6

Characteristics and Value of Machine Learning for Imaging in High Content Screening

Juergen A. Klenk

Summary

Requirements for a flexible image analysis package for high content screening (HCS) are discussed. An overview of tools and techniques for image analysis and machine learning is given. Machine learning for classification and segmentation, the two fundamental elements of image analysis, is discussed. Next generation image analysis packages for HCS are reviewed. Recommendations for the development of image analysis solutions for advanced assays are given.

Key Words: Classification; computer vision; high content screening (HCS); image analysis; machine learning; morphology operations; neural networks; segmentation; semantic networks; thresholding; training.

1. Introduction

There is one universal constant in the field of high content screening (HCS): change—rapid change to be precise! Researchers' requirements for image quantification change constantly as new technologies, instruments, fluorophores, and labeling reagents become available. This continuous change in requirements makes it difficult, if not impossible, for providers of image analysis packages to keep up with the latest needs of their clients. Combining elements of machine intelligence with computer vision based algorithms might remove this bottleneck.

Both computer vision and machine learning are huge fields, and more number of researches has been conducted in refining methods to approach human vision capabilities. The bad news is that despite all efforts there is still no single generic method, which adequately replicates human vision. The good news is that acceptable results can be achieved by optimizing methods for particular domains. Accordingly, we will focus on image analysis methods for biomedical, specifically cellular images. And because this book serves as a practical guide, we will aim for practical recommendations over technical breadth and depth.

2. Problem Description

2.1. HCS User Types Requiring Imaging

Image analysis can be viewed as an independent component that is used along the generic HCS workflow during the three phases as shown in **Fig. 1**, i.e., assay development, screening, and assay evaluation. The usage of image analysis packages varies significantly between these three phases. Assay development users consists of an assay development scientist, a technician, and possibly an image analysis expert who often doubles as technician. They usually have lower

From: *Methods in Molecular Biology, vol. 356:*
High Content Screening: A Powerful Approach to Systems Cell Biology and Drug Discovery
Edited by: D. L. Taylor, J. R. Haskins, and K. Giuliano © Humana Press, Inc., Totowa, NJ

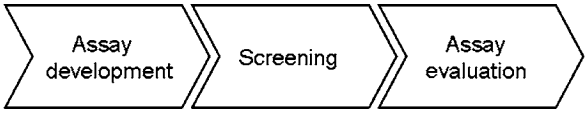

Fig. 1. Generic HCS workflow phases which include image analysis.

Fig. 2. Image analysis workflow.

throughput requirements as they use the image analysis software to validate their assays and to configure the corresponding image analysis protocol for screening.

Screening users consists of a screening scientist and a screening technician, with high throughput requirements to analyze the screens on- or near-line. They use the image analysis software for quality control and might perform minimal fine-tuning of the image analysis protocol.

During the assay evaluation phase a scientist uses data mining and visualization software such as Spotfire DecisionSite or ID Business Solutions (IDBS) ActivityBase to inspect the run data. The image analysis software comes into play by supporting simultaneous inspection of data and corresponding result images. Although image analysis plays a role in all three phases, it is predominantly during assay development that limited flexibility of existing image analysis packages presents a problem to the scientists. Thus, for the remainder of this article we will concentrate on how image analysis can be (and has been, in part) improved using machine learning techniques to optimally support the flexibility needed for rapid development of new assays. But first, let us review some terminology in image analysis and machine learning.

2.2. Image Analysis 101

Image analysis consists of several steps, starting from image loading to the extraction of the results. We specifically exclude the image acquisition part, and assume that knowledge about the camera, illumination, magnification, and so on is provided as metadata and available to the image analysis system. **Figure 2** illustrates the key steps of image analysis.

The processes of loading an image as well as extracting results and writing them to a database require a flexible input/output interface to handle data formats and other specificities of an existing HCS environment, a feature that every hardware-independent image analysis package should offer.

Image preprocessing predominantly deals with compensating acquisition artifacts. There is an abundance of preprocessing strategies. They include pixel brightness transformations (histogram-based equalization, contrast stretching) and local preprocessing (smoothing, convolution operations, noise reduction filtering), to name just a few of the more commonly used methods. Every comprehensive image analysis package should allow for a proper preprocessing of the raw image as this will dramatically improve analysis results. Preprocessing methods are well described in the literature *(1–5)* and can be viewed as independent of the subsequent image analysis—at least for the purpose of this article. Thus, we will assume throughout this article that images have already been preprocessed and artifacts have been removed as much as possible for optimal analysis results.

The two remaining elements of image analysis that we will concentrate on in this article are segmentation and classification (or recognition). Image segmentation is the process of partitioning an image into objects, and image classification is the process of naming these objects. Does

segmentation lead to classification or classification lead to segmentation? Regardless of one's philosophical stand on this question, it is undeniable that a tight connection exists between them, like the chicken-and-egg problem. How can you segment an image if you do not already know what is in the image, and how can you classify an object if you do not already know whether you have got it right? Our research shows that this problem should be solved just as nature solves all chicken-and-egg-type problems: with an evolutionary strategy. In other words, the best image analysis results can be achieved with algorithms that rely on an iteration of segmentation and classification steps. We will discuss segmentation and classification methods in the next section.

2.3. Machine Learning 101

Machine learning is an area of artificial intelligence that deals with finding an optimal algorithm to more or less solve a complex problem. Obviously, our specific problem is analyzing a set of images. There are two key concepts in machine learning, the "finding" strategy for the optimal algorithm and the quality of the "approximation." The "finding" strategy is usually referred to as the machine learning method, and we will discuss such methods in the next section. The quality of the "approximation" is the deviation of the approximate result produced by the learned algorithm from the ideal result, often produced by a human. It is important to realize that for complex problems, finding an algorithm that provides an exact solution is computationally intractable, and in many cases such an algorithm does not even exist. Thus, the key of machine learning is to find an algorithm that consistently solves a problem well enough, according to the user. In other words, it should produce results, which deviate only an acceptable amount from the ideal results in almost all cases.

Learning is achieved by presenting the machine learning method with training data. There are two fundamental types of learning methods, unsupervised learning, and supervised learning. Unsupervised learning relies on purely statistical analysis of the training data and does not require any user input. Simply speaking, the "finding" strategy looks for the algorithm that produces the "most likely" result, given the training data and its statistics. It is usually employed when the user has no or very limited *a priori* knowledge about the problem space. By contrast, supervised learning relies on user feedback, and learning is achieved by comparing machine-produced with user-produced results on the training data, and iteratively improving the algorithm "in the direction" of the desired results until an optimum is reached.

The ultimate challenge in machine learning is that a "learned" algorithm works well on (previously unseen) test data. Only the comparison against such test data provides an objective measure of the quality of the "learned" algorithm, and ultimately of the machine learning method. As intuitive as this might sound, this fact is often overlooked, and quality measures on the training data are given instead.

2.4. Limitations of Conventional Imaging Systems

The problem of conventional image analysis packages lies in the fact that they were developed for and hence are limited to specific assays. They rely on specific cell lines, labeling reagents, phenotypic changes, readouts, and so on. For which specific image analysis algorithms were developed that can analyze a particular set of parameters and allow some configuration by the user. Thus, they provide very limited or no flexibility to the assay development scientist, even if they offer some degree of configuration capability.

For example, a customized algorithm allows measuring the translocation of a green fluorescent protein (GFP)-labeled protein from the cytoplasm to the nucleus as follows: suppose the nuclei are stained with Hoechst, then a simple intensity-based thresholding in that channel will segment the nuclei if they are reasonably well separated. A simple distance-based doughnut around each nucleus provides a reasonably good approximation for the cytoplasm. Once nucleus and cytoplasm are detected, the algorithm then simply needs to compute the GFP-intensities in each compartment and their ratio. The user simply chooses the desired parameter (nucleus/cytoplasm),

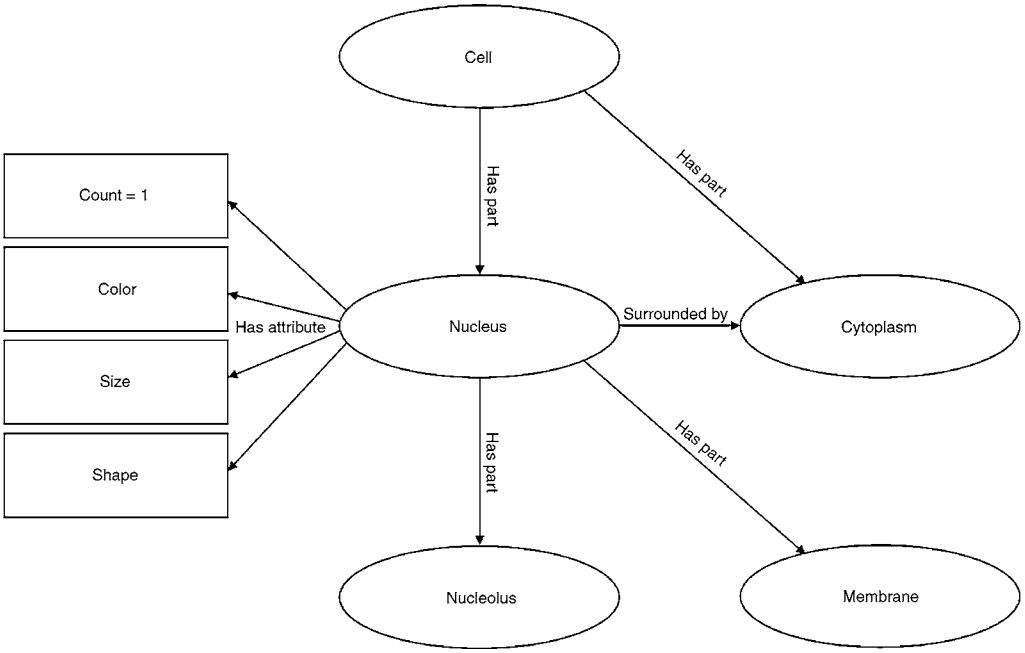

Fig. 3. Semantic network of a nucleus.

and he has three dials to configure the analysis, the nuclear-stain-intensity threshold, the distance, and the GFP-intensity threshold, to adjust nucleus size, cell size, and labeled proteins, respectively.

Such an approach has some obvious limitations. First, the nucleus segmentation algorithm does not work for all assays. Even a smart adjustment of the intensity threshold by the user will not always lead to a desired level of quality for nucleus detection, especially if the nuclei are very close to one another and are not stained homogeneously.

Second, the assumption that the cytoplasm is located in a doughnut around the nucleus is too simplistic. It will miss a significant amount of cytoplasm, and worse, it will pick up background noise. This in turn produces a significant error for the GFP signal in the cytoplasm.

The reason that solutions based on the above algorithm were developed despite their limitations is that it was too hard to develop an algorithm that could correctly detect the structures of interest—nucleus and cytoplasm—under all circumstances. And this is the biggest drawback of latest image analysis packages: there is no way for the scientist to reliably detect all structures of interest with the provided algorithms! In other words, no dial setting; however, smart, will yield a satisfactory analysis for most new assays.

The reason for the limitations of our example, and for most of latest image analysis packages, is that one attribute—intensity—to describe our structures of interest simply is not enough, especially if the restriction to specific assays is removed. Even two or three attributes are not good enough in most cases. Ask yourself—if you were to describe a nucleus, what attributes would you come up with? Its size, shape, color, and texture are just a few such intrinsic attributes. But then there are also extrinsic attributes, or relations, to other structures. A nucleus has a boundary, the nuclear membrane, it has nucleoli inside, and it is surrounded by cytoplasm. Furthermore, there is exactly one nucleus per cell (well, simply speaking, but of course we could also dive into providing phenotypic descriptions and thus distinguish between particular types of nuclei). **Figure 3** provides a sketch of such a description, also referred to as a semantic network. These are more of things that an algorithm could (should? must?) take into account to help reliably detect nuclei and other structures of interest in a wide range of situations.

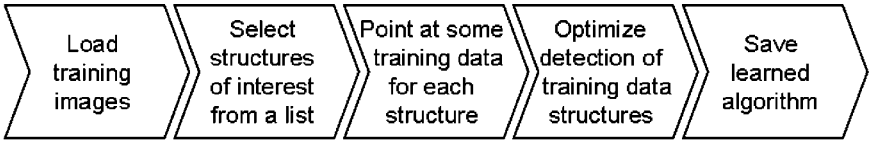

Fig. 4. Workflow of a machine learning based image analysis solution.

But now we have ended up with too many parameters to adjust manually. Imagine a nucleus detection algorithm with 100 independent dials to adjust. You would not know where to begin. And the effect of each dial is not always immediately apparent. There even seems to be crosstalk between the individual dials. A user would be simply overwhelmed by the complexity and we are only on our first structure, the nucleus.

This is where machine learning comes into play. It can assist the user and find the optimal algorithm for nucleus detection by automatically determining the best settings for all dials. As we now know, this takes some training data: the user must point to some examples of well-identified nuclei (and similarly for other structures of interest). From this the machine learning method can learn the settings, which optimally reproduce the training results. If the training data have been well selected (we shall come back to this later), then these settings should also work well on previously unseen test data.

2.5. Features of a Machine Learning-Based Imaging System

More generally, what is needed is a system that can learn to automatically detect structures of interest such as nuclei, cytoplasm, cell membrane, other subcellular structures, or entire cells or even cellular structures, independent of the assay type. It should be a relatively simple process to teach the system how to automatically and reliably detect these structures—ideally much in the same way, as we would educate a student—by pointing at them and naming them on training data. **Figure 4** provides an illustration of key elements of a machine learning-based image analysis solution. The key here is that the individual workflow components must be developed in a generic way so that they will work with (almost) any assay.

3. Tools and Techniques

3.1. Machine Learning for Classification

Using the terminology from the previous section, machine learning methods for classification must help us find an optimal classification scheme for the provided training data. Probably the best-known technique for classification is neural networks. This technique uses a supervised learning algorithm called the backpropagation algorithm. Providing input and output layers and a network of nodes (neurons) and weighted links (axons) in between, neural networks compute an output signal by propagating an input signal through the weighted network. The input nodes are connected to the objects' attributes, and the output nodes to the class types. The backpropagation algorithm changes the weights in the network until an optimal classification scheme for the training data is obtained. Usually this classification scheme provides a very good extrapolation to unseen test data. Furthermore, neural networks feature classification likelihood measures, which yield not just one classification result for each object, but a table with a list of classes it might belong to and the respective likelihoods.

Another technique for classification is known as nearest neighbor, with a learning algorithm based on clustering of the training data. There are many clustering strategies, most of which are geometrically intuitive. Their goal is to group the data points in attribute space which represent the training data objects into sets or clusters of data points which belong to the same class. This is achieved by drawing separation hyperplanes (or, more generally, hypersurfaces), and, thus, partitioning the attribute space in the best possible way so that each partition only contains data

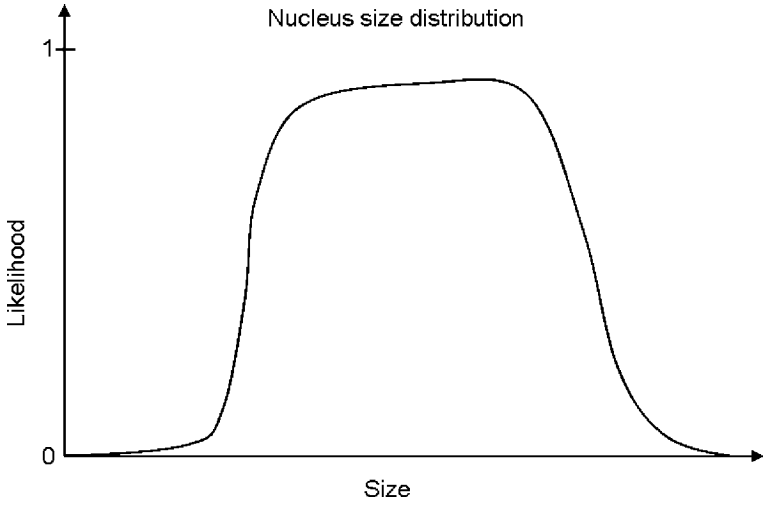

Fig. 5. Fuzzy logical expression of nucleus size.

points (objects) of one class. Methods such as principal component analysis might be employed to reduce the dimensionality of the problem by weeding out dependent attributes. Classification of training or test data objects is then achieved by choosing the class type of the nearest partition. Like neural networks, nearest neighbors also provides classification likelihoods, using a distance measure from partition centerpoints.

Decision trees are a very old technique for classification. A classification result is obtained by traversing a tree, with a decision being made at each branch point, until an end point is reached. Decisions are based on attribute values. The end points of a decision tree are class types. Algorithms have been described in the literature to optimize the decision points in order to improve the overall classification result. In practice, decision trees are limited to simply structured problems and are thus not too useful for our needs.

The last technique we discuss is a knowledge-based approach, also known as semantic networks. We touched on semantic networks already when we gave a knowledge-based description of a nucleus. Semantic networks are not completely independent of the previously discussed techniques. In fact, one can argue that the selection of input nodes for neural networks as well as the selection of attribute dimensions in nearest neighbors is best done by employing prior knowledge about the structures to be identified. However, a knowledge-based approach goes further in that it does not only select the relevant attributes, but also provides meaningful ranges for them. For nuclei, typical ranges for size, shape, color, texture, and so on are given. Mathematically this can be best expressed in terms of fuzzy logical, which provides a way to link attribute values to classification likelihoods. **Figure 5** shows a sample fuzzy logical expression for nucleus size. The basic idea of fuzzy logical is that shapes other than a step function are allowed as likelihood curves.

There are many ways to learn the attribute ranges from the training data. The simplest technology is the minimum to maximum approach, taking for each attribute the minimum and maximum value encountered in the training data and declaring the interval between minimum and maximum as the allowed range, with a classification probability of 1 (i.e., using a step function). This special case of a semantic network coincides with the nearest neighbors approach with mutually orthogonal separation hyperplanes. A more subtle technology employs Gaussian distributions, and yet more subtle technologies vary the shape of the curve using polynomial fitting strategies. It should also be noted that attributes can be connected with functions other than just Boolean, and allowing for even more flexibility in class modeling.

As a rule of thumb, the more prior knowledge a method allows to be used, the more robust it is across the variety of images. This is especially true for images, which are as complex as the ones we are trying to analyze. Thus, the best techniques for classification in machine learning based, robust, generically applicable image analysis solution for HCS are neural or semantic networks.

3.2. Machine Learning for Segmentation

Once again, using our terminology, machine learning methods for segmentation must help us to find an optimal algorithm for segmentation of the provided training data. Before we look at examples, it is important to realize that there is a fundamental difference between classification and segmentation. Classification requires computing a classification result from attribute values, a process, which can be expressed in terms of inserting values into a mathematical function. Thus, improving the classification result by machine learning can be done by searching for an optimum for this mathematical function, a mathematical problem known by the name of variational calculus. Variational calculus is well understood and there are efficient algorithms, which solve the optimization problem (the backpropagation algorithm for neural networks being such an example). By contrast, measuring the quality of a segmentation algorithm requires executing it, and then comparing the produced (predicted) and actual segments. There is usually no closed expression in terms of a mathematical function by which a segmentation result can be computed. Thus, the problem of optimizing a segmentation algorithm can in most cases only be solved by an iterative, trial-and-error approach, which is usually much more computationally expensive. For this reason, machine learning for segmentation is not nearly as far developed as machine learning for classification. Nonetheless, there are some techniques, which apply to our problem.

Another important point to realize is that there are two different philosophies about segmentation. The first and conventional approach aims for a good segmentation result in one step, using a clever algorithm that does not depend on classification results. The second approach uses classification results to control the segmentation algorithm—this is the iterative strategy of segmentation and classification that we discussed earlier. This iterative approach requires some kind of bootstrapping: an initial segmentation must be performed to get a starting point. The segments of this bootstrapping step are often referred to as seed objects. The seeds then undergo an evolution and eventually some of them become the desired objects, whereas others might become helper objects (typically structures that are easily identified) that assist in detecting the desired objects. Because of their classification independence, the one-step segmentation algorithms are often used for the bootstrapping segmentation in the iterative approach.

The first and simplest one-step technique is segmentation by threshold. This is usually done on the basis of brightness, color, or texture, with a channel specific selection for color images. In our example on nucleus detection, all pixels with brightness greater than a given threshold in the Hoechst channel were grouped into objects, and these objects were then classified as nuclei. Because this technique depends on one parameter only, there are rather straightforward algorithms to automatically determine the threshold setting that produces the best segmentation results for the training data. One such algorithm works by lowering the threshold and thus increasing the object sizes until the average size of the produced (predicted) objects equals the average size of the actual objects. The resulting threshold is then the optimal threshold. If used as a bootstrapping algorithm, the threshold is usually set slightly less than the global maximum, so that for example the brightest 10% of all pixels are grouped into seed objects.

A related technique is searching for local extremes, again for brightness, color, or texture. This is often referred to as local thresholding. Every pixel that is only surrounded by darker pixels is a local brightness maximum. Each local maximum gets its own local threshold. All local thresholds are lowered by a certain percentage until again the average size of the produced objects equals the average size of the actual objects. Local thresholding can also be used to

generate seeds for an iterative approach. Apart from outstanding brightness in their centers, another strategy to detect objects is to look for clearly visible boundaries. This is known as edge detection. Because edges partition an image into segments, edge detection is also a one-step segmentation technique.

One of the best-known edge detection techniques is called the Watershed Algorithm. The intuitive idea underlying this technique comes from geography: it is that of a landscape or topographic relief which is flooded by water, watersheds being the dividing lines of the domains of attraction of rain falling over the region. As a result, the landscape is partitioned into domains of attraction separated by watersheds. If the height of a pixel is given by its intensity, then the Watershed Algorithm tends to create objects, which are separated by a visible contrast. This technique usually works reasonably well to separate cells, which in turn can produce good seed objects for nucleus detection (note that this would involve a shrinking evolutionary strategy for nucleus detection). Parameters of the Watershed Algorithm that a machine learning method might adjust are the minimal size of the catchment basins, the stepping height during flooding, and what to do with plateaus. In addition, there are a number of distance measures that can be employed which might lead to different results.

The Hough Transform is another method to detect curves through near-extreme values. It is usually employed for straight lines and circle segments. For straight lines, the idea is to transform the pixel space into a so-called Hough space. Each point in the hough space identifies a unique line in the original space, and the intensity of that point is the accumulated brightness along its corresponding line. Extreme points in Hough space then identify visible lines in the image. The hough transform for circles is sometimes a good method to detect nuclei, especially if the nuclear membrane is well visible.

The next technique is our first iterative approach, and it is know as region growing. It is usually guided by additional local and global attributes, such as size, shape, smoothness, compactness, and neighboring relations criteria and so on, all of which can be captured in a knowledge-based description for the desired objects of interest. The algorithm starts out from a set of seed objects determined by some bootstrapping criterion, for example, finding local maxima, and then continues to grow the seeds by adding new objects or pixels for as long as their classification likelihood improves. During this process the objects and thus their attributes continuously change, which makes it very complicated to optimize the segmentation algorithm. Because the local stopping criterion for each object is reaching a maximum for its classification likelihood, the "learning" strategy must first extract optimal class descriptions from the training data. This can be done using the approach described for semantic networks in the subsection on machine learning for classification. Then all objects must be grown and continuously reclassified. The growing process for each object is individually stopped when its classification likelihood starts to decrease. A simple method to avoid stopping in a local maximum is to continue the growing process a little further it to see if the classification likelihood increases again.

The opposite of region growing is region shrinking, a process that might be employed when starting with seed objects which are larger than the desired objects. Region growing and region shrinking are specific forms of the more general morphology operations dilation and erosion. There are also boundary optimization techniques, known as opening and closing. Opening is the sequence of an erosion step followed by a dilation step, whereas closing is the sequence of a dilation step followed by an erosion step. They result in making a boundary rougher or smoother, respectively. Again, these processes can be guided by knowledge-based descriptions of the desired objects derived from training data. More generally, all of these morphology operations can also be combined with one another (e.g., grow first and then smoothen the boundary), and be tailored for specific needs, such as growing linear structures (for neurite outgrowth), or jumping gaps and building bridges to deal with incomplete structures.

This most general form of an evolutionary strategy often includes the use of helper classes. Helper classes might not have anything to do with real structures. Thus, they are not available from the training data, so that we have no *a priori* knowledge about their attributes. Optimizing the iterative algorithm then involves testing it with a variety of attribute settings for the helper classes. An example would be to determine the maximal bridge length for a linear growth strategy, the bridge being a helper class to close gaps in the linear structure.

Finding the best settings for the helper classes' attributes can be done in a brute force way by computing results for a large number of settings, or more intelligently by an interval halving method (assuming a certain linear relation between the settings and the results). It can also be done by asking the user to provide a first guess for the settings.

In conclusion, it should be noted that there is no single optimal method for segmentation and classification to reliably detect all structures. Rather, a good method will be tailored to the structure of interest it is supposed to detect. A good image analysis package should thus contain such tailored methods for most structures of interest, and provide an easy to use framework for new method development to extend it to new structures of interest.

3.3. Training Techniques

A good image analysis package should minimize the user interaction required for training and make the process as user-friendly as possible. The best results are achieved by wizards or guides, which help the user to prepare the training data. Let us first review some of the important steps in preparing the training data, and then look at the art of selecting good training data.

Preparing training data requires manual delineation of the structures of interest. Any drawing software could be used to do this, but such an approach would be far from practical. A training set requires a significant amount of images. On each image there are many structures of interest. The outlines of these structures are very difficult to trace with a drawing tool. All of this would result in an unacceptable amount of work for the user to prepare the training data. Instead, a point-and-click strategy should be employed to select objects of interest, with effective and easy-to-use tools to optimize the outlines. This approach of training data preparation requires prior segmentation so that objects can be offered for selection. But how can this segmentation are performed without having access to the training data in the first place?

We have, thus, encountered another chicken-and-egg-type problem, which we solve again by an evolutionary strategy. We start out with a structure of interest that is relatively easy to detect, such as a nucleus. Then we use some of our one-step segmentation strategies to get a rough outline of this structure. We let the user select which strategy worked best, and then provide him with morphology tools to optimize the outlines. The only morphology tools he needs are dilation, erosion, opening, and closing. This will allow him to fix up most nuclei very quickly. The nuclei, which do not come out nicely will simply be rejected. This is not a problem, however, because we do not have to outline all nuclei on every training image. Instead, a good set will suffice. Then we repeat this process for other structures of interest, using segmentation strategies, which can now be guided by the structures we already detected in the previous steps. For example, detecting cells is more easier if the algorithm can use the already detected nuclei as seeds. This process, if nicely packaged into a wizard, provides a speedy and user-friendly way to prepare training data.

Now we turn to the art of selecting "good" training data. This has a big impact on how well our "learned" algorithms can extrapolate results from training data to previously unseen test data. The point is to select a "representative training set." What does this mean for our case? First, it refers to the training set size, which depends on the number of classes, or structures of interest that we want to distinguish. For each class type the system must be presented with an adequate number of samples. Second, it refers to picking the training data such that the entire range of each class is covered. This means that all varieties of a particular structure of interest should be present in the training data, and ideally more than once. If samples are collected statistically, then both

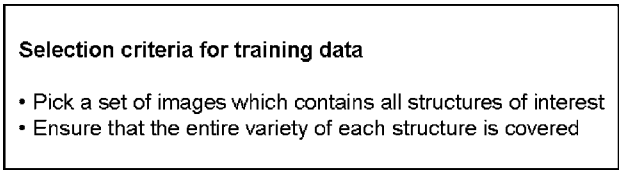

Fig. 6. Selection criteria for training data.

goals are achieved by just picking a large enough training set. There are statistical tools to compute how big the training set should be given the number of classes and their assumed distribution. But we are selecting the training data manually. Thus, a simple rule of thumb is that the training set needs to be characteristic of its usage in the real world. It is a very common mistake to pick only the best nuclei as training data, and then be surprised that only the best nuclei are detected, although most others are badly missed (*see* **Note 1**). All structures of interest vary significantly resulting from staining, biological, or other effects. A good selection includes a few samples from every variant of every class, thus covering the entire range of all classes reasonably well (*see* **Note 2**). This is summarized in **Fig. 6**.

4. Solutions

There are many image analysis packages for HCS on the market. For a good overview *see* ***(6)***. If you are a user who is just entering the field of HCS, or a user who uses HCS only for existing, standardized assays, then you are probably best served by existing turnkey solutions. The best known and most widely used turnkey solution is Cellomics' (http://www.cellomics.com/) ArrayScan system (http://www.cellomics.com/content/menu/ArrayScan%AE/). Other frequently used turnkey systems are offered by GE Healthcare (http://www.gehealthcare.com/), BD Biosciences (http://www.bdbiosciences.com), Evotec (http://www.evotech-technologies.com/), and Molecular Devices (http://www.moleculardevices.com/). These systems allow running frequently used assays, and the corresponding image analysis can be done almost at the push of a button. In particular, no detailed knowledge about image analysis is needed to operate these systems.

In contrast, if you are a user who is developing new assays, you will need a system with open, flexible image analysis capabilities. A more flexible approach is to offer a package of modules instead of prebuilt solutions. A module comprises a configurable algorithm for the detection of a specific structure of interest, for example, the nucleus or the cytoplasm (*see* **Note 3**). Solutions can be assembled by combining the modules, which detect the desired structures into an image analysis workflow. Modules can also be configured to fine- tune analysis to specific assay conditions, and ideally this step is supported by machine learning techniques. GE Healthcare's IN Cell Developer Toolbox (http://www.gehealthcare.com/company/pressroom/releases/pr_release_10287.html), Definiens' (http://www.definiens.com/) Cellenger HCS (http://www.definiens.com/news/ releases/19_e_Bioimage.htm) (*see* **Note 4**), and Evotec's Acapella (http://www.evotec-technologies.com/opencms/export/et/products/life_science_software/software_products/acapella.html) are such modular systems that have just or will soon become available. Definiens' Cellenger HCS includes wizard guided module selection and machine learning supported configuration (*see* **Note 5**). Cellomics, Inc. has also developed a number of its BioApplications in a much more modular way (http://www.cellomics.com/content/menu/BioApplications/). CellProfiler (http://groups.csail.mit.edu/vision/cellprofiler/), an effort at Whitehead and the Massachusetts Institute of Technology (MIT) to develop open-source modules for image analysis, which plug into MATLAB (http://www.mathworks.com/) as well as the open microscopy environment (OME) (http://www.openmicroscopy.org/), might become a choice for the budget-oriented user. All of these next generation, modular image analysis packages provide a significant step forward for the field of HCS. But there will always be structures for which no corresponding module yet exists.

For this reason some of the above-mentioned providers are making available the very techniques and tools which they use to develop their modules. In most cases, their technique is based on some scripting language, public or proprietary, and they have a development environment to assist the programmer. Sometimes additional algorithms for segmentation and classification developed in C++ or Java can be plugged into the package. This is obviously the most flexible approach. But it comes at a price: someone will need to invest a significant amount of time to get acquainted with image analysis in general and with the package and its programing language in particular. This is actually already true for the next generation, modular packages mentioned in the previous paragraph, though to a lesser degree.

If you want to be flexible with your assay development, and you need similar flexibility in image analysis, you will need an image analysis expert in your team. Computer vision is still miles away from human vision. Compared with cars, we are still in the age in which you needed to carefully adjust several levers and crank a handle to start the car, and you needed a flag person to walk in front of the car to warn people, all of which required an expert to operate the machine. Likewise, there is no image analysis push-button solution for the kind of flexibility you are asking for. You will need an expert to run your image analysis. And it will take him at least 3–6 mo of training until he will have an impact. But if you accept this you will be able to support your cutting edge research with cutting edge image analysis.

There is no doubt that we will sometime have push-button image analysis modules for (almost) any cellular or subcellular structure. Getting there will be greatly supported by packages which provide a comprehensive set of machine learning based algorithms for classification and segmentation as previously described. Until then the 80/20 rule applies: push-button solutions only for those 80% which run assays that are in widespread use, the remaining 20% which spearhead the field of HCS will need to pick their package of choice and develop their corresponding image analysis solution themselves (*see* **Note 6**).

5. Notes

During his time at Definiens the author observed several projects to develop customized image analysis solutions for a variety of assays using Definiens' development environment, Cellenger Developer Studio (http://www.definiens.com/cellenger/files/cellenger.pdf). Based on this experience as well as customer feedback, we will now take a look at some effort estimates which should serve as a guideline for your own lab.

1. Another rule for modules is that absolute numbers are to be avoided, and relative expressions with respect to some calibration or metadata should be used instead. For instance, using absolute brightness or size of a structure makes no sense, when fluctuations must be expected between assay types, during image acquisition, or during incubation.
2. When developing a solution for a particular assay, it is best to start with a rather small yet reasonably representative training set, and to focus on the easiest structures first. For a reasonably difficult cellular assay this step takes in the order of 1–4 wk. Afterwards it is important to validate and refine the solution using a larger training set, a process than can take another 1–8 wk, depending on the desired accuracy. Do not forget to separate your data into training and test data beforehand. Test data must never serve for training purposes. If it does, it will immediately become training data as well, and new test data must be obtained. This might sometimes be necessary if the training data is found not to be representative.
3. Developing a module to generically identify a particular structure of interest, for example, the nucleus or the cell membrane, is significantly harder. Our research has shown that it makes sense to build modules from submodules, with each submodule being capable of analyzing a particular subgroup of the structure. For instance, it would be too hard to develop one single strategy to identify nuclei for all stains, because Hoechst only stains the nucleus, but DRAQ5 also stains the cytoplasm. Instead, a submodule for each stain should be developed, and the metadata, which contains information on the used stain triggers the appropriate submodule.
4. Cellenger Developer Studio is an integrated development environment, which uses a graphical programing paradigm to assemble image analysis solutions. It supports Semantic Networks (i.e., a knowledge-based

description of structures of interest) as well as a list of classification and segmentation algorithms. Machine learning techniques are not included, although the graphical programing language is flexible enough to allow users to include machine-learning elements in their programs.

5. For Cellenger's graphical programing language, just as for any other programing language, the average training time for a new user to become productive was in the order of 3–6 mo, although it would take an additional 6 mo for the user to really master the system to a degree that he could also crack hard problems. Training comprises learning the language (theory) as well as working through examples (practice). Programing experience is quite useful though not a must. A solid theoretical understanding is important early on, whereas later building a repertoire of recipes from practical experience is critical.
6. Our last remark returns us to the importance of machine learning for the development of generically applicable modules. We found that without some form of machine learning assistance, for instance to adjust values of a region growing segmentation step, or to optimize a classification step, it was not possible to develop a generically usable module, not even for a structure as simple as a nucleus. The variation of nuclei in assays is simply too big. We had to include a step in which knowledge, automatically gathered from the training data, was fed into the module to automatically fine-tune the evolution of structures to the specific assay in question.

Acknowledgments

The author thanks Prof. Gerd Binnig for stimulating and enjoyable years of research during which a new framework for human understanding on computers, called Cognition Networks *(7–9)*, could be developed. This article is based mostly on experience gained from applying Cognition Networks to image analysis for biomedical images. Most of the work was carried out at IBM Research and Definiens.

References

1. Gonzalez, R. C. and Woods, R. E. (eds.) (2003) *Digital Image Processing*, Prentice Hall, Englewood Cliffs, NJ.
2. Gonzalez, R. C., Eddins, S. L., and Woods, R. E. (eds.) (2003) *Digital Image Processing Using MATLAB*, Prentice Hall, Englewood Cliffs, NJ.
3. Parker, J. R. (ed.) (1996) *Algorithms for Image Processing and Computer Vision*, John Wiley and Sons Inc., New York, NY.
4. Myler, H. R. and Weeks, A. R. (eds.) (1993) *The Pocket Handbook of Image Processing Algorithms in C*, Prentice Hall, Englewood Cliffs, NJ.
5. Seul, M., O'Gorman, L., and Sammon, M. J. (eds.) (2000) *Practical Algorithms for Image Analysis*, Cambridge University Press, Cambridge, UK.
6. Comley, J. (2005) High content screening. *Drug Discov. World* Summer 2005, 31–53.
7. Klenk, J., Binnig, G., and Schmidt, G. (2000) Handling complexity with self-organizing fractal semantic networks. *Emergence* **2(4),** 151–162.
8. Klenk, J., Binnig, G., and Bergan, E. (2001) Modeling knowledge and reasoning using locally active elements in semantic networks. In *Proceedings of ES2001, the Twenty-first SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence* (Bramer, M., Coenen, F., and Preece, A., eds., Springer, New York, NY.
9. Binnig, G., Baatz, M., Klenk, J., and Schmidt, G. (2002) Will machines start to think like humans? *Europhys. News* **33(2),** 44–47.