CELLO

An Interactive System for Image Analysis

Bengtsson E, Ph.D[1], Eriksson O, B.Sc[2], Jarkrans T, M.E[2] ,
Nordin B, B.Sc[2] and Stenkvist B, M.D[3]

(1) Dept. of Physics, Uppsala University, Sweden
(2) Dept. of Computer Science, Uppsala University
(3) Dept. of Clin. Cytology, Uppsala University Hospital

ABSTRACT

CELLO is an interactive, command oriented image analysis system. It has been developed for applications in automated cytology i.e. for the development of microscopic cell image analysis algorithms but is very flexible and can be used for a wide range of image analysis applications.

The system allows interactive as well as batch processing. In fact the degree of user interaction is easily adapted to the users needs. The image processing language is command oriented and is easy to expand by adding new commands. This is mainly due to the fact that CELLO has a high degree of modularity. Each command has its own independently loadable program module. The system monitor itself also consists of several relatively independent modules.

CELLO has been implemented on a large mini computer, a PDP 11/55 equipped with a custom-built image display system. The display interface of the software is general so that almost any type of display terminal can be used. A simpler CELLO version that runs on a PDP 8 computer has also been developed.

INTRODUCTION

The development of image processing algorithms is a highly experimental undertaking. Different tentative algorithms are applied to an image, the result is checked, the algorithm is modified and applied again until an acceptable result is obtained. To facilitate this kind of work a highly interactive system is needed.

On the other hand, when a promising algorithm has been found, it is desirable to apply it to a large number of different images in order to obtain reliable statistics about its performance. At that time it is only cumbersome if each step requires human interaction. Thus a system with a variable degree of interactivity is desirable.

Most image processing systems are heavily oriented towards image processing in a narrow sense i.e. transforming one image into other images. This certainly is a necessary facility in all image processing systems. To support this facility one needs a powerful way of specifying what operations should be performed. In order to keep the execution times reasonable a high performance computer is desirable. Such computers can very well take advantage of the parallell nature of most image processing operations.

However, many applications require a few scalar features, extracted from the images, as the end result of the algorithms. Thus for an application oriented system it is equally important to have good facilities for feature extraction and convenient handling of the extracted data in some kind of database. A system that is designed with both these aspects in mind can perhaps be called an image analysis system as opposed to a pure image processing system.

For a number of years research in the field of cytology auto-

mation has been carried out at the Department of Clinical Cytology, Uppsala University Hospital. Problems concerning automation of the uterine cancer cytology have been studied as well as more basic studies in computerized nuclear morphometry in relation to epidemiological and clinical aspects of breast cancer [1,2,3,4].

A number of interactive image analysis systems are continuously being developed in order to facilitate this research work. The first of these, SCANCANS [5], was a simple, easy to use, command driven software system implemented on a PDP-8 interfaced to a Leitz MPV II microphotometer, using a Tektronix 4010 storage tube terminal as the operator interface. This was a very limited system, although exciting enough to initiate the development of a new more powerful interactive software system on the PDP-8, basically using the same hardware. The new system, which was called CELLO-8, had a richer syntax and was highly modular in its structure [6]. It could be used in interactive as well as in batch mode, and could in fact be easily adapted to any level of interactivity. The success of this system was essentially the result of modern programming technology.

A PDP-8, however, is not an ideal computer for image processing, mainly due to the fact that it is a relatively old and slow computer and that the word length is 12 bits only. When a large modern mini-computer, a PDP-11/55, became available to us we therefore developed a third generation of our interactive image analysis system giving it the name CELLO-11. The basic system structure and command syntax is similar to that of the predecessor. However, the new system accepts several simultaneous users and also takes advantage of the greater flexibility of the larger computer system in several other ways. This paper discusses some of the design considerations that went into the system, outlines the structure of the system and gives some examples of its use. In a final section some of the experiences from using the system are reported and a new fourth generation system which presently is under implementation is outlined.

DESIGN CONSIDERATIONS

Although generally applicable in the field of image analysis, the CELLO system was developed for applications in automated cytology. The system was designed to fulfill two different needs:

1. A flexible interactive system for measuring and evaluating numerical parameters from cell nuclei and cytoplasms

2. A platform for algorithm and method development in automation of uterine cancer cytology

Both these applications require that not only image processing operations can be performed on the cells, but also that the results from such operations can be stored in data bases and that tools are available for evaluation of the data bases. They also require the possibility of varying the level of user interaction from a step by step control of each single operation to automatic batch runs that generate large sets of data which subsequently can be statistically analyzed.

These requirements were translated into the following design goals:

1. The user should be able to interact with the system by means of free format commands typed on the keyboard of a text terminal. This implies that the system has to scrutinize the input text string carefully for errors before any processing starts and write out suitable diagnostics if anything is wrong.

2. It should be possible to put commands together to sequences, to be used as procedures or new powerful commands (super commands). This feature enables the system manager to define commands suitable for use by unexperienced operators and non programmers, e.g. biologists and pathologists, in well defined studies of various kinds.

3. It should be possible to automatically repeat the same sequence of commands for a whole set of different images and save the result in a data base. This introduces a batch facility in the system.

4. The design had to be completely open-ended, i.e. it should be possible to add new functions to the system without having to modify anything in the old system. This was considered to be absolutely necessary, as the system was expected to grow more or less continuously during its lifetime in terms of new commands for new applications.

HARDWARE

Computer system

The CELLO-11 system was implemented on a PDP-11/55 computer running under the RSX-11/M multi-user operating system. The basic characteristics of the computer system are: 256 k bytes of core memory, an 8 lines multiplexed terminal interface, two 40 M bytes disk memories, an 800 bpi magnetic tape drive, a floating point processor and a Versatec electrostatic printer/plotter. The system also has a link to the old PDP-8 based image processing system (see Figure 1).

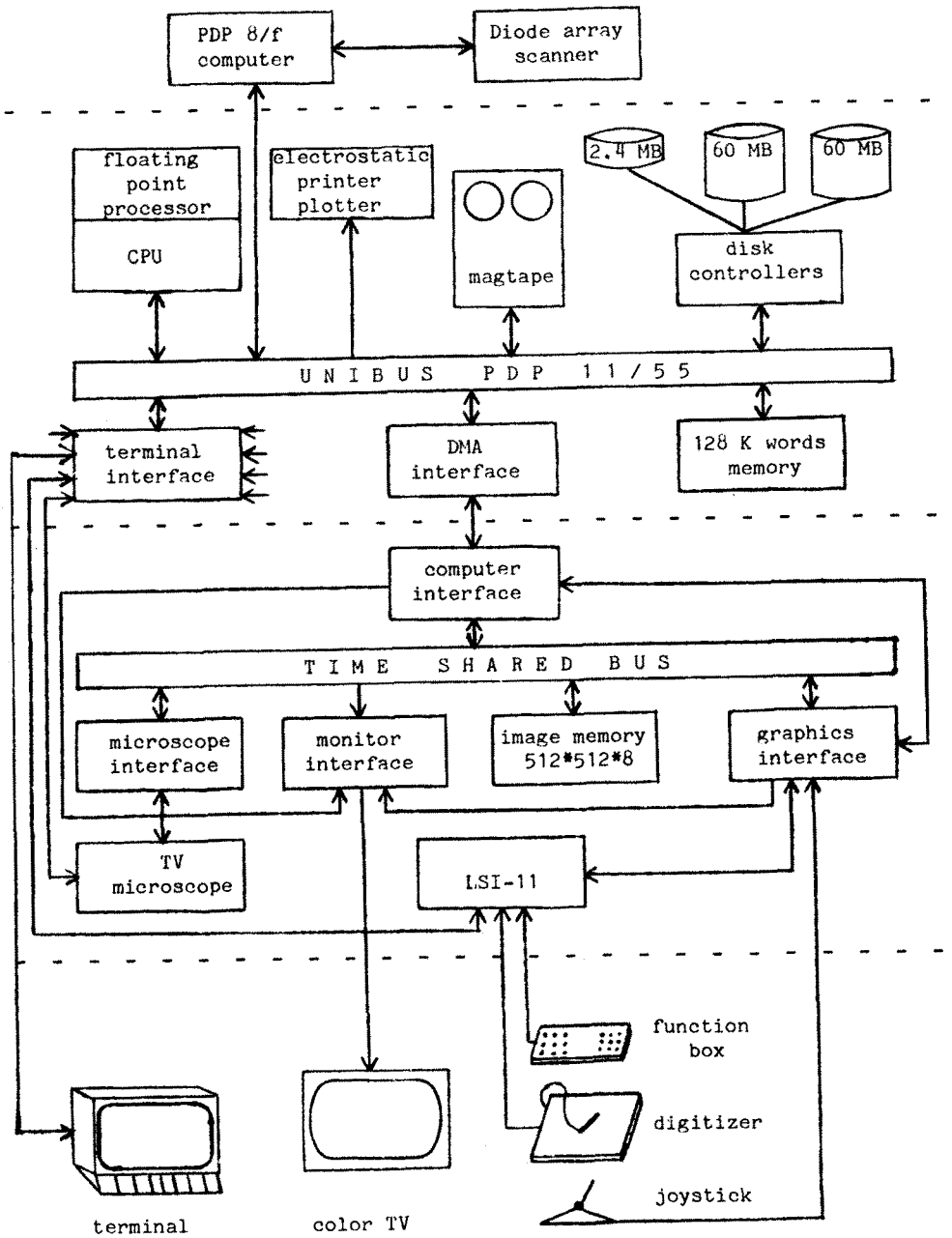

Figure 1. Block diagram of the hardware. The horisontal dashed lines separate the different subsystems. These are the image scanner, the host computer, the image display system with the TV-microscope and the interactive workstation.

Terminals

Three alphanumerical terminals are connected to the system as well as three graphics terminals (Tektronix 4010, Tektronix 4006 and Hewlett-Packard 2648A). A custom-built image display system connected via a DMA channel is also available.

CELLO can be run from any terminal but a graphics terminal is needed for most image processing operations since the user otherwise will have very limited possibilities of interacting with the images. The simple graphics terminals are well suited for presenting graphical information such as histograms and object outlines but can also be used for grayscale images through the use of dithering techniques [7] i.e. each picture element is represented by a small matrix of black or white points. The grayscale obtainable in this manner is limited to a few gray levels but this is sufficient for many purposes. In fact, CELLO was originally developed for simple graphics terminals and is still frequently run from such terminals. However, an image display system capable of displaying a full 6 bit grayscale as well as graphics overlays in color is a more ideal tool for interactive image processing work. Such a system has also been built for us by the Department of Electrical Engineering at Linkoeping University and integrated with the CELLO software. It is briefly described in the next section. For a more complete description see [8,9]

Image display system

The image display system is built up around a high speed time shared data bus. The capacity of the bus is more than 40 M byte/sec. An image memory with a capacity of 512*512 picture points, each with 8 bits, and eight general I/O ports are connected to the bus. Three of these I/O ports are presently occupied, namely by a display processor, a microcomputer controlled graphics generator and an interface to the host computer.

The display processor is used for generating signals from the content of the image memory for a standard RGB monitor and a black and white monitor. The color tables in this processor can be loaded from the micro computer. This makes it possible to have full software control over the grayscale and pseudocoloring of the displayed images. Zooming and scrolling of the images is also possible. Finally the display processor has registers for controlling the position of a cursor symbol, allowing convenient operator interaction with the image display.

The microcomputer, a LSI-11 [10], is connected to an interface essentially intended for generating graphics using two of the bitplanes of the image memory. A function box and a digitizer pad are connected to the microcomputer via separate I/O interfaces.

The microcomputer software handles control functions and vector graphics to be overlaid on the gray level images. It accepts two different types of commands. The first type of commands are sent together with their data from the host to the microcomputer, where they are executed by a small interpreter. Some typical ex-

amples of such commands are

1. Draw/erase a sequence of vectors from the data that follows

2. Write/erase the text that follows

3. Set a subrectangle of the image to a given value

4. Enable the cursor and read back the cursor coordinates to the host

5. Enable the graphics tablet and track the stylus position until a termination signal is received

6. Load the color memory with the data that follows

The other type of commands are given via the buttons on the function box. These commands are typically used to control the video interface. Some examples are:

1. Turn the graphical overlay on/off

2. Zoom in/out

3. Move the display window up/down/left/right

4. Turn the cursor symbol on/off

The main communication line between the PDP-11 and the image display hardware is an asynchronous 16 bit parallel I/O interface working at a rate of about 0.5 M byte/sec. This interface can also be multiplexed to communicate with the graphics interface and thus with the microcomputer. Through this interface gray level images can be transferred directly from the host computer into the display memory and instructions can be sent to the microcomputer.

Image scanners

Image input to the system is presently through the link from the old CELLO-8 system. In that system a diode-array scanner [11] is used to provide high quality digitizations of microscopic images. This solution has some advantages in that image aquisition can either be done off-line using only the PDP-8 system or on line under direct control from CELLO-11. It is however a rather slow system and the added complexity of having to run two computers in order to obtain images is a disadvantage. We are therefore presently waiting for a new TV based automated microscope which will be connected to the high speed bus and thus input its images directly into the image memory of the display system. This microscope is being constructed by the group which constructed the display system.

COMMAND LANGUAGE

Syntax

The system is controlled by commands typed on the terminal keyboard. The syntax of the commands is defined by the grammar in figure 2, i.e. a command has the following general format:

```
$LABEL COMMAND QUALIFIERS (PARAMETERS) ->RESULT;
```

The only mandatory part of a command is the actual COMMAND verb. All other parts assume default values when absent. The qualifiers are used to request a special function of the command.

Two kinds of parameters, string parameters and numerical parameters, can be given to the command.

String parameters, which must preceed the numerical parameters, are made up from three different kinds of string data: string constants, string representation of numerical values and file references. Several substrings of different kinds can be concatenated to constitute a string parameter for a command.

Numerical parameters are arbitrary arithmetic expressions. Thus a numerical parameter can be a constant, a variable or a more complex expression. All arithmetic operations are performed in floating point representation.

```
Z ::= [<label>] <command> [<parameter>] [ -> <variable>] ;
<label> ::= $<identifier>
<command> ::=  <identifier>!<identifier><qualifierpart>
<qualifierpart> ::= <identifier> { &<identifier>}
<parameters> ::= ( {<strpar>,} {<numpar>,} )
<strpar> ::= '<char> {<char>}'
<numpar> ::= <expr>
<expr> ::= <term>|<term> + <expr>!<term> - <expr>
<term> ::= <factor>|<term> * <factor>!<term> / <factor>
<factor> ::= <const>|<variable>|(<expr>)
<const> ::= <unop><digit>{<digit>} [.{<digit>}] [E<unop><digit>]
<variable> ::= <identifier>
<identifier> ::= <letter>{<alphanum>}
<alphanum> ::= <digit>|<letter>
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<letter> ::= A | B | ..... | Z
<char> ::= ASCII(40) ..... ASCII(176)
<unop> ::= + | -
```

Figure 2. Pseudo BNF-grammar describing the syntax of the CELLO command language. Entities enclosed within a bracket pair ([]) are optional, entities enclosed within curly brackets ({}) can occur from zero to any number of times in the definition.

Many commands can return a value or a set of values. These values can optionally be stored in a predefined data record by means of the assignment operator '->'. The values are stored in symbolic addresses, which implies that the same command can store its data in several positions of the data record. The names of these positions can be used as variables in arithmetic expressions. Originally this feature was intended for storing extracted numerical parameters from cell images and later for retrieving the information for e.g. statistical analyses , but it can also be used as a means of transferring information from one command to another.

Labels make it possible for one command to reference another one. In combination with the possibility of entering several commands in sequence before the execution starts, the labels can be used to impose a control structure on the language. In the CELLO system this has been achieved by introducing a set of conditional jump commands which transfer control to a referenced command when a certain condition is fulfilled. In this way conditional execution and loops become possible, although perhaps not always in a clean and structured way.

Below some examples of commands are shown:

```
DISPLAY;                      "display the image"
DISPLAY MASK;                 "display the mask (binary image)"
AREA(0,40);                   "count all picture elements with gray
                               values in the range 0 - 40"
AREA(0,40)->NUCLEUSAREA;      "as above except that the computed
                               value is stored in a user defined
                               variable NUCLEUSAREA"
```

Command types

In an attempt to impose some structure on the wide variety of commands we try to classify them as belonging to one of eight different categories. These categories are:

1. System control commands.
   Controls the CELLO system providing the tools for writing and maintaining procedures, libraries, indirect files, data files etc.

   Examples:

   SET - turn various system options on or off.
         More than 25 different qualifiers.
   IF - conditional jump to specified label.
   GOTO - jump to specified label.
   RESET - open indirect file.
   RENEW - move pointer to next record in indirect file.
   DEFINE - insert a new command in the system tables
            making it immediately available for use.
   REQUEST - send instructions to the CELLO-8 system,
             mainly used to request the scanning of new images.
   VARIABLE - define new variables to the system and open
              a new data base where corresponding values
              can be saved

2. User interaction commands
Display images, histograms etc on the terminal. Accept graphical input via joystick or cursor.

Examples:

DISPLAY - show an image on the terminal in one of many possible ways depending on qualifier.
HISTOGRAM - generate and display the gray level frequency histogram
HARDCOPY - make a paper copy of the display file.
CROSS - show the density cross-section of the image along specified line

3. Image processing commands
These are the traditional image to image operators accepting one or more images as input and generating one or more output images, either binary or grayscale.

Examples:

MASK - generate binary image through thresholding or some other procedure depending on qualifier
ADD - add two images together with optional scaling
GRADIENT - generate gradient image, either only magnitude or both direction and magnitude.
SMOOTH - apply one of several selectable smoothing operators to the image
ERODE - run the generalized erosion operator on the (usually binary) image

4. Feature extraction commands
Extract numerical data from objects in the images. Usually the objects are defined through one of the binary masks. About a dozen such commands are available which together with their qualifiers and parameters makes it possible to extract hundreds of features.

Examples:

AREA - compute the object area.
EXTINCTION - compute the integrated optical extinction value.
DENSITY - compute various measures from the density distribution depending on the qualifier.
SHAPE - compute various features based on a fourier analysis of the object contour.

5. Data base manipulation commands.
Handle data that have been extracted from the images. Write out selected parts of the data base in numerical or graphical format. Create new populations by splitting or merging old ones etc.

Examples:

SCATTER - show a scatter plot of two selected variables
MERGE - create a new data set from two old ones
FISHER - run Fisher linear discriminant analysis on selected features

6. Segmentation commands
Process images and other data in order to produce new representations of the information more useful for further processing. Both general purpose image processing algorithms and specialized cell analysis procedures are available.

Examples:

KILL - delete all connected objects from a mask except specified ones e.g. largest, in certain size range, containing point x,y
OVERLAP - analyze contour of object and decide whether it is likely that it consists of more than one cell nucleus.

7. Miscellaneous commands
Various odd commands that do not fit any of the above categories.

Examples:

SIERPINSKI - Draw a Sierpinski curve on display.
TESTSEG - Compare an image segmentation to a reference one (see section 6.2)

8. Temporary commands
New or experimental commands that have not yet been accepted as useful permanent additions to CELLO.

Command procedures

The power of the command language has been amplified through the use of procedures. Procedures can be assembled from sequences of commands and other procedures. When a procedure is called, the call is replaced by its procedure definition. The syntax of the procedure calls is a subset of the command syntax, implying that the user can not see any actual difference between using a command or a procedure. In this way a complex processing algorithm, which can be expressed as a sequence of simple commands, can be defined as a procedure and executed as a single command by the user. The following is an example of a short procedure:

```
NUCLEUSVALUES('$1')

   MASK(0,$1);                "create a mask (binary image) of
                               all picture elements having gray
                               values in the range 0 - $1"
   EXTINCTION MASK->EXT;      "compute the integrated light
```

```
                                extinction and store in the
                                variable EXT"
   DENSITY MEAN->AVLEVEL;       "compute the average gray level within
                                the mask and store in AVLEVEL"
```

This procedure is activated by giving its name and a formal parameter in replacement for its symbolic parameter in the description above, i.e.

```
NUCLEUSVALUES('35');
```

gives as a result integrated light extinction and average gray level calculated from the gray level image, but only for the image points with gray values in the range 0 - 35.

The help facility

Presently we have about 150 different commands available in CELLO. Of these about one third can be said to constitute the basic system. Another third are commands that have been developed for various applications but found to be of general usefulness. The rest are special purpose commands of little general interest. The sheer size of the system makes it difficult for the casual user to remember everything he needs to know. The fact that CELLO is growing and changing with at least a few commands each week makes it even more difficult to keep up with the present status of the system.

An aid in coping with these problems is the HELP command. This is a command which gives various kinds of information about the system. Used without any qualifiers or parameters it explains its own use. With the qualifier SHORT it gives a compact listing of the abbreviated names of all commands in the system. With a command name as string parameter it explains the use of that command. This is done on two levels. Firstly the command syntax as defined by the system tables is shown. Secondly approximately one screen full of text is written about the use of the command. It is the responsibility of the programmers to write such texts in a standardized format as soon as a new command has been written and permanently added to the system.

Other qualifiers and parameters to the HELP command produces other kinds of information about the system. Our general experience is that the help facility has been extremely useful in maintaining an up to date documentation of the system available to all users.

IMPLEMENTATION

The main key to reaching the design goals within the limited resources of the available hardware was modularity. Thus each command corresponds to a single independently loadable program module (task). Adding a new command to the system requires writing a program for it with a standardized beginning and end as well as a standardized communication area. By means of prewritten subroutines in a library this is very simple. The new command has to be identified for the system (a couple of system tables have to be

updated), which is done with a special command. The rest of the system is left completely unaltered. This procedure makes it quite easy to add new commands.

The monitor itself also consists of four tasks; a text editor, a macro processor, a translator and an interpreter. These tasks communicate with each other via a global common data area (one for each active user). The user enters and edits command strings with the text editor via the terminal keyboard. If any procedure calls are included they are replaced by their procedure bodies by the macro processor. Then these strings are checked for appropriate syntax and translated to internal form by the translator. Finally the internal form of the command strings is executed by the interpreter, which means one of two actions: either a command induces execution of a small piece of code in the interpreter itself, or, in most cases, an external task corresponding to the command is activated. Figure 3 shows the logical steps in the processing of a command or a sequence of commands. In the following paragraphs we briefly outline how these tasks operate. A more detailed description can be found in [12].

Text editor

The text editor has two different functions: firstly all input is entered and edited with it. Secondly, it is used to maintain a library of procedures as described above.

It works like any normal text editor: text can be entered and manipulated by means of simple commands. It has some similarities with the TECO editor [13]. Text can be entered either from the keyboard or can be read in from a file.

Macro processor

The procedure facility is implemented by using a macro processor. This processor is enabled when the LIBRARY command is used to specify which procedure library should be used. When the macro processor is activiated it processes the output from the text editor before it is used as input for the translator task. The macro processor replaces all procedure calls with the corresponding procedure definition. The macro processor is fully recursive allowing integer arithmetic, text string operations, conditional expansion, logical tests, etc. [14].

Command language translator

The input text string is checked syntactically and translated to interpretable code by the translator task. This task processes the input text in two passes. In the first pass all labels are entered into a special symbol table and assigned a relative value. The second pass performs the syntax checking, translates the input into an internal form and assigns an absolute value to the labels.

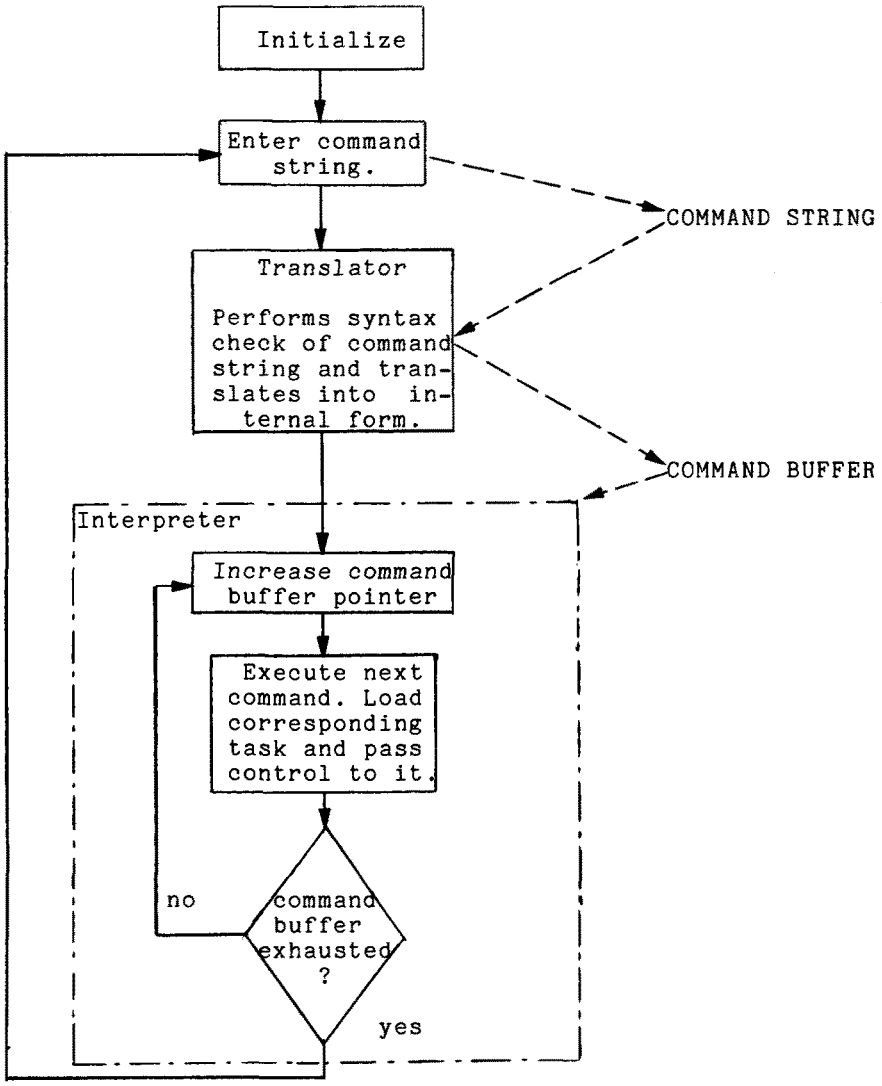

Figure 3. Flow chart showing the logical steps in the processing of a sequence of commands.

Internally the translator consists of a scanner which reduces the input into tokens (identifiers, operators ...) and a parser, which is a collection of procedures, one for each syntactic element. Each of these procedures performs the symbol table handling, the code generation and other tasks associated with its particular syntactic element. The input is checked syntactically and translated one command at a time.

The parser translates the commands into a useful form for the interpretation task, i.e. arithmetic expressions have to be stored in a form that is easy to evaluate at run time etc. If any errors are detected in the input text, an error message is written, the rest of that command is skipped and control is returned to the text editor rather than given to the interpreter when all commands have been processed.

The transition matrix technique [15] is used for syntax checking. This is a simple and fast method for parsing small grammars.

Interpreter and run time system

The input for the interpreter consists of the translated form of the input text together with the present state of the system. The system status is stored in a dedicated data area (the communication area).

Executing a command roughly consists of the following steps in order:

1. Read the internal form of the command with its parameter field and identify the command.

2. Evaluate the string parameters.

3. Evaluate the numerical parameters. The parser evaluates all numerical expressions which do not contain any variables. The remaining expressions are stored in a text pool and are evaluated by the interpreter at run time.

4. The final version of the internal form is stored in the communication area.

5. A check is made in order to see whether the command can be executed immediately, or if an external task execution is required. If the command is internal the code is executed and control returns to step 1. If the execution of an external task is required, that task is given control. The synchronization is accomplished using global event flags.

The external task also has access to the communication area, where it can leave calculated results, a modified image or mask etc. When completed the external task returns the control to the interpreter. Then the next command is executed or control is given to the text editor task if the input string is exhausted.

Graphics software

CELLO can be operated from any user terminal as was described in the hardware section above. The problem of incompatibility between different graphical devices has been solved by letting CELLO maintain a display file. The display file is a data area unique to each user containing primitive graphical operations, such as move to a coordinate, draw a vector from one coordinate to another. Text strings and image file identifiers are also stored in the display file. The display file is built up by different segments, where each segment corresponds to a particular subgraph to be shown on the display. Segments are deleted from and added to the display file by the various commands, each command having its own unique segment.

In order to be shown on a graphical device the display file has to be interpreted by an interpreter specific for the selected graphical device. Thus one interpreter exists for each device. Incorporating a new graphical device into the system requires a display file interpeter to be written for the device, which in general is very easy. This technique also gives a convenient hardcopy facility, as an interpreter has been written for the Versatec printer/plotter.

The graphics software in CELLO is a slightly modified version of a general purpose graphical package developed at our department [16].

Programming languages

The interpreter, translator, text editor and macro processor tasks are all written in the programming language PASCAL. PASCAL was chosen because of its possibilities for data structuring and clean control structure, making it a suitable high level implementation language. External tasks corresponding to commands are programmed in FORTRAN IV and compiled by using an optimizing compiler. FORTRAN is efficient for heavy calculations on relatively simple data structures, making it suitable for the commands, which perform the actual image processing operations.

EXAMPLES OF USE

The system internally handles two gray level images consisting of 128x128x6 bits each, and four masks (binary images) with 128x128x1 bits. The images and masks are referenced by commands by means of numerical parameters. Many commands in the system are designed for analysis of objects (particles) in the gray level images. An object in this context is a part of a gray level image defined by a connected region in a mask. Thus the gray level images and the masks are very closely connected to each other.

In this section two examples are given, illustrating how the system can be used. The first example is a purely interactive application, where only one command at a time is entered to the CELLO monitor. The second is a pure batch example comparing two

methods for automatic segmentation of cytoplasms of cervical cells.

The selected examples represent two extremes - interactivity versus batch - , and it should be pointed out that it is possible to adapt the system to any level of interactivity between these limits. In a particular study procedures often are written for all well defined parts in the processing before the actual work begins, while the remaining parts are written as procedures when a couple of cells have been processed and experience has been gained.

Interactive application

This example discusses the processing of an image of cell nuclei from prostatic cancer. It is assumed that the images have been scanned earlier and reside on disk in special disk files. The command

```
GET ('WEST1.PI',1);
```

loads image register 1 with the image file WEST1.PI,

```
HISTOGRAM;
```

shows a histogram of the gray levels in the image on the display, and the command

```
MASK (3,0,40);
```

generates a binary image of the picture elements with gray level 0-40 in mask register 3 and shows the mask on the display. Generally masks 0 and 1 are superimposed on image 1, mask 2 and 3 on image 2. Mask 3 was selected in this example in order not to overlay image 1 on the display. Figure 4 is a photograph of the TV monitor at this point of the processing. With the command

```
KILL EXCLUSIVE(3);
```

the cursor is enabled, whereupon the operator is supposed to point at the object in the mask that he wants to keep. At this moment mask 3 contains the definition of one of the cell nuclei.

When an object is well defined and isolated, various numerical parameters can be extracted, e.g.

```
AREA MASK(3) -> NUCAREA;
```

calculates the area of the mask and stores the value in the variable NUCAREA.

When an object has been processed and the numerical values have been stored in the data record, the data record must be saved before the next cell is processed. This is done with the command

```
SAVE DATA ('WEST1DATA');
```

which saves the data record under the name WEST1DATA in a dedicat-

ed file that was allocated when the variables were defined. Later on commands are used for calculating statistics from the data file.

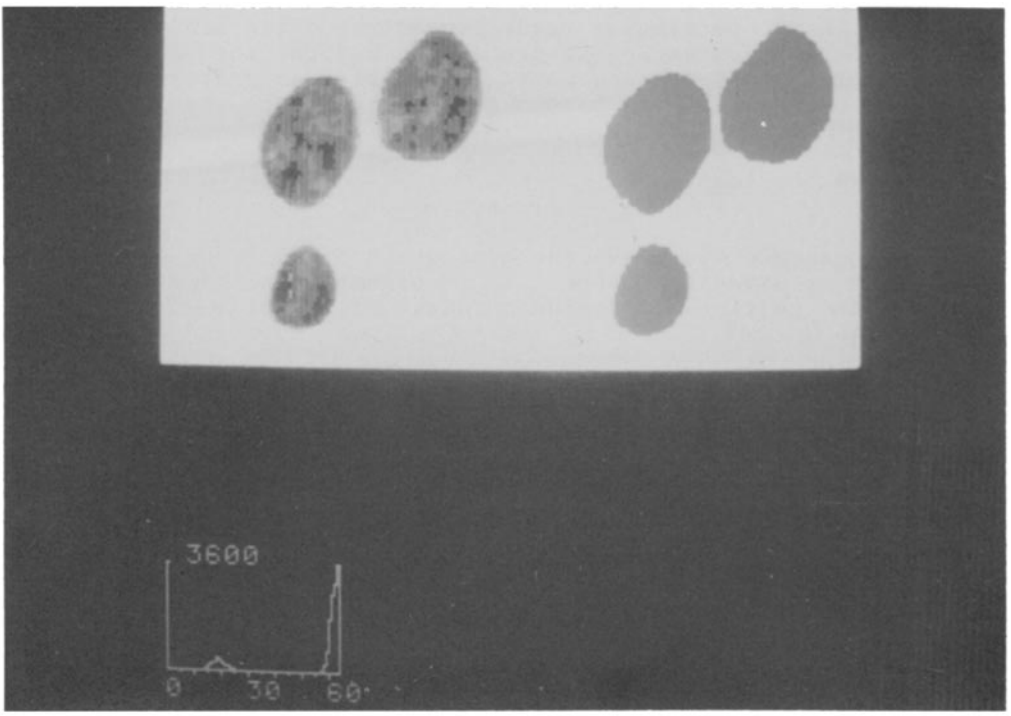

Figure 4. A photograph of the color TV monitor during the interactive application example as described in the text.

Batch application

This example is somewhat more complicated than the former, but it shows how CELLO can be used as a more powerful tool in image analysis. It should be emphasized that we do not wish to discuss the relevance of the selected image processing algorithms or experimental designs, but rather to illustrate the convenience and flexibility of the system.

Two methods for segmentation of cervical cells are compared. Both methods segment the images into background, cytoplasm and nucleus. In this study we are particularly interested in comparing the cytoplasm identifications. It is assumed that two gray level images of each cell are available from disk, one scanned at an illumination wavelength of 530 nm and the other at 570 nm [17]. Furthermore it is also assumed that three masks for each cell are also available from disk. The first one is an automatically produced nuclear mask. The other two are a nucleus definition mask

and a cytoplasm definition mask, both of which have been defined by user interaction in order to obtain a reference segmentation. The accuracy of this reference may be questionable but it is here treated as "the truth". In addition to these data a data file is required containing parameters extracted from the cell, e.g. the center of gravity of the automatically produced nuclear mask.

A fourth mask is produced during the processing in this example, an automatically calculated cytoplasm mask. This is done by means of a dynamic thresholding in the two dimensional gray level histogram obtained from the two original images. Thus an automated segmentation procedure is compared to a manual one.

As a criterion for comparing results from the different segmentations, the method proposed by Yasnoff et al [18] is used. A command, TESTSEG, was written and incorporated into CELLO for this purpose. Programming and testing of this command required only about two hours work which indicates how simple it is to add new facilities to the system. TESTSEG operates on the four mask registers and can return a value to be stored in a variable of the data record.

In addition to the new command a procedure, EVALSEG, was developed for processing a large number of cells without any operator interaction. The procedure is listed in Table 1. It has a file containing identifiers for all cells to be processed as a parameter. This file is called an indirect file. When a command references an indirect file (the file name is preceeded by '@'), the reference is replaced by the current record in the indirect file. Together with a command for moving the current record pointer this gives a convenient facility for obtaining different text parameters in commands each time a command in a loop is executed.

The first two commands in EVALSEG are outside the main loop. RESET sets the current record pointer to the beginning of the indirect file, while FATAL defines a label to which control is transferred when a fatal error occurs in the processing. The latter command together with

```
$NEXT RENEW ('$1');
```

at the bottom of the loop forces the procedure to continue with the next cell instead of stopping the whole run in case of an error.

The five GET commands at the beginning of the loop will load the two images, the two reference masks and the old data record of the cell.

HIST2D computes the two dimensional gray level histogram, stores it in a file (transparent to the user) which in turn is used by CYTMASK which does the two dimensional thresholding for defining the cytoplasm in mask 3. The next three commands will clean the cytoplasm mask by using an ERODE operation and by deleting all objects except the one containing the coordinates CENPOS[1], CENPOS[2], the center of gravity of the automatically produced nuclear mask. These coordinates are available via the data record.

The next three commands in turn load the automatically segmented nuclear mask, calculate the comparison measure between the two segmentations, store that value in the data record and finally save the data record.

The final processing consists in computing two difference masks, i.e. masks showing the difference between the results from the two methods. The first one shows the nuclear differences and the other one the cytoplasmic differences. The cell image in image register 1 is loaded into image register 2 as well and the masks 0 and 1 are cleared. At this point two identical gray level images are shown on the display with the two difference masks superimposed on the right one. Figure 5 shows a photograph of the TV-monitor at this time.

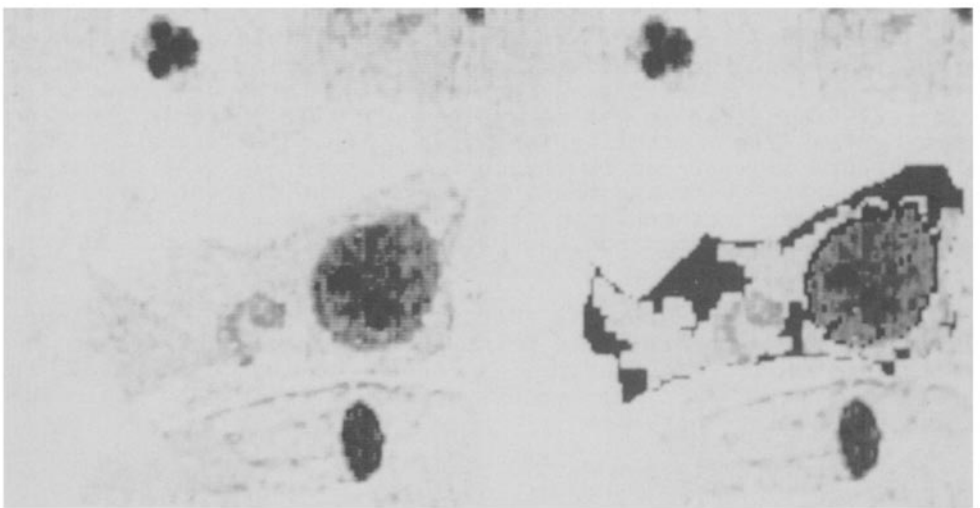

Figure 5. A photograph of the color TV monitor during the batch application example as described in the text. The dark areas along the cytoplasmic- and nuclear borders which indicate the difference between the two segmentation methods are shown in different colors on the TV.

The only thing that remains to do in the loop is to advance the current record pointer in the indirect file and jump to the beginning of the loop.

The next step in an evaluation of data in general is to plot frequency histograms and scatter diagrams over selected variables in a population of cells. As an illustration of this the result of the following command sequence is shown in figure 6:

```
WINDOW VIRTUAL (-1000.,11000.,-1000.,11000.);
SCATTER ('MALIGN','CMSRES','THRRES','X');
```

The first command defines a new coordinate system to be scaled down to the display screen coordinates. The second command draws a scatter diagram on the display of the population MALIGN (a data record file with the name "MALIGN") using the variables CMSRES and THRRES on the horizontal and vertical axes respectively. CMSRES is the variable which was stored in the data record by the procedure in the example above. THRRES is the same segmentation comparison measure but using a slightly modified method for producing the cytoplasm mask. Thus the scatter diagram illustrates the differences in performance of two different segmentation techniques tested on the same material. There are also commands available for more sophisticated statistical analyses such as linear discriminant analysis and cluster analysis.

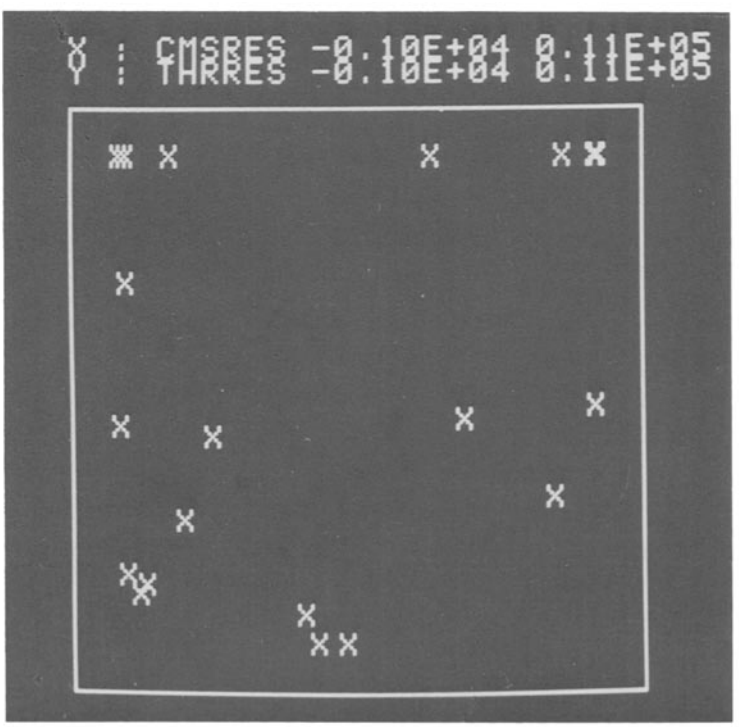

Figure 6. An example of a scatter diagram as produced on the TV-screen when the error measures from the two different segmentation experiments are compared (for details see text).

PERSPECTIVE

CELLO-11 has now been in routine use for about two years. The main application has been the development and testing of methods for automated screening of cervical smears. The general experience is that the system has been very useful. The reason for this is to a large extent the modularity and the fact that the system is programmable with procedures. Since the procedures can call other procedures it is easy to start with one part of a problem and write a procedure for it. When this part is solved, another part is attacked in the same way. Partial solutions can then be put together to new procedures and so on. This is known as the bottom up approach in computer science. In the procedure for segmentation of cervical cells [19] for instance there are several hundred commands. These commands are put together into subprocedures, each one solving a particular subproblem.

There are always possibilities to improve even an already successful system. Despite the fact that it is convenient to add new commands, a highly desirable feature would be the possibility to program all parts of an algorithm directly in the command language without having to write FORTRAN programs as external tasks for each command. The fact that CELLO has been used for programming in the command language to such a large extent further amplifies the need of improving this language. Thus what is needed is the development of a new interactive programming language with powerful general structuring facilities and with special data types for image processing.

Such a language has been designed and named ILIAD (Interactive Language for Image Analysis and Display). The general syntax is similar to that of PASCAL but the system is fully interactive. Thus new procedures and variables can be defined at any time. An image data type is available which makes the buffered access to images of any size and picture element type completely transparent to the user. The procedures can be of two types, internal or external. The internal procedures are procedures previously written in ILIAD and saved in libraries. The external procedures are similar to the CELLO command modules, i e independent programs linked to the system through certain global data areas. The syntax of the calls of both types of procedures is exactly the same. Thus the user will be able to develop his image processing algorithms entirely in the ILIAD language and, when he is satisfied with a certain procedure, reprogram the most time consuming parts in FORTRAN or assembler language to gain speed (if necessary).

The present status of our implementation of ILIAD is that we have the basic system working but that the handling of images, libraries and external tasks is still very primitive. We expect to complete the new system within the next year. Most of the CELLO commands will then be available in ILIAD as external procedures.

The custom-built image display system that is used by CELLO has an architecture that gives it the potential of doing much more than merely displaying images. Different kinds of image processors can be attached to the I/O ports of the high speed bus e g

segmentation processors or linear filter processors. The modular structure of CELLO (and ILIAD) makes it very easy to adapt the system to control such hardware. One only needs to write simple programs that sends the appropriate control information to the hardware when certain commands are given.

The modular structure and device independent graphics also makes it easy to use CELLO with other display systems. All that needs to be modified is the display file interpreter for the image display and some commands that use special hardware features in the display system.

The modularity of the CELLO system has been mentioned several times in this paper as the key to the success of the system in several respects. But this modularity has a price. It does introduce a certain amount of overhead in the system monitor. This is in the order of one second per command. It also makes the total size of the system greater than it otherwise would need to be since significant parts of the code are used in several modules. Considering all the advantages of the modular structure we certainly think it is worth paying this price.

ACKNOWLEDGEMENT

This research was supported by the Swedish Board of Technical Development under grant no 77-3815 and the Bank of Sweden Tercentenary Foundation under grant no 77-112.

REFERENCES

1. Holmquist J.
On analysis methods and software design for computer processing of digitized microscopic cell images.
Ph. D. thesis from Department of Computer Science, Uppsala University, 1977.

2. Bengtsson E.
On the design of systems for computer aided analysis of microscopic images.
Ph. D. thesis from Department of Physics, Uppsala University, 1977.

3. Holmquist J, Bengtsson E, Eriksson O, Nordin B, Stenkvist B.
Computer analysis of cervical cells. Automatic feature extraction and classification.
J. Histochem. Cytochem, Vol 26, No 11, pp 1000-1017, 1978.

4. Stenkvist B, Westman-Naeser S, Holmquist J, Nordin B, Bengtsson E, Vegelius J, Eriksson O, Fox C.H.
Computerized Nuclear Morphometry as an Objective Method for Characterizing Human Cancer Cell Populations.
Cancer Research 38, 4688-4697, Dec 1978.

5. Bengtsson E, Holmquist J, Olsen B, Stenkvist B.
SCANCANS - An interactive scanning cell analysis system.
Computer Programs in Biomedicine 6, pp 39-49, 1976.

6. Holmquist J, Bengtsson E, Eriksson O, Stenkvist B.
A program system for interactive measurements on digitized cell images.
J. Histochem. Cytochem, Vol 25, No 7, pp 641-654, 1977.

7. Knowlton K, Harmon L.
Computer-Produced Gray Scales.
Computer Graphics and Image Processing, Vol 1, No 1, pp 1-20, 1972.

8. Holmquist J, Antonsson D, Bengtsson E, Danielsson P-E, Eriksson O, Hedblom T, Martensson A, Nordin B, Olsson T, Stenkvist B.
TULIPS, The Uppsala-Linkoping Image Processing System
Analytical and Quantitative Cytology. In press.

9. Antonsson D, Danielsson P-E, Malmberg B, Martensson A, Olsson T.
A two Mbit random access memory with 512 Mbit/sec data rate.
LiTH-ISY-I-0127. Dept of Electrical Eng., Linkoping University 1977.

10. Microcomputer handbook.
Digital Equipment Corporation, Maynard, Massachusetts 1978.

11. Bengtsson E, Eriksson O, Holmquist J, Stenkvist B.
Implementation and evaluation of a diode array scanner for digitizing microscopic images.
In the Automation of Cancer Cytology and Cell Image Analysis.
Tutorials of Cytology, 1979.

12. Eriksson O, Holmquist J, Bengtsson E, Nordin B.
CELLO - An interactive image analysis system.
Proceedings of Digital Equipment Computer Users Society, Copenhagen, Denmark Sep 1978.

13. OS/8 Handbook.
Digital Equipment Corporation, Maynard, Massachusetts 1974.

14. Holmquist J.
Mll - A general purpose macro processor.
Report 77:7, Dept. of Clin. Cytology, Uppsala University 1977.

15. Day A.C.
The use of symbol state tables.
Computer Journal 13, pp 4- , 1970.

16. Holmquist J, Johansson J, Bengtsson E, Eriksson O, Nordin B.
MTGP - A device independent graphical package for RSX-11/M.
Proceedings of Digital Equipment Computer Users Society, Copenhagen, Denmark - Sep 1978.

17. Holmquist J, Imasato Y, Bengtsson E, Stenkvist B.
A microspectrophotometric study of Papanicolaou-stained cervical cells as an aid om computerized image processing.
J. Histochem. Cytochem, Vol 24, No 12, pp 1218-1224, 1976.

18. Yasnoff W.A, Galbraith W, Bacus J.W.
Error measures for objective assessment of scene segmentation algorithms.
Analytical and Quantitative Cytology, Vol 1, No 2, pp 107-121, 1979.

19. Bengtsson E, Eriksson O, Holmquist J, Nordin B, Stenkvist B.
High resolution segmentation of cervical cells.
J. Histochem. Cytochem, Vol 27, No 1, pp 621-628, 1979.