MED. INFORM. (1984), VOL. 9, NO. 2, 93–102

The interpreter PIC: A tool in the field of image processing

H. P. MEINZER and U. ENGELMANN
Institute of Documentation, Information and Statistics,
German Cancer Research Centre,
Im Neuenheimer Feld 280, D-6900 Heidelberg, FR Germany

(Received September 1983)

Keywords: Command language; Image processing; Interactive computing; Interpreter; Pattern recognition; Picture processing.

1. Introduction

Many languages used in image processing have been coded, around 50 in the USA alone. Whilst some are merely collections of subroutines [1], others are compilable languages with the features of classical programming languages [2, 3, 4]. There are also command- and menu-driven languages which can be used interactively. The syntax, readability and power vary significantly, and are often quite poor [5]. PIC, the language described here, is an attempt to integrate an easy dialogue and flexibility not commonly found in other languages [6].

In image processing today a system designer often retains his own set of subroutines for both basic and sophisticated operations. Thus, with the help of these, a specific solution for a specific problem is tailored. This approach is very costly and is manpower-dependent. A better solution is the use of one common subroutine package by more than one programmer. Thus, programs for one problem are not developed at several different times by several different programmers. The disadvantage of this method is that every time when you want to use one of the features of the subroutine package, it is necessary to write a new main program. Thus it is a method which lacks flexibility.

Experience shows that it is more productive to develop an integrated easy-to-use package which can be applied by not only the programmer or system designer, but also by the user. Of course, such a package must be interactive to support the user who is new to the field. On the other hand, it should be very flexible and new approaches should easily be integrated.

2. Why a command language?

There are three basic methods of controlling a dialogue with a computer. One is the sequential dialogue, a sequence of questions and answers. Such a dialogue is popular with the inexperienced user but proves to be inflexible both for the system designer and the more adept user. Control of a sophisticated dialogue is not feasible with this method.

The second approach to a man–machine dialogue is the menu technique where a user picks an item from a set of predefined possibilities. Initially this method also looks promising. It can be quite attractive for the control of simple systems. The moment a system has more than a few variable parameters, however, this method is no longer suitable because you need more and more menus each of which has to be put through a process of selection.

Another method of controlling a dialogue is a command language. For a given field of application all necessary functions are made available through commands. The set of all necessary commands forms an application-oriented language. A good example for such a language are the text-editing systems (editors) which are available on virtually any computer. The disadvantage of this approach is the greater amount of training required to make use of such a language. Once a user has learned to use a few basic functions, he will appreciate the efficacy of this method. Both the user and the system designer profit from the enormous flexibility of a command language which can also be called a problem-oriented interpreter.

Thus, we have in our institution a generator for building our own interpreters [7] and good experience with interpreters for graphic problems [8, 9]. We decided to develop another interpreter for image-processing problems [10].

3. The language PIC

The language PIC consists of about 110 commands in nine groups:

— dataset management functions;
— display functions;
— simple image manipulations;
— histogram modification;
— gradient operators;
— smoothing algorithms;
— Fourier manipulations;
— arithmetic and Boolean operations;
— auxiliary functions.

Selected commands from each group will be described in more detail. When reading a command it should be explained that all expressions in parenthesis may be dropped. The sign '|' is the logical OR-function. Some dataset management functions are for example

```
LOAD Name
STORE Name
ERASE Name
```

All images are stored online on disc. The size of any image can be chosen arbitrarily with a maximum of 900 × 900 pixels. When an image is loaded into the PIC workspace, name and size are also transferred. Thus, the user addresses an image only via its name and need not deal with the actual size. All following operations are performed on the image stored in the workspace, except in certain cases when an operation is performed on two images.

```
DISPLAY
PRINT (REV) (ON WHITE PAPER)
PLOT HISTO
PLOT TOPO
PLOT 3D (PHI, THETA)
PLOT ISO IVALUE
PLOT CONTOUR
PLOT GREY
```

DISPLAY produces output on a Tektronix 4112A terminal.
PRINT permits a low quality print with no more than 14 grey-levels on a standard alphanumeric printer. The five PLOT functions all need a graphic terminal Tektronix 4014 for the display of the output. HISTO plots the histogram of the grey-values. The most useful commands here are TOPO and 3D which plot a pseudo three-dimensional topogram of an image (figure 1 (*a*)). They have a hidden line algorithm integrated and produce reasonable output. ISO and CONTOUR plot isophotes and contours (figure 1 (*b*)) and PLOT GREY produces a pseudo grey-value image (see figure 3).

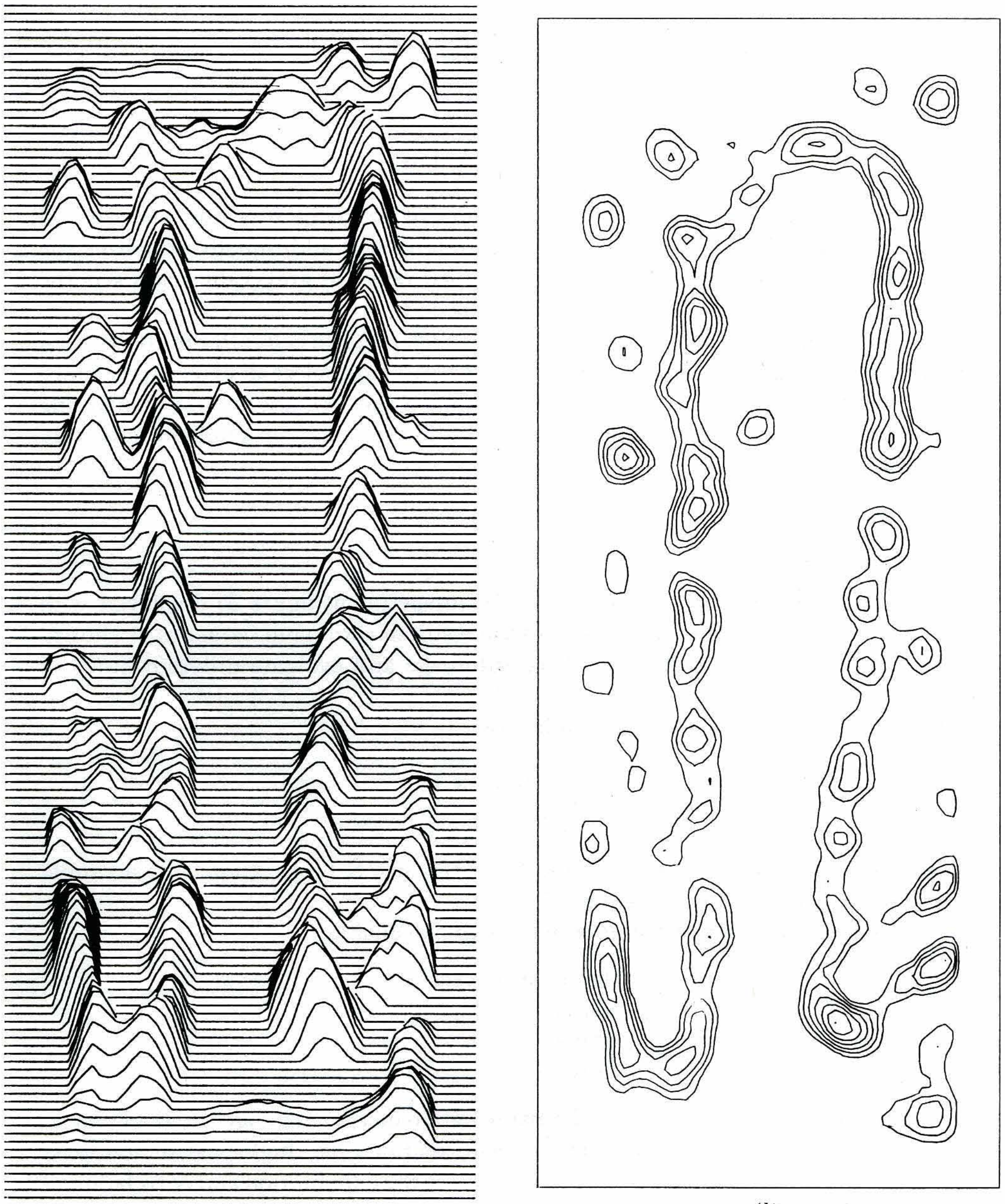

(*a*) (*b*)

Figure 1. (*a*) The topogram of a rat's colon crypt. (*b*) The contour plot of the same image.

Four simple image manipulations are

```
WINDOW IX, IY, IDELTAX, IDELTAY
SCALE N(/M)
ROTATE (90|180|270)
REFLECT (HOR|VER|DIAG1|DIAG2)
```

With these commands both the size and the orientation of an image can be influenced.

Histogram modifications are:

```
NORM GREYSCALE
FLAT
SELECT N-M (SET IVALUE)
```

The first command transforms the grey-values of an image in a way that the full range from 0 to 255 is used. FLAT works similarly and additionally equalizes the histogram. The last command selects a specified range of grey-values and sets the remaining values to zero or to the value indicated.

As an example, we load an image with 'LOAD CELLS' (two cells after mitosis); then we display a topogram of it with 'PLOT TOPO' (figure 2 (*a*)), select all pixels in the range of 50 to 150 by 'SELECT 50-150' (the remaining pixels become 0) and display the topogram of the resulting image (figure 2 (*b*)). There are seven gradient operators available in PIC, e.g.,

```
SOBEL
ROBERTS
```

For example, figure 3 (*a*) shows an original image (as in figure 2) displayed with command 'PLOT GREY' and in figure 3 (*b*) the result of command 'ROBERTS' is shown. All functions work on a three by three subimage and evaluate different kinds of gradients.

For smoothing we have commands like

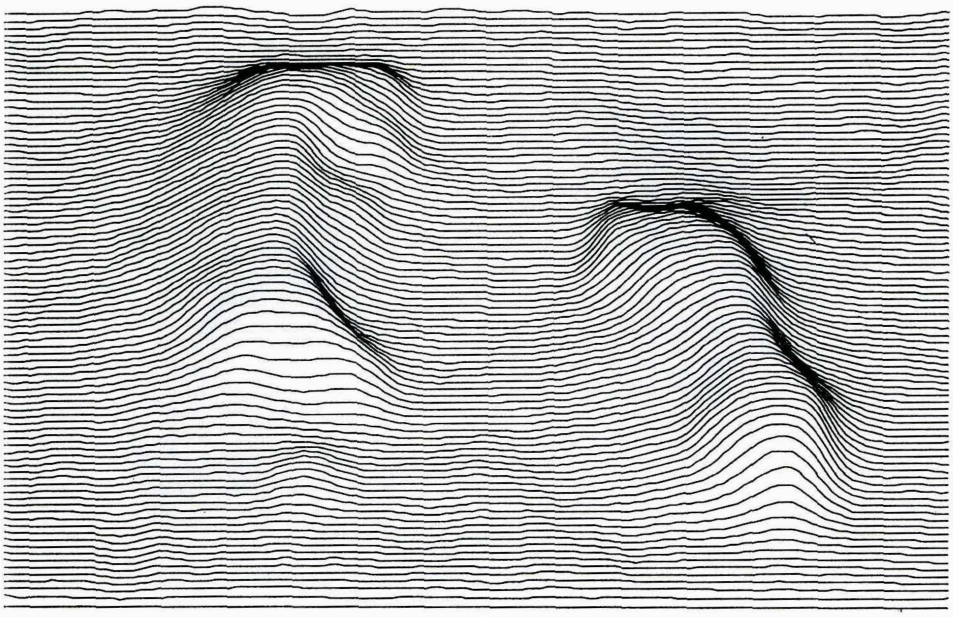
```
MEDIAN N
SMOOTH
```

MEDIAN is a median filter with a window of the size N by N. SMOOTH is a simple operator on a three by three window.

There are a number of FOURIER OPERATIONS available:

```
FOURIER (INVERSE)
CONV
CORR
POWER
```

FOURIER evaluates the Fast Fourier Transformation of a given image and the inverse. Operations on the resulting Fourier Transformations like high-pass or low-pass filters are also available. CONV and CORR determine the convolution and correlation of two images in the frequency domain. POWER evaluates the power spectrum of an image (figure 4).


(*a*)

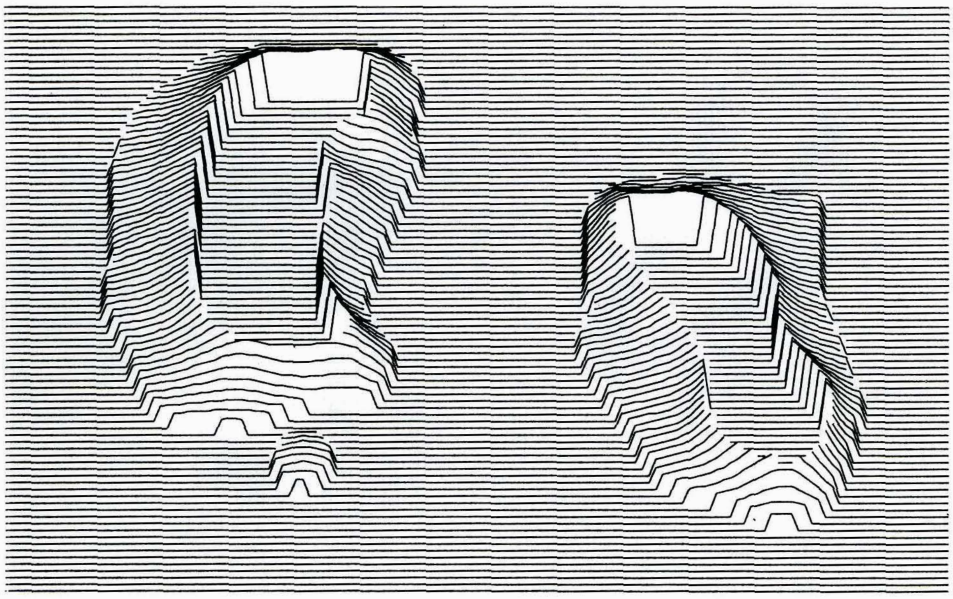
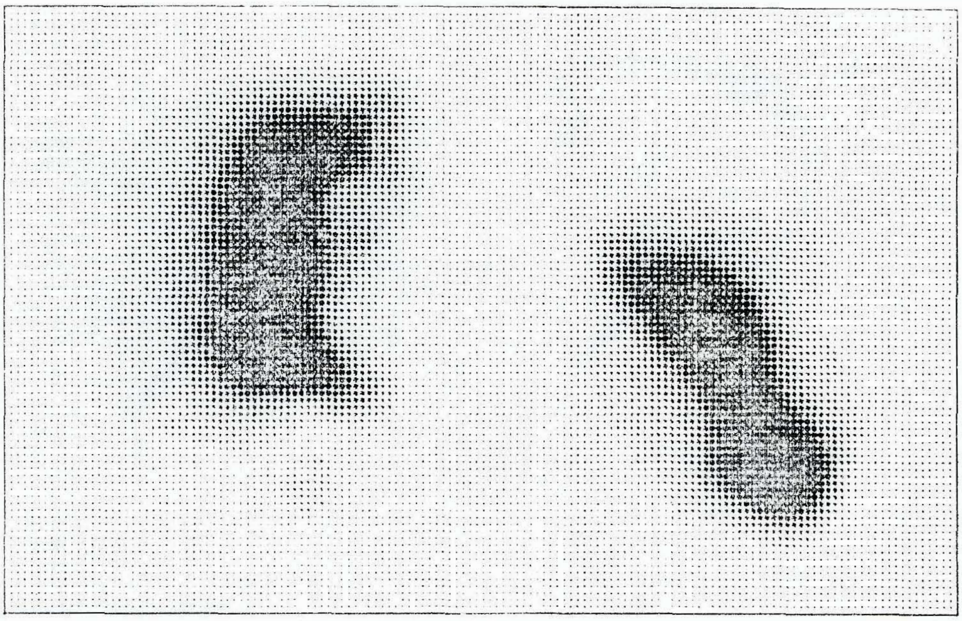

(*b*)

Figure 2. (*a*) Topogram of two cells after mitosis. (*b*) All pixels out of range 50–150 set to 0.


(*a*)

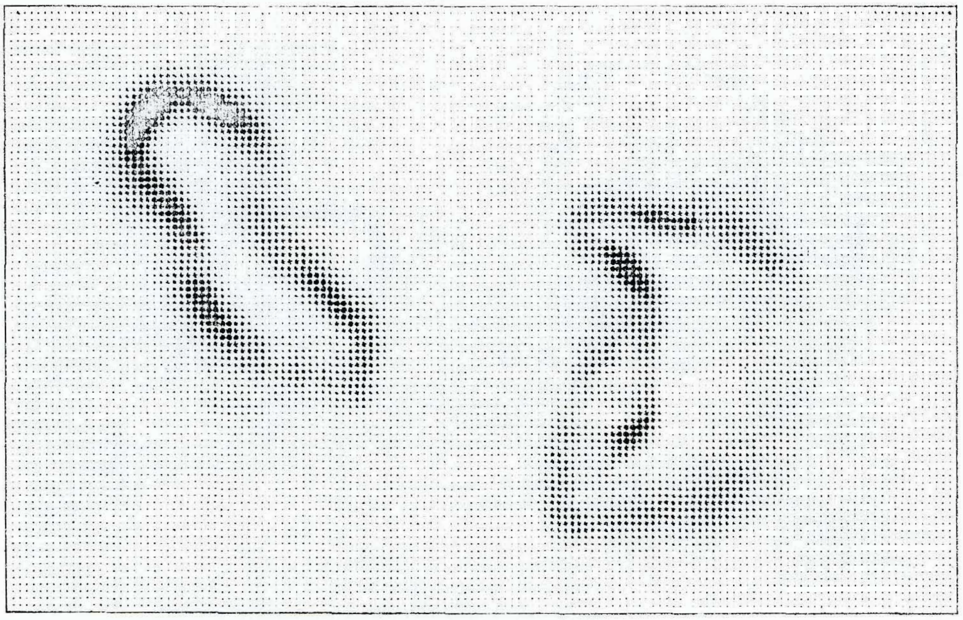

(*b*)

Figure 3. (*a*) A display of two cells with 'PLOT GREY'. (*b*) The plot of the same image after applying the 'ROBERTS'-operator.

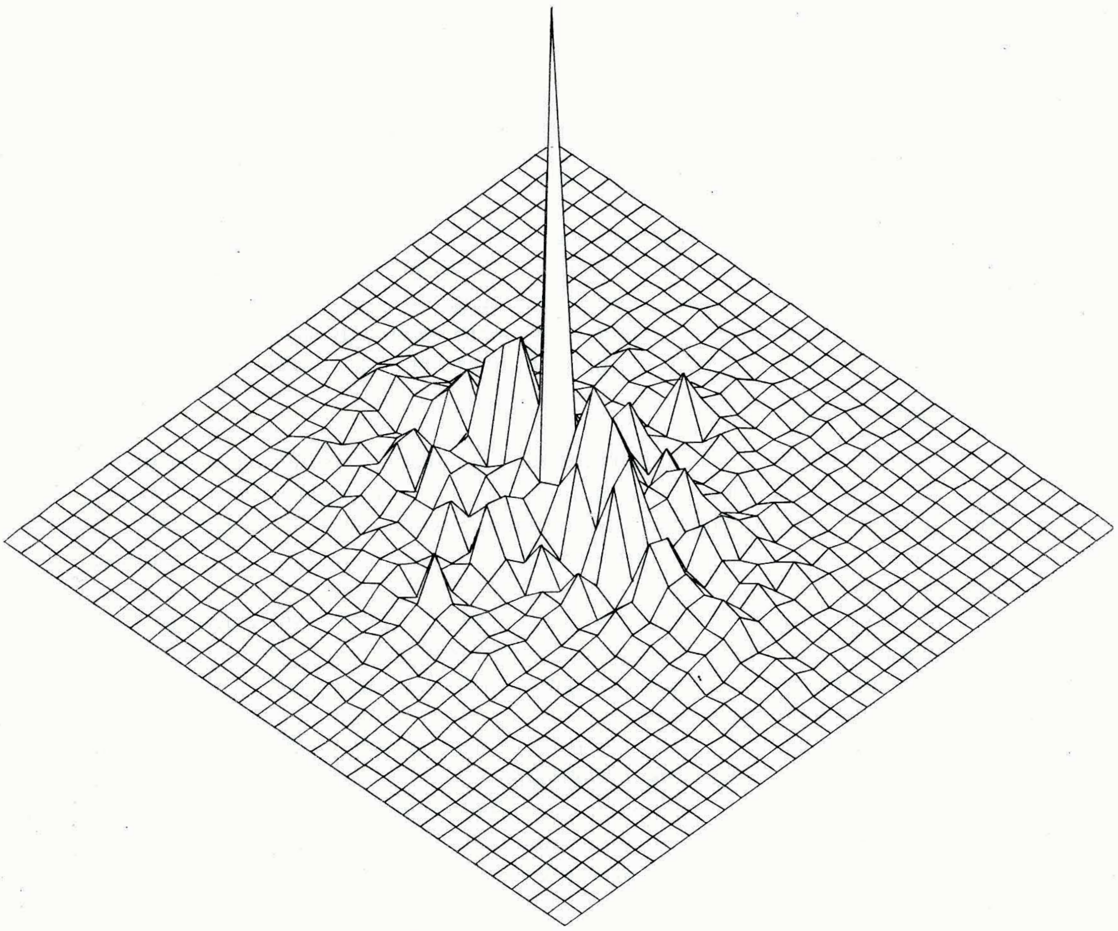


Figure 4. 3D plot of a POWER SPECTRUM.

Some basic arithmetic functions are:

ADD (CONST)
SUB (CONST)
MULT (CONST)
DIV (CONST)

If no constant is specified, the functions work on two images; otherwise, a constant is added, subtracted etc. The Boolean functions AND and OR are working on two images pixel by pixel.

Finally there are a few auxiliary functions:

STATUS
HELP
END

STATUS displays any available information on the image actually in the workspace and the values of a large number of variables. HELP and END are self-explanatory.

Usually the commands are entered and executed line by line. That is not very flexible. Therefore we allow the concatenation of several commands by a semicolon or the word 'AND', e.g.,

'LOAD IMAGE; SMOOTH AND DISPLAY'

These minor extensions make the language look less formal.

4. Special features of the interpreter

Though PIC already includes a number of generally useful commands and though we continue to expand the language whenever we find a feature which can be of use for more than one specialized application, something will always be missing. Specialized individual programs can be included in PIC via a command

CALL subroutine

The subroutines have a predefined parameter list and are loaded dynamically at runtime only if necessary. This command exploits a special feature of our operating system. The use of this option will be restricted to system designers and programmers with a solid knowledge of the internal structure of PIC, our operating system and FORTRAN.

PIC can be used in several ways. Most frequently it is used in the conversational mode at a graphic terminal, where the user can verify the result immediately. For extremely time-consuming operations it is better to collect the command sequence in a file and execute it in batch mode.

The system permits abbreviation of most of the commands. This is very useful for a user who is familiar with the commands. Sometimes a user wants to alter a command word or perhaps to abbreviate a command in a different way. He can then define a synonym for a command or command sequence. Naturally, the building of procedures consisting of PIC commands is also supported. An example for defining synonyms is:

+SHOW:='PLOT TOPO 10, 10, 150, 200'

Abbreviations, synonyms and procedures can be defined locally only for one user or globally for all users. These features were made available through the use of a special input processor developed in our institution [11].

5. Implementation

PIC runs on an IBM 3032 with the operating.system TSS (time-sharing system). It needs at least one graphic output device like a Tektronix 4014 storage tube. The software is written in FORTRAN IV.

The space required for running PIC is less a result of the size of the programs (as we can load subroutines dynamically at runtime only when they are really needed) but more a function of the size of the workspace reserved for the storage of pictures in the program. Today we allow up to three integer pictures (max. 900×900 pixels) and four real pictures (max. 256×256 pixels) which add to a little more than 6 Mbytes.

A simple smoothing algorithm on a 256×256 picture takes 1·3 seconds. A median filter on a 64×64 picture takes two to three seconds, depending on how clever the sort algorithm is. The times quoted are also depending on the workload on our time-sharing machine.

The analysis of the input strings (commands) is done by a parser which was generated by the parser and lexical analyser generating system developed in our institution [7, 12]. For the graphic output we used the Tektronix packages TCS (terminal control system) and AG-II (advanced graphics two) [13, 14]. Some of the image processing software was copied from the literature or contributed by other individuals or institutions [15, 16, 17, 18, 19].

6. Application

As the majority of the commands have a very general character, PIC can be applied to all kinds of digitized images. Usually it is sufficient to add very few specialized routines to solve a special pattern recognition problem. As PIC was developed at the German Cancer Research Centre (DKFZ), Heidelberg, all our applications are in the medical-biological field.

We first applied PIC to one-dimensional electrophoresis gels [20]. These gels are produced by the thousand in all groups working on DNA and RNA mapping and sequencing. The sequencing of viruses of a length of more than ten thousand nucleotids involves production and analysis of hundreds of images. At the DKFZ a very sophisticated software package is also available for the reconstruction of DNA or RNA sequences. The input for the reconstruction is a lot of randomly-cut subsequences. These subsequences have a variable length and can be defined by analysing the pattern of a gel image.

Another field of application is the evaluation and comparison of two-dimensional gels. These images stem from various fields like immunology or cell and tumour biology. The method of pattern recognition allows a comparison of two or more images. In contrast to the purely visual evaluation, the image processing methods permit a quantitative description of images.

In another project we tried to analyse the different stages of tumour growth in the rat's colon. Pattern recognition methods can probably analyse an image more precisely than a pathologist. The results of these evaluations will be used to build a computer simulation model of cell kinetic processes.

Some other projects involving image processing in the biological-medical field will be worked on in the near future, e.g., automatic measuring of DNA length in electron microscope images.

7. Conclusion

Experience with the language PIC shows that the user can understand it and work with it very easily. A beginner in the field of image processing can learn and use (!) standard methods in this field rather quickly.

The didactic value is matched by the simplicity of building problem-oriented programs. Usually only a few specialized routines have to be added to solve a special problem. A researcher can concentrate on the features specific to his application and need not deal with all the standard problems that usually consume so much of his manpower.

Finally, PIC may be expanded whenever a useful algorithm comes up. This can be done because PIC itself is generated by other programs which allow an easy addition of further commands and the appropriate FORTRAN code.

Acknowledgements

Many people contributed ideas and programs to PIC. We wish to thank J. Dengler, M. Jaksch and G. Zinser for the software they made available and Dr Komitowski who provided images. Special thanks go to Professor G. Zajicek, who helped in many ways, and the colleagues in our institution who advised us in using the parser and lexical analyser generators.

References

1. TAMURA, H., TOMITA, F., SAKANE, S., YOKAYA, N., SAKAUE, K. and KANEKO, M. (1982). A transportable image processing software package: SPIDER. In M. Lang (ed.), *Proceedings of the 6th International Conference on Pattern Recognition, Munich, 1982* (IEEE Computer Society Press, Silver Spring, MD), pp. 75–78.
2. AHO, A. V. and ULLMANN, J. D. (1972) *The Theory of Parsing, Translation and Compiling*. Volumes 1 and 2 (Prentice Hall, Englewood Cliffs, NJ).
3. NEWMAN, W. M. and SPROULL, R. F. (1973) *Principles of Interactive Computer Graphics* (McGraw Hill, New York).
4. WIRTH, N. (1977) *Compilerbau* (Teubner, Stuttgart).
5. PRESTON, K. (1980) Image manipulative languages: A preliminary survey. In E. S. Gelsema and L. N. Kanal (eds.), *Pattern Recognition in Practise* (North-Holland Publ. Co., Amsterdam), pp. 5–20.
6. MEINZER, H. P. (1980) Command languages in application programming. In D. A. B. Lindberg and S. Kaihara (eds.): *MEDINFO 80* (North-Holland Publ. Co., Amsterdam), pp. 719–722.
7. BECKER, N., OSTERBURG, G. and SCHADEWALDT, K. (1977) PAULA: Generator für LL (1)-Parser und lexikalische Analyseprogramme, *Technical Report No. 10* (DKFZ, Heidelberg).
8. ALBERTS, S. and MEINZER, H. P. (1981) INSTANT2 – Ein Dialogprogramm zur Erstellung von Graphiken, *Technical Report No. 21* (DKFZ, Heidelberg).
9. MEINZER, H. P. (1979) INSTANT1 – Ein Programm zur Herstellung von Erhebungsbogen und Postern, *Technical Report No. 16* (DKFZ, Heidelberg).
10. ENGELMANN, U. and MEINZER, H. P. (1982) PIC – Ein Interpreter zur Bildbearbeitung, *Technical Report No. 25* (DKFZ, Heidelberg).
11. SCHADEWALD, K., MERX, R. and KYNAST, W. (1980) Der Input-Prozessor, *Technical Report No. 18* (DKFZ, Heidelberg).
12. SCHADEWALDT, K., MERX, R. and LICHT, G. (1983) PES – Programentwicklungssystem. *Technical Report No. 29* (DFKZ, Heidelberg).
13. HAHNE, H. (1974) DKFGRAF – Graphisches Programmpaket. *Technical Report No. 1* (DKFZ, Heidelberg).
14. HAHNE, H. (1976) TEKTRONIX: TCS und AG-II. *Technical Report No. 8* (DFKZ, Heidelberg).
15. CASTLEMAN, K. R. (1979) *Digital Image Processing* (Prentice Hall, Englewood Cliffs, NJ).
16. GONZALEZ, R. C. and WINTZ, P. (1977) *Digital Image Processing* (Addison-Wesley Publ. Co., Reading, MA).
17. PRATT, W. K. (1978) *Digital Image Processing* (John Wiley & Sons, New York).
18. ROSENFELD, A. and KAK, A. C. (1976) *Digital Picture Processing* (Academic Press, New York).
19. ROSENFELD, A. (1979) *Picture Languages* (Academic Press, New York).
20. ENGELMANN, U. and MEINZER, H. P. (1983) Analysis of electrophoresis gels by an image processing system. In J. A. van Bemmel, M. J. Ball and O. Wigertz (eds.), *MEDINFO '83 Proceedings* (North-Holland, Amsterdam), pp. 410–413.