Computer Methods and Programs in Biomedicine (2005) 77, 99–113

ELSEVIER

Computer Methods and Programs in Biomedicine

www.intl.elsevierhealth.com/journals/cmpb

A Java application for tissue section image analysis

R. Kamalov[a], M. Guillaud[a], D. Haskins[a], A. Harrison[b], R. Kemp[b], D. Chiu[b], M. Follen[c,*], C. MacAulay[a]

[a] *Department of Cancer Imaging, BC Cancer Agency, Vancouver, BC, Canada*
[b] *Perceptronix Medical Inc., Vancouver, BC, Canada*
[c] *Department of Gynecologic Oncology, Center for Biomedical Engineering, Unit 193, University of Texas M.D. Anderson Cancer Center, 1515 Holcombe Blvd., Houston, TX 77030, USA*

Received 5 November 2003; received in revised form 21 April 2004; accepted 21 April 2004

KEYWORDS
Java;
Object-oriented;
Platform-independent;
Quantitative histopathology;
Quantitative cytology;
Image cytometry

Summary The medical industry has taken advantage of Java and Java technologies over the past few years, in large part due to the language's platform-independence and object-oriented structure. As such, Java provides powerful and effective tools for developing tissue section analysis software. The background and execution of this development are discussed in this publication. Object-oriented structure allows for the creation of ''Slide'', ''Unit'', and ''Cell'' objects to simulate the corresponding real-world objects. Different functions may then be created to perform various tasks on these objects, thus facilitating the development of the software package as a whole. At the current time, substantial parts of the initially planned functionality have been implemented. Getafics 1.0® is fully operational and currently supports a variety of research projects; however, there are certain features of the software that currently introduce unnecessary complexity and inefficiency. In the future, we hope to include features that obviate these problems.

1. Introduction

Many industries have taken significant advantages of Java and Java technologies during the last few years, and medicine is no exception [1]. Since Java is designed to be an object-oriented language from the ground up, writing software applications in Java means that one can benefit from object-oriented concepts [2]. Apart from this important aspect, there are other specific reasons why Java is our programming language of choice. Java is platform-independent, has a complete collection of GUI elements and supports Java Native Interface (JNI), web-services, multithreading, and connection with databases.

2. Background

There are two main reasons why Java is an ideal language with which to construct a tissue section analysis software package. First, Java is platform-independent. A platform simply refers to an underlying computer system on which various

*Corresponding author. Tel.: +1 713 745 2564; fax: +1 713 792 4856.
E-mail address: mfollen@mdanderson.org (M. Follen).

0169-2607/$ – see front matter © 2004 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.cmpb.2004.04.003

applications are run. Examples include UNIX, Mac OS X, Windows, and Linux. Platform-independence, then, indicates that a program developed on one platform may be executed on any other computer regardless of platform. From this, we see that Java proves powerful, because many different users (on several different platforms) may use the tissue section analysis software without the need for software modifications [3—5].

Secondly, Java is an object-oriented language. In essence, this means that the language simulates real-world objects. Rather than having raw code that is not easily decipherable in terms of functionality and semantics, object-oriented structures and syntax allow for facilitated understanding and manageable design. A simple example is the creation of a ''Cell'' object in the code of the software (to simulate a real-world cell). Rather than using complex code, object-oriented programming allows for the declaration of an object called ''Cell'', and consequently lets a user code in such a way that he or she may easily implement all the real-world functions and structures indicative of a cell. Like platform-independence, this proves extremely powerful, as we are coding for software that deals with tangible, real-world objects, not abstract concepts [3—5].

Much of screening in medicine involves obtaining cytologic (cell) samples. These samples are obtained by scraping cell surfaces of organ sites, such as the cervix, the mouth, or even the lungs. These samples are fixed in solutions, placed on slides, stained with chemicals, and viewed under the microscope by technologists and physicians with special training called cytopathologists. A great deal of expertise is required to distinguish the range of possible diagnoses, which include cells from normal to those with cancer. The chemical stains that are used by cytopathologists by convention for training have no mathematical logic. A Feulgin-thionin stain is a type of stain that is stoichiometrically related to the amount of DNA in the cell nucleus and therefore makes an ideal stain for a mathematical interpretation of a slide. Fig. 1 shows a Papanicolaou smear from the cervix stained with both the conventional and the Feulgin-thionin stains [6—8].

Computer-assisted image cytometry or computer reading of cells, attempts to sample the objects on the slide and subject them to mathematical analysis. In this case, the Cytosavant® measures 144 features (Table 1) on a cell-by-cell basis. Fig. 2 is a photograph of the Cytosavant®, which was used as a prototype for Getafics 1.0® the system discussed in this paper. The microscope is programmed to randomly sample 2000 cells on each slide. Statistical analyses can be used to derive algorithms (described elsewhere) that classify the cells. These algorithms, once validated, can classify the cells objectively without much human intervention. Figure shows a cell sample with Feulgen and Papanicolaou staining. Table 1 outlines the cytometry features that are mentioned [6—8].

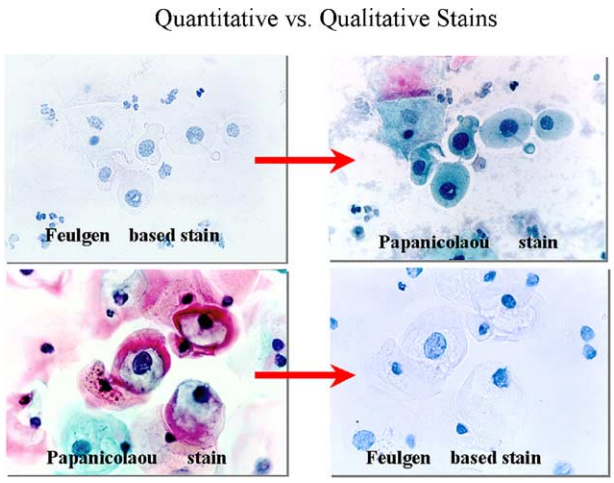


Fig. 1 Quantitative vs. qualitative stains. This figure demonstrates a cytologic sample of cells from the cervix. The top left panel shows that cells from the cervix can be stained with the Feulgen stain, and then, on the right panel, counterstained with the Papanicolaou stain. On the bottom left panel, the cells from the cervix are stained first with the Papanicolaou stain and then counterstained with the Feulgen stain. The Feulgin stain is a quantitative stain, while the Papanicolaou stain is a qualitative stain. The Papanicolaou stain is used by physicians to judge the amount of abnormality of the Pap smear, while the Feulgen stain is used by computer-assisted image cytometry for quantitative measurement.

Histopathology or tissue sections are obtained from biopsies (small pieces of tissue). These

Fig. 2 Cytosavant® is a computer-assisted image analysis system for the quantitative measurement of cytopathology and histopathology. The system consists of an autofocusing microscope, two computers, and an automatic slide loader.

Table 1 List of features used in the study

Category	Features
Cytometric features	
Morphometry (42)	
Size (3)	Area, mean_radius, and variance_radius
Shape (5)	Eccentricity, sphericity, elongation, compactness, and inertia_shape
Boundaries (34)	Low_freq_fft, freq_low_fft, and harm01—32_fft
Photometric (5)	DNA_index OD_max, OD_var, OD_skew, and OD_kurt
Discrete texture (24)	Low, medium, and high DNA amount Low, medium, and high DNA area Low, medium, high and medium_high DNA compactness Low, medium, high and medium_high DNA average distance Low, medium, medium_high, and high density object Low, medium, and high center mass Low_vs._medium, low_vs._high, and low_vs._medium-high DNA
Markovian texture (7)	Entropy, energy, contrast, correlation, homogeneity, cl_shade, and cl_prommence
Non-markovian texture (5)	Density_light_spots, density_dark_spots, center_of_gravity, range_extreme, and range_average
Fractal texture (3)	Fractal_areal, fractal_area 2, and fractal_dimension
Run length texture (10)	Short_runs_mean, and short_run_stdv Long_runs_mean, and long_run_stdv Gray_level_mean, and gray_level_stdv Run_length_mean, and run_length_stdv Run_percent_mean, and run_percent_stdv

biopsies provide a criterion standard for diagnosis. Like cytology, the sections are fixed, stained, cut, placed on slides, and read by pathologists. The computer-assisted image analysis of these sections is also performed using a Feulgen-thionin stain, but it is more complicated to identify either an area that is to be measured or a particular region of interest (ROI). Tissue sections cannot be randomly sampled; they must be carefully outlined by the histopathologist so that the ROI corresponds to the diagnostically important area. This is a labor-intensive process. Both the cytometric and architectural features in Table 1 apply to tissue sections. Fig. 3 shows the ROI and the measurement of tissue [6—8]. The software must accommodate the image as it is analyzed as a ''Slide-Unit-Cell-structure'' (Fig. 4). A hierarchical spatial model of this process is shown in Fig. 5.

3. Design considerations

The main goal of the system is to provide efficient tissue section analysis. It means, first of all, that the software system should be specialized and should reflect some of the most common peculiarities of a tissue section. This idea corresponds with an intensive approach of the system development concept. An extensive approach is focused on improvement of the quantitative properties of the system, such as development of new image processing algorithms, supporting new devices, and providing new services, etc. To maximize efficiency of the system we use both of these approaches to develop Getafics 1.0®.

Analysis software for histopathology image data tends to be expensive. Often it requires a specific hardware platform to run, communicates with specific external devices, and supports input and output data. As a result, substantial update fees are charged for any changes in an existing system, such as adding another external device or even correcting system bugs. The aim of this project is to implement a flexible and portable software tool for tissue section image processing which avoids the aforementioned problems. It needs to be available on all computer platforms and accept data in various formats from different sources. In addition, it must also include an interactive mode for general viewing and have an interface which will enable the addition of special-purpose image processing modules and external device modules. Currently the functionality of Getafics 1.0® is focused on the morphological, architectural, and statistical

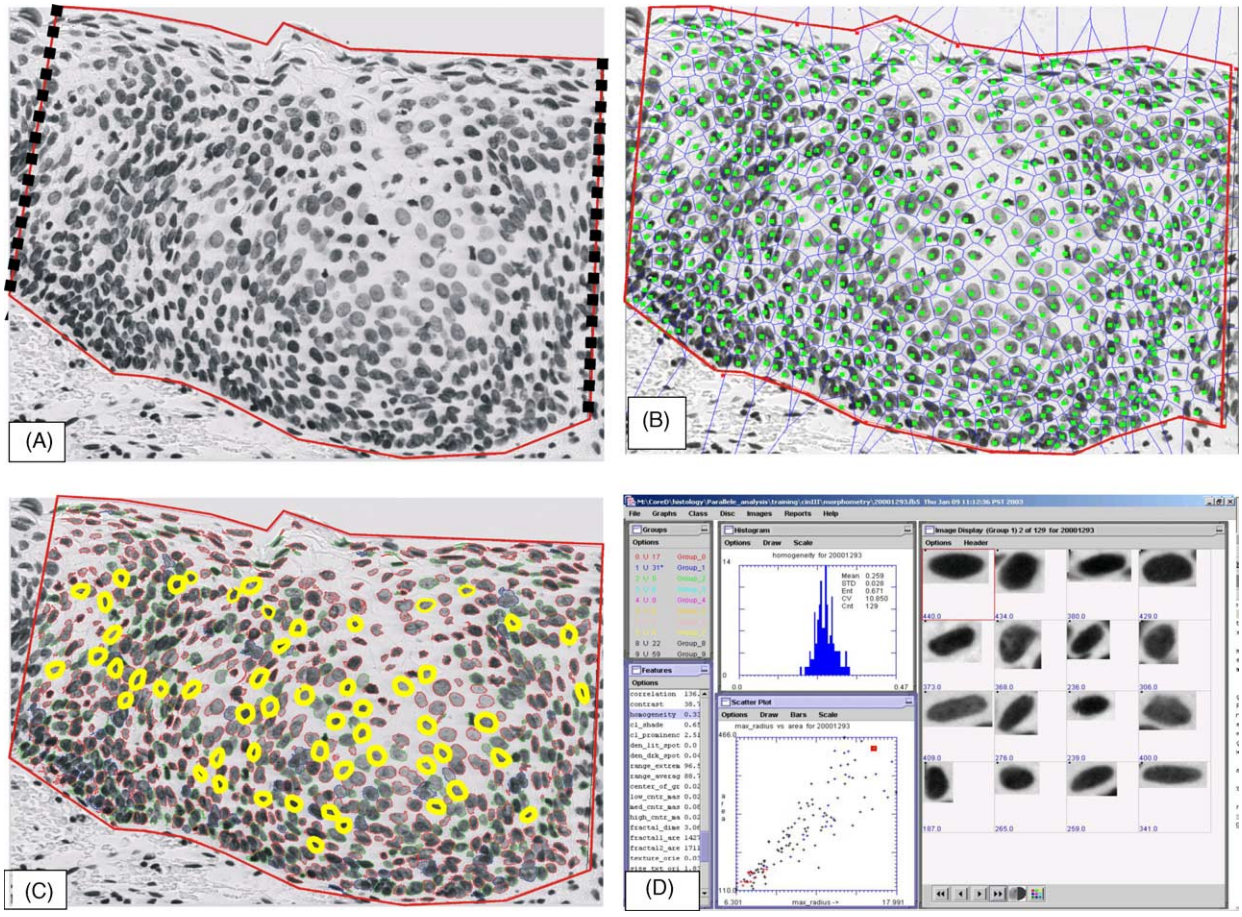


Fig. 3 Steps of the semi-automated analysis of cervical tissue lesions: (A) definition of the region of interest or the abnormal area to be studied; (B) automatic locations of the positions of the nuclei or the DNA containing area of the cell known to be abnormal in pre-cancer and cancerous cells; (C) automatic segmentation of the nuclei in the intermediate layers of the epithelium the area least disrupted in the fixation process; (D) snapshot of the interface of our software used for analyses and for quality control.

analysis of histopathology images [6—9]. However, the software design is general enough to allow the use of Getafics 1.0® for alternate areas of medical image processing.

4. System description

The reasons for developing Getafics as the generation software for histopathologic image analysis are:

- Experience accumulated from the long-term implementation of previous versions of the software suggested new ways to improve the efficiency of the system.
- Necessity to reorganize and update existing image processing algorithms.
- Continuous need to design and develop new algorithms on the basis of recent scientific investigations.
- Necessity to improve reliability and flexibility of the system using modern software technologies.

The following sections present background material and in-detail explanation of the system itself.

4.1. Object-oriented analysis

Object-oriented analysis is concerned with building a model of the problem to be solved. It addresses what software will do independently of how it will do it. The primary product of object-oriented analysis is a conceptual model of the problem that shows the proposed system and the real-world objects (samples, cameras, filters, processing, user, and database, etc.) with which the system interacts.

Conceptual models are usually constructed in three phases: defining the system requirements, identifying the entities that are involved in the

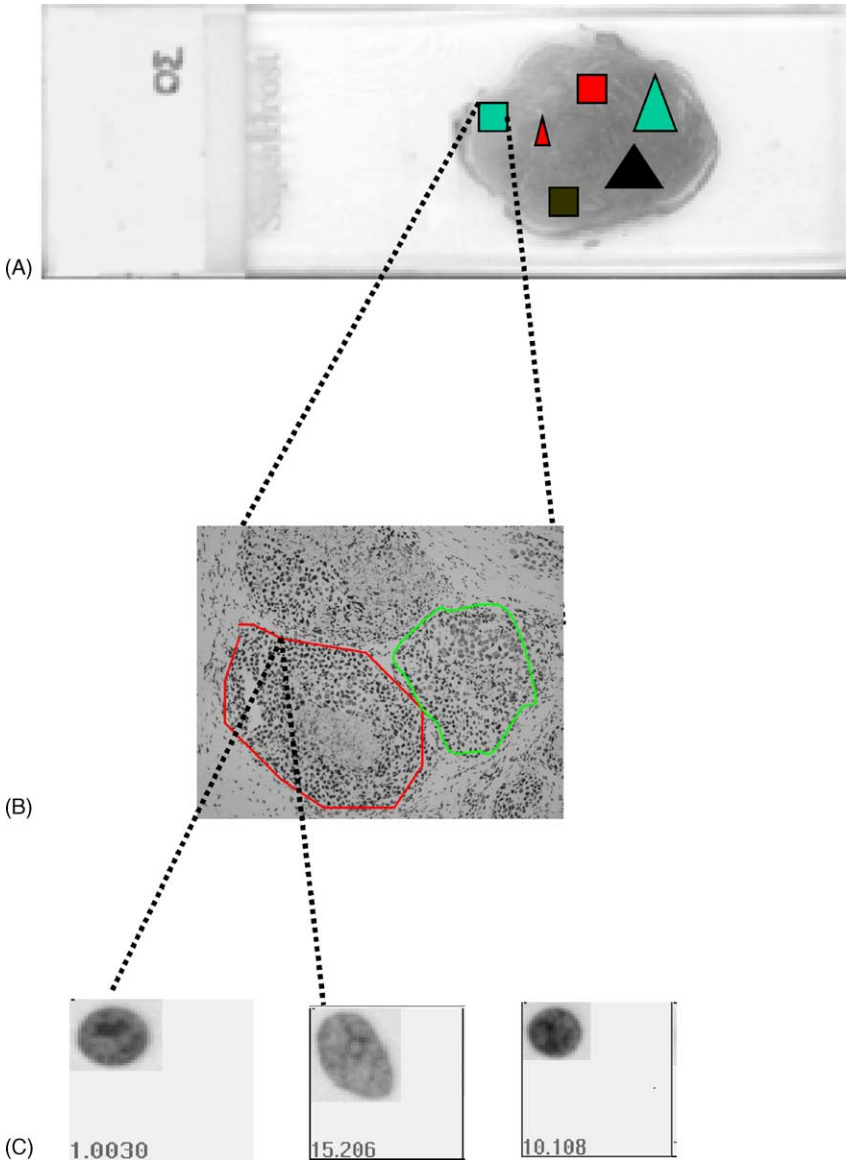


Fig. 4 ''Slide-Unit-Cell'' structure. This figure represents the hierarchical three-layer data structure and shows how the program moves from one level to the next. The ''Slide'' shows a section of tissue taken from a biopsy in a patient. The ''Unit'' layer shows a enlargement of the tissue at 10× magnification. The ''Cell'' layer shows the individual cells as they would be seen through the Cytosavant® at approximately 40× magnification.

problem, and identifying the relationships between entities.

4.1.1. System requirements

Our experience with the prototype system has enabled us to define the following top-level requirements for the Getafics 1.0® system:

- Specialization to reflect core peculiarities of a tissue section.
- Flexibility to provide effective data processing for different quantitative histopathological projects and different tissue structures.
- Platform-independence to minimize future deployment and maintenance costs.
- Openness structure to allow interaction with other systems, such as databases and web-services.

Fulfillment of these requirements ensures maximum efficiency and facilitation for current and future users.

4.1.2. Conceptual model of the system

The object-oriented approach allows one to consider the whole system as a group of objects that interact with each other in accordance to a set of

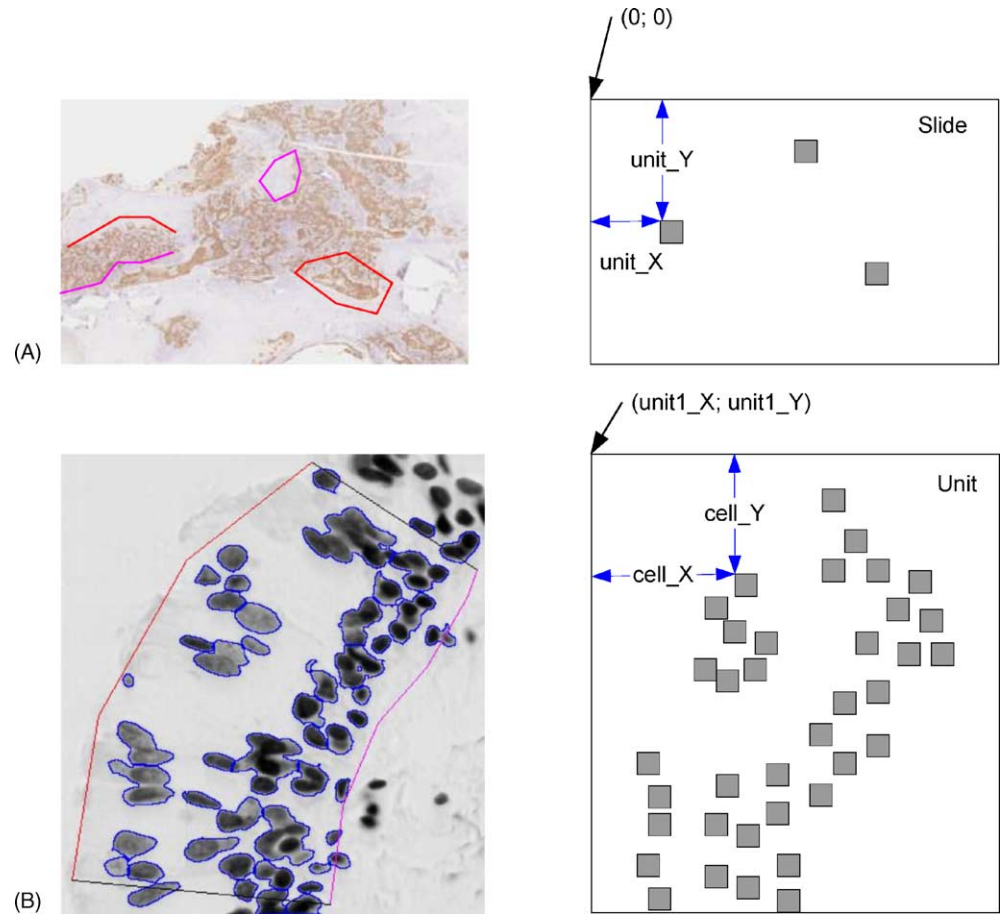


Fig. 5 Hierarchical spatial model. For the ''Slide'' layer, the model shows a 2-D distribution of the ''Units'' of tissue on the slide. For the ''Unit'' layer, the model represents a 2-D distribution of the cells in the tissue.

rules. Usually this group of entities can be separated into two main groups: data entities and activity entities (methods in object-oriented programming). The data entities are responsible for the storing of structured information, thus the efficiency of the whole system depends on the optimization of the data model. The activity entities process this information in order to obtain new information and are also responsible for event handling.

The conceptual model is a high-level view of the system based on the major functional components. One of the main requirements of the system design was to minimize interconnections between subsystems. A schematic model of the Getafics 1.0® system is presented in Fig. 6.

Examples of how the system functions are as follows. A working session begins with the initializing of the ''Project Manager'' component, which creates an execution profile for the specific application. This module performs user authentication and loads a default set of preferences for the user's selected project. If necessary, these preferences can be corrected manually by the user. The module then generates a specific configuration set-up for the system and stores this information in a special object denoted ''Status'', which is accessible by all the other modules during the working session.

Next the ''Hardware Manager'' subsystem interacts with external devices to provide image acquisition. It uses information from the ''Status'' object to set camera and automated microscope parameters. The output of the ''Hardware Manager'' module is a standard gray scale image frame, which is then transferred to the ''Unit Generator'' module, whose main function is the creation of a ''Unit'' object. This interactive module provides visualization of the original image frame and allows the user to define and process a ''region of interest''.

The output ''Unit'' object is then transferred to the ''Slide Monitor'' module, which governs all upper layers operations (''Add unit'' and ''Remove unit'', etc.) described in the previous section. This module is responsible for the processing of the ''Slide'' object and for interacting with the ''Data Manager'' module to provide ''Save'' and ''Load'' routines. The ''Slide'' object is constructed of

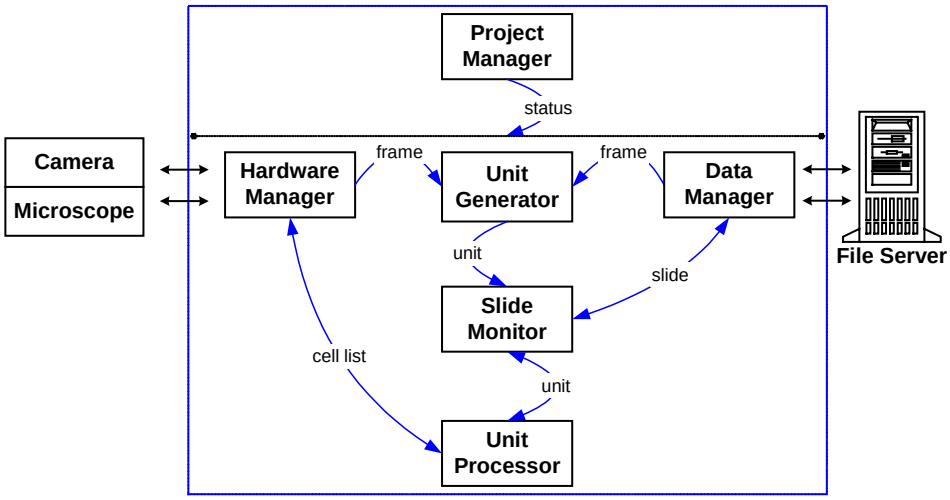


Fig. 6 Overview of software model. This figure shows an overview of the software model. This is a cartoon of how the objects shown in Fig. 2 and the features in Table 1 are related.

several ''Units'', but only one ''Unit'' may be active at a given time (i.e., prepared for data processing by the lower layer of the data model).

The ''Unit Processor'' module accepts the active ''Unit'' object from the ''Slide Monitor'' and generates a display of all completed and possible processing. After that, the object is available for further forms of analysis and processings, such as morphological, architectural, and statistical analysis.

The main function of the ''Data Manager'' module is to provide the necessary interactions with the ''File Server'' subsystem. These include naming, routing, testing consistency, and the saving and loading of input and output files. The main advantage of using ''Data Manager'' is functional separation of core modules of the system (''Unit Generator'' and ''Slide Monitor'') from the data sources. This approach guarantees flexibility of the system in the future. For example, it will simplify integration of the next version of the system with the database.

4.1.2.1. The data model. To develop Getafics 1.0®, we used three-layer hierarchical data structure denoted ''Slide-Unit-Cell'' (Fig. 4). The ''Cell'' object lies at the lowest layer of this structure while the ''Slide'' object is at the highest layer and represents a container for the full ensemble of ''Cells''. The ''Unit'' object is the core component of the data structure. Like the ''Slide'', it is a container but includes only a specific subset of ''Cells''. Usually the ''Cells'' are grouped into a ''Unit'' based on spatial locations, but in general, any of the ''Cell'' features can be used to create a specific ''Unit''. On the basis of a particular spatial distribution of cells common in tissue sections, three types of the ''Units'' were defined: ''Region'', ''Epithelium'', and ''Duct''.

All the elements of the hierarchical structure ''Slide-Unit-Cell'' are real-world objects and have regular spatial features, which can be described by a mathematical model (Fig. 5). For the ''Slide'' layer, this model represents a 2-D distribution of the ''Units'' objects. For the ''Unit'' layer, the same model represents a 2-D distribution of the ''Cells'' objects.

4.1.2.2. Object-oriented design. The ultimate goal of object-oriented design is to produce a detailed design of the classes that will provide the internal logic of the system. This is generally done by defining both the classes in the system's structure and the interactions between those classes.

4.2. Class diagram

Class diagrams express, in a general way, the static structure of a system. A class diagram does not express anything specific about the links of a given object, but it describes, in abstract, the potential links from an object to other objects (Fig. 7).

The structure of the class diagram is based on the well-known architecture ''model-view-controller''. Main information entities (''Dot'', ''Heap'', ''Membrane'', ''Cell'', ''Unit'', and ''Slide'') form a data skeleton of the ''model'' and activity entities represent ''view-controller'' block (Fig. 7).

''Dot'' is the simplest class, which underlies the information structure of the system. It represents a single 2-D point with several properties (active, selected, and focused, etc.). The ''Heap'' and ''Membrane'' classes are collections of ''Dots''

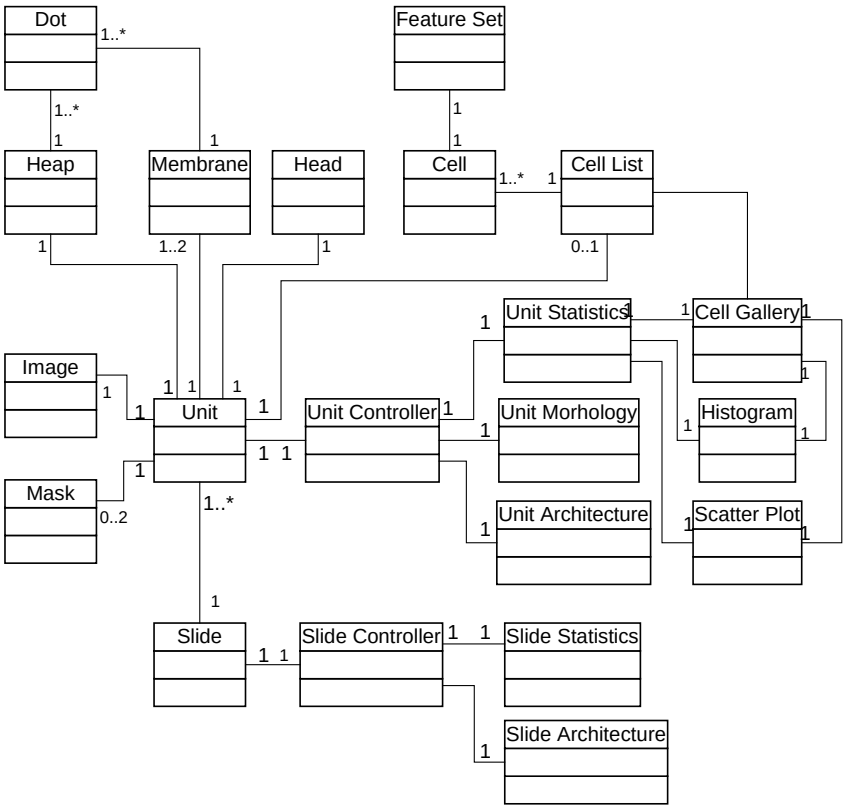


Fig. 7 Class diagram. This describes the potential links from objects to other objects in the software.

associated with a specified set of methods (Fig. 5). The ''Membrane'' class represents the boundary around a region of interest; therefore, there are two types of ''Membranes'': internal and external. The ''Heap'' class represents a collection of cells. The ''Head'' class combines and reflects ''Heap'' and ''Membrane'' information and consists of the following properties: type of structure, diagnosis, type of analysis, the number of vertices of both the internal and external membranes stored in the ''Membrane'' class, and the number of objects stored in the ''Heap'' class. The ''Unit Model'' represents a region of interest and encompasses all the information associated with that selected region. The ''Slide'' class operates as a container for multiple ''Units'' and provides simple operations like ''Add'', ''Remove'', ''Load'', and ''Save''.

The presentation layer is composed of eight main displays on view components, which provide for the visualization of the data. The ''Slide Model'' class is associated with two views: ''Statistics'' and ''Architecture''. The first presents quantitative information about the data, such as how many units are in a slide and how many cells are contained in each unit. The second presents a slide's spatial layout and deals with information about unit locations.

The ''Unit Model'' class is associated with three view components: ''Morphology'', ''Architecture'', and ''Statistics''. The ''Morphology'' view module provides basic image processing functions like segmentation, autofocus, interactive correction, and cell grouping, etc. The ''Architecture'' view module provides operations associated with the spatial coordinates of cells [9]. The ''Statistics'' view module consists of three sub-views (''Gallery'', Histogram'', and ''Scatter plot'') and represents the visualization of cell data and the operations performed on this data (Fig. 3).

4.3. Sequence diagram

The sequence diagram displays the interactions between objects from a temporal standpoint (Fig. 8). In this case, the context of each object is not represented explicitly. The representation focuses on expressing interactions based upon the temporal order of the message broadcast chronology.

The ''Unit Processor'' module is the most complicated part of the Getafics 1.0® system. The sequence diagram presented in Fig. 8 clarifies some details of this modules operation. The ''Slide Controller'' module defines a ''Unit'' for processing and informs the ''Unit Controller'' module by

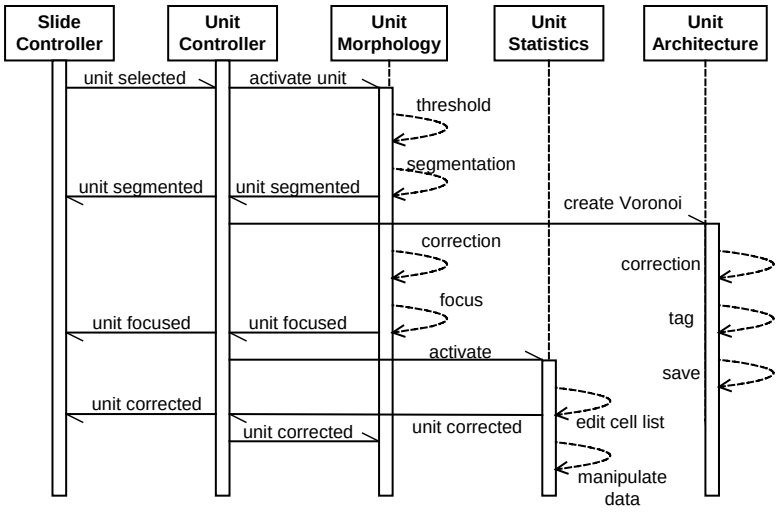


Fig. 8 Sequence diagram. This diagram shows how tissues are selected, measured, analyzed, subjected to statistical analysis, and how the output appears on the screen. This process encompasses what is shown in Fig. 3 and how it relates to Fig. 7 and this figure.

sending a message. The ''Unit Controller'' then accepts a ''Unit'' and sends the message ''Activate Unit'' to the ''Unit Morphology'' object. The ''Unit Morphology'' component then activates the selected ''Unit'' (i.e., makes it visible and available for processing). Then, depending on the type of project, a threshold value can be defined automatically or manually).

The next step after thresholding is segmentation. This routine can also be initialized automatically or manually [7]. When segmentation is completed, the ''Unit Morphology'' module sends the message ''the Unit Segmented'' to the ''Unit Controller'' module. Immediately after this event, the ''Unit Controller'' module sends the message ''Create Voronoi'' to the ''Unit Architecture'' module, which in turn generates an architectural model (''Voronoi diagram'') of the ''Unit'' object [9].

At this point, there are two active presentation layer objects, which independently provide morphological and architectural operations. In asynchronous mode, the ''Unit Architecture'' module provides ''Correction'', ''Tag'', and ''Save'' routines, while the ''Unit Morphology'' module provides ''Correction'' and ''Focus'' routines.

After the ''focus'' procedure, the ''Unit Morphology'' module sends the message ''Unit focused'' to the ''Unit Controller'' module. The message is then relayed to the ''Slide Controller'', and an ''activate'' event for the ''Unit Statistics'' module is generated. The ''Unit Statistics'' module initiates feature calculation for the collected ''Cell'' objects and provides visualization of the resultant data in different views, such as ''Gallery'', ''Histogram'', and ''Scatter plot'' (Fig. 3). Besides data presentation functions, the ''Unit Statistics'' module supports an ''Edit'' mode, which allows the user to correct selected ''Cell'' objects directly and interactively in ''Gallery''. After a correction event, the ''Unit Statistics'' module sends the special message ''Unit Corrected'' to the ''Unit Controller'' module, which in turn transfers it to the ''Slide Controller'' module and ''Unit Morphology'' module.

4.4. State diagram

The state diagram represents the state of the system from the perspective of allowed states and possible transitions (Fig. 9). The behavior of objects of a class may be described formally in terms of states and events, using a state machine connected to the class under consideration. Objects that do not have a very pronounced reactive behavior may be considered always to remain in the same state. In this case, their classes do not possess a state machine.

In accordance with the sequence diagram described in the previous section, the ''Unit Morphology'' module is an object which incorporates the following data processing. During these operations, the module utilizes the object ''Unit'' in two ways: as a source for input information and as a target for results. In other words, the ''Unit Morphology'' object reacts to the ''Unit'' object and changes its state (Fig. 9).

In its initial state, the ''Unit'' contains information about membrane locations and primary gray scale images. A ''Threshold'' event switches the ''Unit'' object to its second state where the

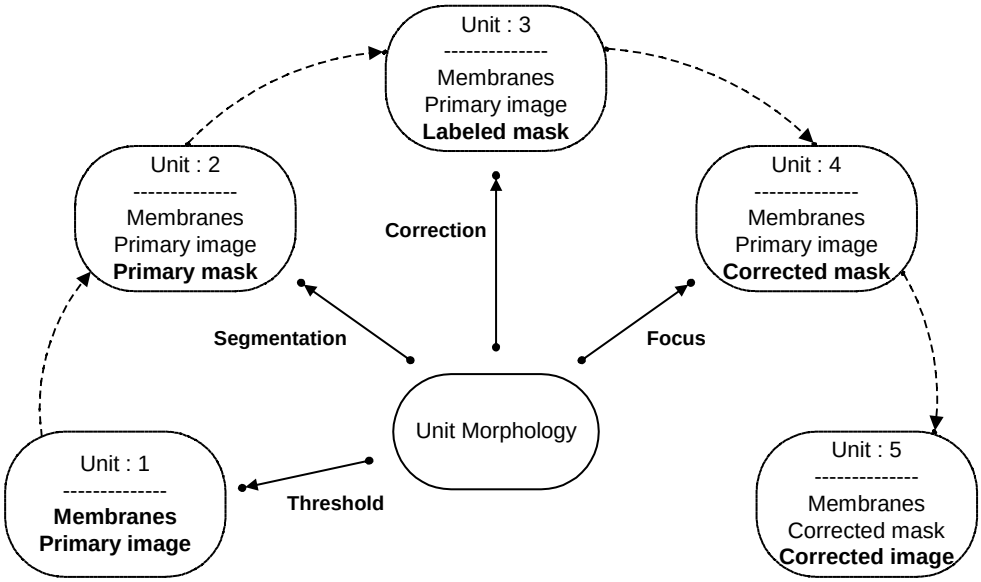


Fig. 9 State diagram. This represents the system from the perspective of allowed states and possible transitions in the software.

''Unit'' obtains a primary mask image using simple binarization of the gray scale image with selected threshold value. The next event (''Segmentation'') switches the ''Unit'' into its third state with a labeled mask. After a generalized interactive ''Correction'' event, the ''Unit'' moves into its next state with a corrected mask. After the final event (''Focus''), the ''Unit'' has a corrected segmented gray scale image as its fifth state.

The ''Unit Statistics'' module has a simple state diagram consisting only of two states: ''initial'' and ''corrected''. The correction function here is available only under the manual mode of the system.

4.5. Data processing

Every event described in the previous section consists of several data processing steps and can be initialized manually by a user and automatically by the program. All or some of these events may be combined into high-level events depending upon the automation category of the project.

Intensity thresholding is a widely used method of converting a gray scale image into a binary image [10]. In the simplest thresholding version, a single threshold level is established (either manually or automatically). All pixels whose brightness lies below this threshold are considered as the part of the objects of interest, and all others are considered as the background.

Getafics 1.0® provides different methods to set the threshold value. The simplest way is to use a default value defined by the supervisor project. Sometimes the user may wish to define this parameter manually, but usually, a threshold can be calculated automatically.

Image segmentation is the division of an image into different regions, each having certain properties [11]. In a segmented image, the elementary picture elements are no longer independent pixels, but rather a connected set of pixels all belonging to the same region. Once the image has been segmented, measurements can be performed on each region and neighboring relationships between adjacent regions can be investigated. Image segmentation, therefore, is a key step towards the quantitative interpretation of image data.

There are several different methods to allow for the correction of the mask after segmentation, all of which can be separated in two main groups: quantitative and qualitative. The first group operates on quantitative parameters of objects, for example, size and location. These methods allow improvement of data quality by deleting objects if any parameters of interest are out of a predefined range. The second group uses shape characteristics.

Qualitative correction can be represented by the ''Watershed'' procedure, which provides splitting of contacting objects [12]. Quantitative correction is represented by two-feature filtration procedures, which allow the rejection of objects based on area and/or location.

All of these correction procedures are available manually and automatically. In the automatic mode, the decision about object acceptance is made on the basis of the specific project setting, while in the manual mode, all decisions are made by the user. The ''Focus'' procedure is only

available in automatic mode. Information about selected group of cell objects is transferred from the ''Unit Processor'' module to the ''Hardware Manager'' module. The ''Hardware Manager'' sequentially changes *z*-coordinates of the stage and grabs additional 20-image frames. After this, the best focus image is selected for every cell from these 20-image frames. If this image is better than the original image, it is kept, and the module updates the appropriate structures.

4.6. User interface

The standard Java library contains a Swing package, which is a complete collection of GUI elements. With this package, powerful, and scaleable GUIs may be designed [13]. Getafics 1.0® can be classified as a user-centric application. It was designed on the basis of the three-layer data model (''Slide-Unit-Cell''), and user interface of the system provides access to the data from all of these layers.

At the ''Slide'' level, the user has two-visual components (''Slide Architecture'' and ''Slide Statistics'') which are always enabled and provide permanent dynamic control of quantity, spatial location, type (''Region'', ''Epithelium'', or ''Duct''), state (''Segmented'' or ''Not segmented''), and status (''Active'' or ''Passive'') of existing ''Units'' (Fig. 10). Also information about quantity of collected ''Cells'' is available. To create a new ''Unit'', a user has to switch the program to ''Slide'' mode by pressing the ''Slide View'' button. In this mode, the user can communicate with the automated microscope and camera, look through a slide, select a region of interest, and create a new ''Unit''. This new ''Unit'', by default, has status ''Active'', which means that it is prepared for visualization. According to the business logic of the system, only one active ''Unit'' is allowed at the moment. The user can move from the ''Slide'' level to the ''Unit'' level by pressing the ''Unit View'' button.

There are three powerful visual components at the ''Unit'' level: ''Unit Morphology'', ''Unit Architecture'', and ''Unit Statistics''. The ''Unit Morphology'' view is presented in Fig. 10a. Initially, after component initialization, the user can see a primary gray scale image of the selected unit. Then, using tool bar buttons or popup menu commands, the user can execute different routines, such as threshold, segmentation, correction, and autofocus, etc. The threshold value is adjustable and can be set manually or automatically in a special window (Fig. 6). After segmentation, the resulting ''mask'' and ''contour'' images are available for visualization and correction. Four informational counters provide the following statistical information: total quantity of cells, selected cells, focused cells, and active cells.

The ''Unit Architecture'' module view is presented in Fig. 10b. The core activity here is the calculation of the Voronoi diagram. Initially, when segmentation is complete, the Voronoi diagram is calculated automatically. Subsequently, it is recalculated after every correction event. Besides manual correction, a ''Tag'' routine is available in this window to separate cells in different groups. A special button allows the user to initialize a ''VORSTAIN'' routine [14] which provides re-segmentation of the active ''Unit'' object on the basis of the Voronoi diagram. The resulting mask is stored in the ''Unit'' object and can be used, if necessary, as a mathematical-statistical model of the origin mask to minimize temporal processing costs during the next steps of data processing.

The ''Unit Statistics'' view component is the most complex because it consists of three subcomponents: ''Gallery'', ''Histogram'', and ''Scatter plot'' (Fig. 10c). The ''Gallery'' operates as a container for ''Cell'' objects, which can be displayed the different views, such as: gray-scale image, mask, or contour. Additionally, every ''Cell'' object is associated with the feature set, which can be displayed in the ''Gallery''. The user can easily list, sort, filter and zoom several ''Cells'' or only a selected group of ''Cells''. The ''Gallery'' module also supports group operations, such as ''Delete'', ''Move'', ''Lock'', and ''Hide''. The powerful ''Cell Edit'' routine provides the user with possibility to correct a selected ''Cell'' object directly from the ''Gallery'' module.

The ''Histogram'' module displays a histogram distribution of a selected ''Cell'' feature. There are three different view modes: histograms for the whole ensemble of ''Cells'', for a selected group of ''Cells'', or a display of both histograms simultaneously. Besides graphical presentation of the selected feature distribution, the ''Histogram'' module provides the user with the main statistical data, such as total objects quantity, minimum, maximum, mean, standard deviation, and coefficient of variation.

The ''Scatter plot'' module displays distribution of any two ''Cell'' features selected by the user in the two-dimensional metrics. There are two available view modes: distribution of a groups of ''Cell'' objects and distribution of a selected group of ''Cells''. Both of the described modules (''Histogram'' and ''Scatter plot'') support automatic scale adjustment and axes recalculation.

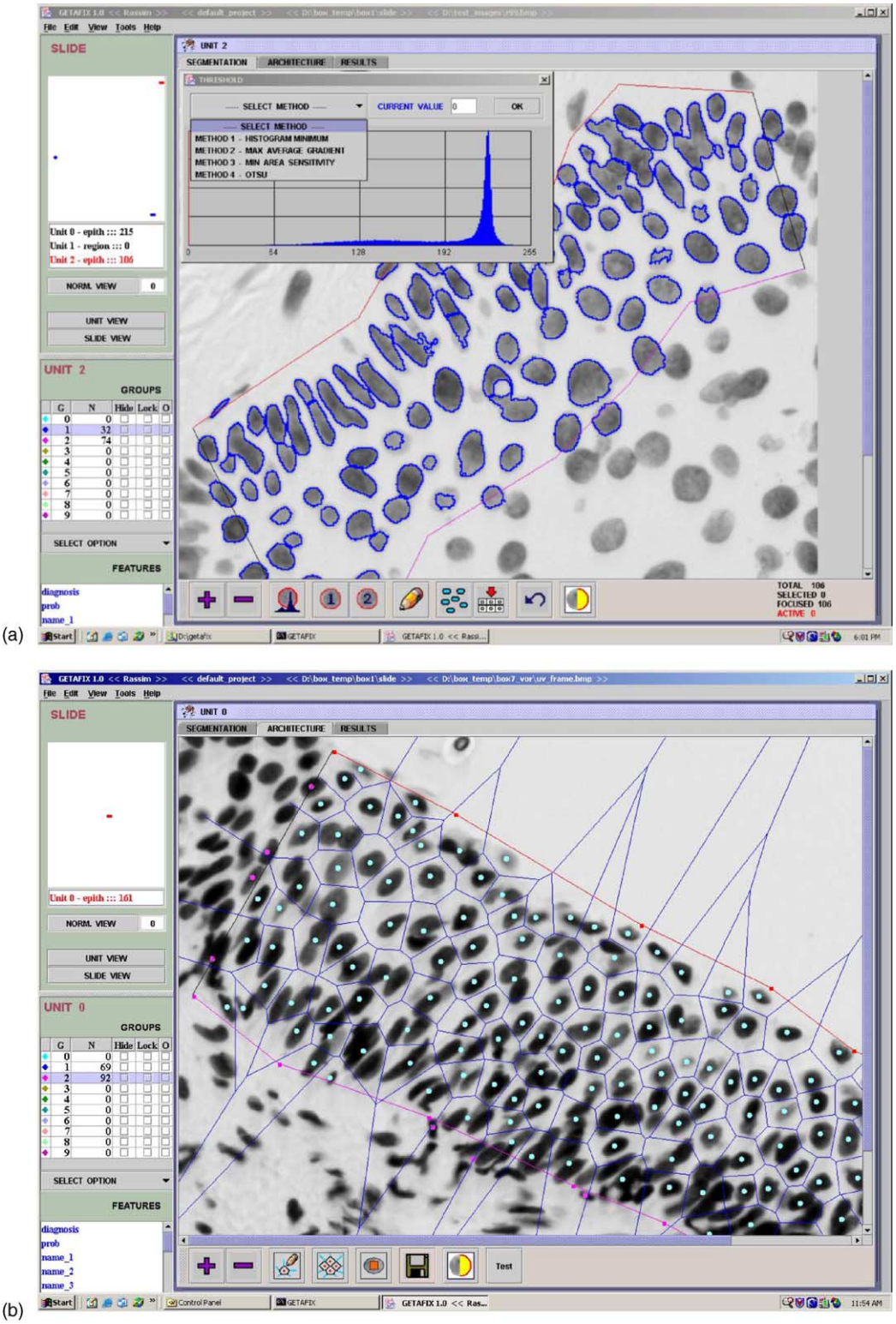
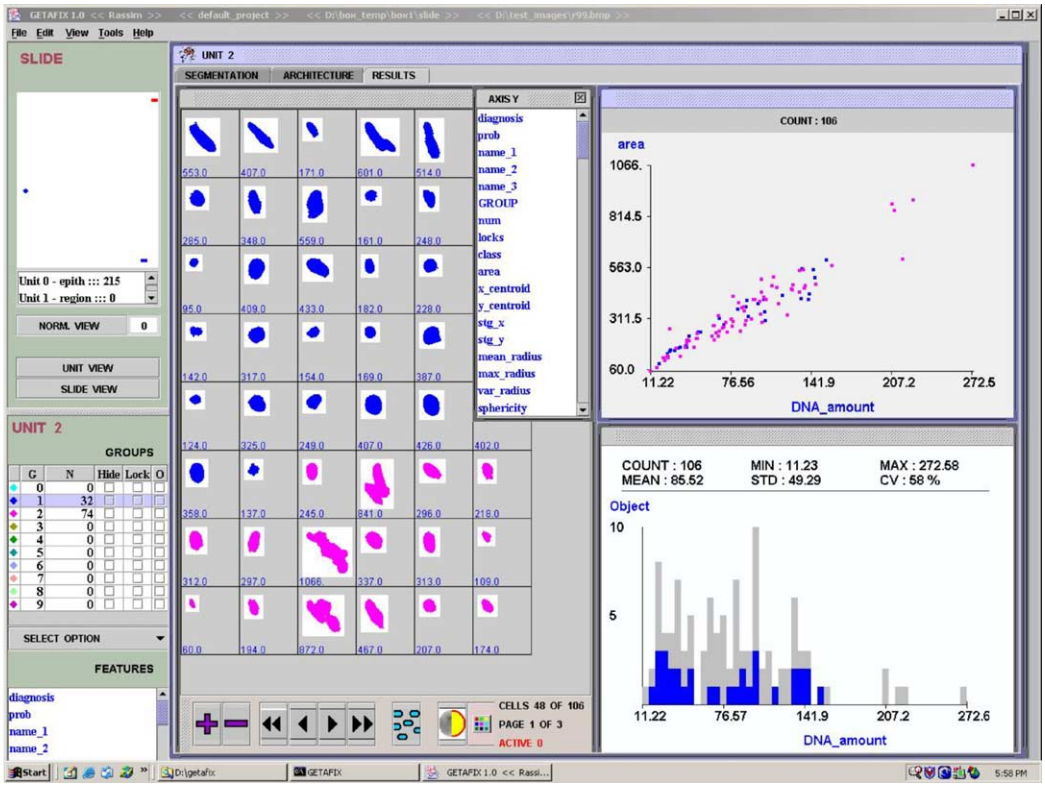


Fig. 10 (a) Unit morphology. This shows a small section of cervical tissue. The cells have been selected and are ready for the execution of different routines, such as thresholding, segmentation, correction, and analysis. (b) Unit architecture. It shows a portion of tissue which has been analyzed using Voronoi diagrams. Special analyses are done allowing us to analyze the spacing of these polygons. The spacing tells us if the cells are uniformally separated (as they would be if they were grown normally) or if they are nonuniformally separated (as they are in precancerous and cancerous states); (c) unit statistics. This component has three sub-components, including gallery, histogram, and scatter plot. Data may be exported to other statistical programs for analysis.



(c)

Fig. 10 (*Continued*).

Also, some special group operations are available, for example, ''Select object'', ''Select group of objects'', ''Move selected group'', and ''Delete selected group'', etc.

4.7. Hardware and software tools

The computers we used for developing and testing the Getafics 1.0® system are standard Pentium III models with 512 MB of memory. Windows NT and Windows 2000 were used as operating systems during development cycle, while the OS/2 operating system was used for portability testing. We used the Retiga 1300 (QImaging) monochrome camera with 1280 $\times$ 1024 resolution and 6.7-μm pixel sizes. Interface: IEEE 1394 (FireWire).

The software tools used to develop the application were the Java 2 Platform, Standard Edition (J2SE), which is the standard package provided by Sun Microsystems, and Visual Age for Java1.4 provided by IBM, which is an integrated, visual development environment that supports the complete cycle of Java program development.

5. Status report

The software is fully operational and implemented on three devices at the Vancouver site and one device at the Houston site. Getafics 1.0® is fully operational and currently supports the ''Lang Immune'' research study, in which several thousand lung biopsies are being processed. We are also using Getafics 1.0® for the analysis of cervical samples in a Program Project in which 2000 Papanicolaou smears and 5000 biopsies will be obtained.

5.1. Object-oriented architecture

Developed on the basis of object-oriented technology, Getafics 1.0® is a universal framework that will be applied across an entire line of projects under development in our department. One of the most important advantages of Getafics 1.0® is that it was developed as a set of subsystems, each of which is responsible for a specific function. This approach provides great flexibility and expandability.

5.2. Data model reflects tissue section

The core data model of the system (''Slide-Unit-Cell'' hierarchical triad) reflects the biological and spatial structure of the tissue section. This approach allows us to use the developed application for tissue section analysis in two ''directions'': ''Horizontal'', which takes into account biological and spatial relations between objects on the same layer, and ''Vertical'', which deals with biologi-

cal and spatial relations between objects on the different layers.

5.3. Variety supported projects

Getafics 1.0® implements many algorithms designed for histopathology image analysis. These algorithms include image acquisition and registration, filtering and enhancement, objects detection and correction, data classification, and statistical analysis. As an integrated part of a comprehensive application, these algorithms can easily be combined in specific processing chains in accordance with the specifications of a project to be performed.

5.4. Automated mode

An automated mode would not only speed up the processing of tissue section images, but would also allow new questions to be posted in the area of medical image analysis. The ability to automatically process tissue section images will allow us to expand the scope of investigations and better understand the precision and accuracy of current analysis methods.

5.5. Variety input formats

An important advantage of the Getafics 1.0® system is the variety of input data acceptable for processing. It could be a standard image frame grabbed from a camera or an image file downloaded from a database in GIF, BMP, JPEG, or TIFF format. The system also allows for the saving and reloading of data at any intermediate phase of the full image processing cycle. For example, a user who has created ''Units'' may save all necessary data as a ''Slide'' object, and that data could later be retrieved by a second user who carries out all other necessary steps.

This status report shows the great amount of progress that has been made in the development of tissue section analysis software, but there are still improvements to be made.

6. Future plans

There is a general perception that Java programs are slow, because many people assume that if program is not compiled, it must be slow. Part of this perception is based on reality. Many early applets and applications were slow, because of non-optimized versions of the Java Virtual Machine (JVM) and the overhead of a new language. Now, Java technologies have progressed to the point where a Java application is not particularly handicapped. With good design and coding practices, applications usually run fast enough [15]. Sometimes the performance benefits of native code outweigh platform-independence offered by Java. In this case, high-performance just-in-time (JIT) compilers can help.

Another important point is data management. At the present moment, Getafics 1.0® uses a ''flat file'' approach, in which all of the information about an application is written to the disk without internal pointers or indexes. This approach can introduce unnecessary complexity and inefficiency when used in a sophisticated application like Getafics 1.0®. When the application must capture information about more than one entity, a Relational Database Management System (RDBMS) is a better choice than a ''flat file''. Sun offers a package (java.sql) that allows Java programs to access RDBMS through a Java Database Connectivity (JDBC) package and interact with the database by using standard SQL language. We hope to include these features in the future versions of Getafics 1.0®.

7. Conclusion

The successful development of the Getafics 1.0® as a portable, object-oriented, and extensible software system illustrates the effectiveness of Java as an application development environment for biomedical research. The core structural component of the system-hierarchical data model 'Slide-Unit-Cell'' reflects the biological and spatial structure of the tissue section and allows for multidimensional tissue analysis. This fact, in combination with the advantages of Java technology, allowed us to create an efficient software tool for a wide range of histopathologic projects.

References

[1] Sun Microsystems Inc., The source for Java technology (http://java.sun.com).
[2] G. Booch, Object Solutions: Managing the Object-Oriented Project, Addison-Wesley, Menlo Park, CA, 1996.
[3] J.A. Borror, Java Principles of Object-Oriented Programming, R&D Books, Lawrence, Kansas, 1999.
[4] B. Simpson, J. Mitchell, B. Christeson, R. Zaidi, J. Levine, Making Sense of Java, Manning Publications Co., Greenwich, CT, 1996.
[5] R. Decker, S. Hirshfield, An Introduction to Programming Using Java, second ed., Brooks/Cole, Pacific Grove, CA, 2000.

[6] M. Doudkine, N. Poulin, B. Palcic, Nuclear texture measurements in image cytometry, Pathologica 87 (1995) 286—299.
[7] C. MacAulay, B. Palcic, An edge relocation segmentation algorithm, Anal. Quant. Cytol. Histol. 12 (1990) 165—171.
[8] N. Poulin, I. Boiko, C. MacAulay, C. Boone, K. Nishioka, W. Hittleman, M.F. Mitchell, Nuclear morphometry as an intermediate endpoint biomarker in chemoprevention of cervical carcinoma using alpha-difluoromethylornithine, Cytometry 38 (5) (1999) 214—233.
[9] M. Guillaud, R. Marcelpoil, P. Payne, C. MacAulay, S. Lam, B. Palcic, Multi-scale cellular sociology analysis: application to the bronchial epithelium, in: International Workshop on Information Processing in Cells and Tissues, Liverpool, England, 1995, pp. 557—568.
[10] J. C. Russ, The Image Processing Handbook, CRC Press, Boca Raton, FL, 1992.
[11] P. Soille, Morphological Image Analysis, Springer, Germany, 1999.
[12] R. Kemp, Segmentation of Overlapping Nuclei in Lung Epithelial Biopsy Sections, The University of British Columbia, 2001.
[13] J. Zukowski, Definitive Guide to Swing for Java 2, APress, Emeryville, CA, 1999.
[14] M. Guillaud, J.B. Matthews, A. Harrison, C. MacAulay, K. Skov, A novel image cytometric method for quantization of immunohistochemical staining of cytoplasmic antigens, Anal. Cell. Pathol. 14 (1997) 87—89.
[15] J. Shirazi, Java Performance Tuning, O'Reilly, Sebastopol, CA, 2000.