gradmag(a));    % this is the snake's ''external
                           force''
for ii = 1:N    % we compute the snake for each nucleus
                  separately
    ini = im2snake(c==ii);    % initial snake given by
                                segmentation
    s{ii} = snakeminimize(ini,vf,0.1,2,1,0,10);
        % move snake so its energy is minimized
    snakedraw(s{ii}) % overlays the snake on the image
end
```

The snake is initialized by a rough segmentation of the nuclei (Fig. 2.10a), then refined by an iterative energy minimization procedure (`snakeminimize`, Fig. 2.10b). Note that we used the function `vfc` to compute the *external force*, the image that drives the snake towards the edges of the objects. This VFC (vector field

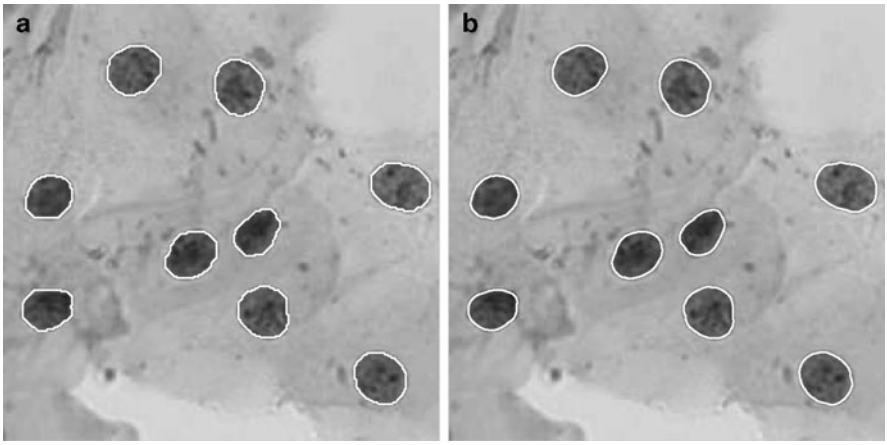

Fig. 2.10 (**a**) Initial snake and (**b**) final snake in active contour method to accurately delineate nuclei

convolution) approach is a recent improvement to the traditional snake [25]. For this example image, the results are rather similar when using the traditional gradient, because the initial snake position is close to its optimal. Also note the large number of parameters used as input to the function `snakeminimize`, the function that moves the control points of the snake to minimize the snake's energy function. This large number of parameters (corresponding to the various weights in the energy function) indicates a potential problem with this type of approach: many values need to be set correctly for the method to work optimally, and thus this particular program is only applicable to images obtained under specific circumstances.

2.5 The Future of Automated Cytology

An interesting observation that can be made from the brief samples of the long history of automated cervical cytology that has been presented in this chapter is that the focus of the research in the field has been moving around the world about once every decade – Eastern USA, Japan, Europe, Western USA/Canada, and Australia (Fig. 2.11) – although there are, of course, outliers to this pattern. This pattern might have arisen because of the apparent ease of the problem: when researchers in one region have been making strong promises of progress for too long, without being able to deliver on these promises, it becomes increasingly difficult to obtain more research funds in that region; researchers in a different region in the world are then able to take the lead.

One significant development that we have not discussed so far is the efforts of producing cleaner specimens that are more easy to analyze than the smears, which can be very uneven in thickness and general presentation of the cells. These

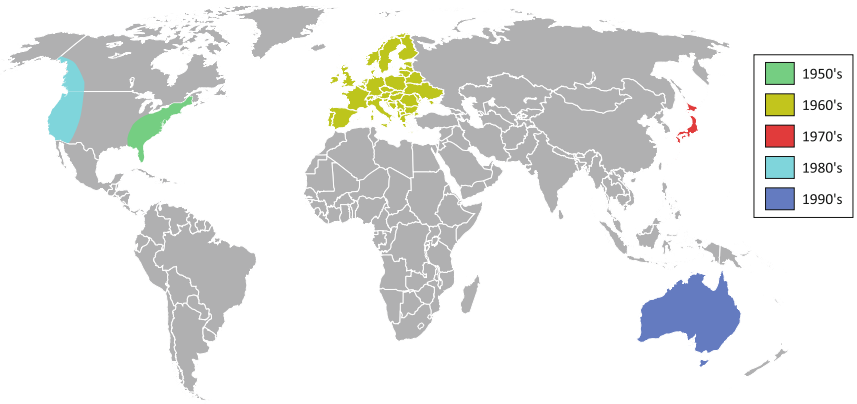

Fig. 2.11 A map of the world showing the location of major Pap smear analysis automation research over the decades

efforts led to two commercial liquid cytology systems in the 1990s [26, 27]. The companies behind these sample preparation devices have also developed dedicated image analysis systems. These devices work well, but the modified kits for preparing the specimens are so expensive that most of the economic gain from automation disappears.

The cost of automation is an important issue. One of the original motivations for developing automation was the high cost of visual screening. Still, the first generation automated systems were very complex and expensive machines costing about as much to operate as visual screening. The need for modified sample preparation for some systems added to these costs. A third aspect was that the combined effect of visual and machine screening gives a higher probability of actually detecting a lesion than either one alone, making it hard in some countries, due to legal liability reasons, to use automation alone even if it is comparable in performance to visual screening. All of this has made the impact of automation very limited in the poorer parts of the world, so cervical cancer is still killing a quarter million women each year. Most of these deaths could be prevented by a globally functioning screening program.

A significant challenge for the future, therefore, is to come up with a screening system that is significantly cheaper than the present generation. How can this be achieved? Looking at the history we can see two approaches.

One is the "rare event" approach. Modern whole-slide scanners are much more robust, cheaper, and easier to operate than earlier generations of robot microscopes. With software for such systems, a competitive screening system should be possible, perhaps utilizing a somewhat modified specimen preparation approach that gives cleaner specimens without the high cost of the present liquid-based preparations.

The alternative approach is to use the MAC concept. The obstacle there is to achieve sufficiently robust imaging to consistently detect the subtle changes of chromatin texture between normal and malignancy-influenced cells in an automated

system. A malignancy-influenced cell looks too much like a normal cell when it is slightly out of focus or poorly segmented. Here, modern 3D scanning concepts may make a significant difference.

So perhaps the next generation systems will be developed in third-world countries, to solve their great need for systems that are better and more economical than the systems that have been developed in the richer parts of the world.

2.6 Conclusions

As we have seen, the availability of a toolbox of image analysis routines greatly simplifies the process of "quickly trying out" an idea. Some of the algorithms that we approximated using only two or three lines of code would require many hundreds of lines of code without such a toolbox.

Another benefit to using an environment like MATLAB is the availability of many other tools not directly related to images that are very useful in developing novel algorithms. For example, in this chapter we have created graphs with some of the intermediate results. Being able to graphically see the numbers obtained is invaluable. In Sect. 2.4.5, we implemented only the first stage of the PAPNET system, but with equal ease we could have constructed the rest, using any of the neural network toolboxes that exist for MATLAB.

These two benefits are augmented in the MATLAB/DIPimage environment with an interpreted language, allowing interactive examination of intermediate results, and a high-level syntax, allowing easy expression of mathematical operations. The downside is that certain algorithms can be two orders of magnitude faster when expressed in C than in MATLAB, and algorithms written in MATLAB are more difficult to deploy. A common approach is to develop the algorithms in MATLAB, and translate them to C, C++, or Java when the experimentation phase is over. Though it is not always trivial to translate MATLAB code to C, MATLAB code that uses DIPimage has a big advantage: most of the image processing and analysis routines are implemented in a C library, DIPlib, that can be distributed independently of the DIPimage toolbox and MATLAB. All these things considered, even having to do the programming a second time in C, one can save large amounts of time when doing the first development in an environment such as that described here.

References

1. Luengo Hendriks, C.L., van Vliet, L.J., Rieger, B., van Ginkel, M., Ligteringen, R.: DIPimage User Manual. Quantitative Imaging Group, Delft University of Technology, Delft, The Netherlands (1999–2010)

2. Traut, H.F., Papanicolaou, G.N.: Cancer of the uterus: the vaginal smear in its diagnosis. Cal. West. Med. 59(2), 121–122 (1943)
3. Christopherson, W.M., Parker, J.E., Mendez, W.M., Lundin Jr., F.E.: Cervix cancer death rates and mass cytologic screening. Cancer **26**(4), 808–811 (1970)
4. Bengtsson, E.: Fifty years of attempts to automate screening for cervical cancer. Med. Imaging Technol. **17**(3), 203–210 (1999)
5. Tolles, W.E., Bostrom, R.C.: Automatic screening of cytological smears for cancer: the instrumentation. Ann. N. Y. Acad. Sci. **63**, 1211–1218 (1956)
6. Spencer, C.C., Bostrom, R.C.: Performance of the Cytoanalyzer in recent clinical trials. J. Natl. Cancer Inst. **29**, 267–276 (1962)
7. Prewitt, J.M.S., Mendelsohn, M.L.: The analysis of cell images. Ann. N. Y. Acad. Sci. 128(3), 1035–1053 (1965)
8. Watanabe, S., the CYBEST group: An automated apparatus for cancer prescreening: CYBEST. Comput. Graph. Image Process. **3**(4), 350–358 (1974)
9. Tanaka, N., Ikeda, H., Ueno, T., Watanabe, S., Imasato, Y.: Fundamental study of automatic cyto-screening for uterine cancer. II. Segmentation of cells and computer simulation. Acta Cytol. **21**(1), 79–84 (1977)
10. Tanaka, N., Ikeda, H., Ueno, T., Takahashi, M., Imasato, Y.: Fundamental study of automatic cyto-screening for uterine cancer. I. Feature evaluation for the pattern recognition system. Acta Cytol. **21**(1), 72–78 (1977)
11. Tanaka, N., Ueno, T., Ikeda, H., Ishikawa, A., Yamauchi, K., Okamoto, Y., Hosoi, S.: CYBEST model 4: automated cytologic screening system for uterine cancer utilizing image analysis processing. Anal. Quant. Cytol. Histol. **9**(5), 449–453 (1987)
12. Burger, G., Jutting, U., Rodenacker, K.: Changes in benign cell populations in cases of cervical cancer and its precursors. Anal. Quant. Cytol. **3**(4), 261–271 (1981)
13. MacAulay, C., Palcic, B.: An edge relocation segmentation algorithm. Anal. Quant. Cytol. Histol. **12**(3), 165–171 (1990)
14. Serra, J.: Image Analysis and Mathematical Morphology. Academic, London (1982)
15. Kuhl, F.P., Giardina, C.R.: Elliptic Fourier features of a closed contour. Comput. Graph. Image Process. **18**(3), 236–258 (1982)
16. Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural features for image classification. IEEE Trans. Syst. Man Cybern. **3**(6), 610–621 (1973)
17. Galloway, M.M.: Texture analysis using gray level run lengths. Comput. Graph. Image Process. **4**(2), 172–179 (1975)
18. MacAulay, C., Palcic, B.: Fractal texture features based on optical density surface area. Use in image analysis of cervical cells. Anal. Quant. Cytol. Histol. **12**(6), 394–398 (1990)
19. Lee, J., Nelson, A., Wilbur, D.C., Patten, S.F.: The development of an automated Papanicolaou smear screening system. Cancer **81**, 332–336 (1998)
20. DeCresce, R.P., Lifshitz, M.S.: PAPNET cytological screening system. Lab Med. **22**, 276–280 (1991)
21. Luck, R.L., Scott, R.: Morphological classification system and method. US Patent 5,257,182, 1993
22. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, New York (2001)
23. Mehnert, A.J.H.: Image Analysis for the Study of Chromatin Distribution in Cell Nuclei. Ph.D. Thesis, University of Queensland, Brisbane, Australia, 2003
24. Bamford, P., Lovell, B.: Unsupervised cell nucleus segmentation with active contours. Signal Process. **71**(2), 203–213 (1998)
25. Li, B., Acton, S.T.: Active contour external force using vector field convolution for image segmentation. IEEE Trans. Image Process. **16**(8), 2096–2106 (2007)
26. Hutchinson, M.L., Cassin, C.M., Ball, H.G.: The efficacy of an automated preparation device for cervical cytology. Am. J. Clin. Pathol. **96**(3), 300–305 (1991)
27. Howell, L.P., Davis, R.L., Belk, T.I., Agdigos, R., Lowe, J.: The AutoCyte preparation system for gynaecologic cytology. Acta Cytol. 42(1), 171–177 (1998)

http://www.springer.com/978-1-4419-9769-2

Medical Image Processing
Techniques and Applications
Dougherty, G. (Ed.)
2011, XVI, 380 p., Hardcover
ISBN: 978-1-4419-9769-2