A FLEXIBLE APPROACH TO THE CONSTRUCTION OF EXPERT SYSTEMS FOR THE APPLICATION OF IMAGE PROCESSING SOFTWARE

Horst Bunke, Felix Grimm
Institut für Informatik, Universität Bern,
Länggassstrasse 51, CH-3012 Bern, Switzerland

Song Maoqiang
Computer Engineering Dept., Beijing University of Posts and Telecommunications,
100088 Beijing, P.R. China

ABSTRACT

Expert systems are useful tools for the efficient application of image processing software packages. This paper describes an image processing expert system based on the SPIDER package [1]. The system supports unexperienced users in two different ways: an **interactive mode** in which the selected methods are applied immediately to an actual picture, and a **program generating mode** in which high level program code is generated for later compilation and execution. The modular structure of the knowledge base can be considered as a generalized approach to software configuration expert systems which can be adapted easily to other image processing software packages and other problem domains, too.

INTRODUCTION

A large number of versatile image processing algorithms have been developed in recent years. So the application of standard software packages is becoming more and more important in this field. Such packages are usually powerful tools and are described in voluminous documentations. It often requires much time and effort to become familiar with a specific software package. Especially for a novice user, it may be very hard to find out appropriate programs and correct parameter values to process a given image successfully.

The use of expert systems has been proposed to overcome these problems. Such expert systems support the user in the selection and application of appropriate programs. In order to do this task, these systems contain knowledge about the software package under consideration, the specific problem domain, and software configuration in general which includes knowledge about the selection and combination of program modules and the adaption of parameters. Expert systems for software configuration in various domains are described in [2] - [5].

In this paper, we describe an expert system that supports unexperienced users in the application of the SPIDER image processing software package. Our system provides two different modes of operation: In the first mode, a problem is interactively solved in a stepwise manner. In each step an image processing subroutine is automatically selected and executed by the system, and its results are displayed to the user who can either accept or reject them. In the case of rejection the system explores alternative subroutines. In the second mode of operation no interactive display of the intermediate results is provided. Instead, a program consisting of a number of subroutine calls (including parameter values) embedded in a proper control structure and all necessary declarations is generated. This program can subsequently be compiled and used for the same kind of problems on different input pictures.

Some systems similar to ours have been proposed in [6] and [7]. In contrast with other approaches, our system is based on a more generalized structure which allows an easy adaption to other software packages, and even other problem domains.

SYSTEM OVERVIEW

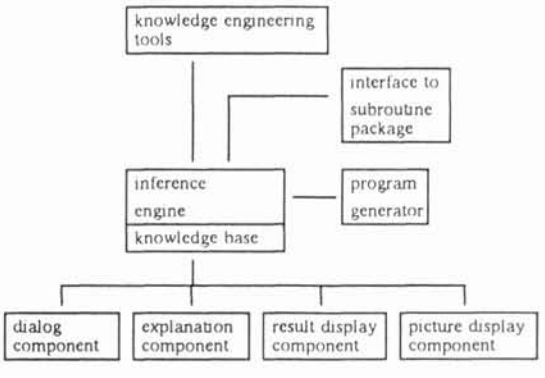

Fig. 1. System overview

The basic structure of our expert system is shown in Fig. 1. Most of the components of our system like the inference engine, the dialog or the explanation component, are similar to equivalent components of other expert systems. Therefore they are not further discussed here. The knowledge base of our system will be described in the following section.

The **program generator** is used in the *program generating mode*. Based on the results of a consultation, it produces a program written in Fortran-77 for a given image processing problem. The program generator stores the produced program into a file which can be compiled and executed subsequently.

The **interface to the subroutine package** and the **picture display component** are used in the *interactive mode*. The interface to the subroutine package allows external subroutines written in any conventional procedural programming language to be executed under the control of the expert system, and parameters to be passed from the expert system to the subroutines and vice versa. The picture display component handles the display of all pictorial information including the original image and all images generated during a consultation by the selected SPIDER routines.

KNOWLEDGE BASE STRUCTURE

To solve image processing problems by means of SPIDER subroutines, three kinds of knowledge are required:

1. **General knowledge** about the application of software packages. This knowledge is independent of the underlying problem domain and includes general strategies of problem solving.
2. Knowledge about **image processing**. This knowledge consists of the standard methods and the kinds of problems they can solve, the possible versions of standard methods due to parameter variation, and the possible combination of standard methods.
3. Knowledge about the **SPIDER software package**. This knowledge relates the methods of image processing with particular names of programs of the software package. Also, it has to provide all necessary details about number, type and meaning of the required parameters thus ensuring proper use and execution of the programs.

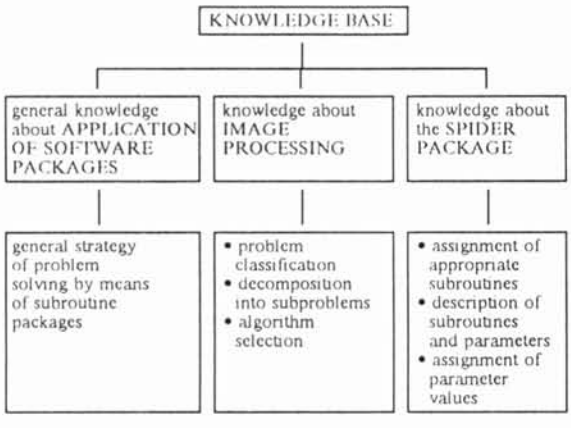

Fig. 2. Knowledge base structure

According to the above categorization, we have logically divided our knowledge base into three parts (see Fig. 2). In this way a great degree of flexibility and versatility is achieved. For example, if the system is to be adapted to another image processing subroutine package, only part 3 of the knowledge base has to be changed. Similarly, if another application domain is to be considered, parts 2 and 3 have to be replaced but part 1 - or at least a major portion of it - can still be used. A similar knowledge base division in the context of an expert system for selecting operation system commands has been described in [8].

1. General knowledge

A strategy to solve problems by means of subroutine packages includes the following steps:

A) **Problem classification.**

B) **Decomposition** of the problem into less complicated subproblems which can be solved separately, including the determination of the order in which they have to be solved.

C) **Selection** of appropriate **methods** and **algorithms** for each subproblem.

D) Fine tuning of the selected methods by assigning various **parameter** values.

This strategy is of very general nature and can be used for problem solving by means of subroutine packages in many other problem domains, too. The only restriction is that the subroutine package under consideration is written in a conventional procedural programming language (like Fortran for the case of SPIDER).

The problem classification is done by knowledge depending on the specific problem domain. After the problem is classified and the goal of the consultation is specified, the whole problem represented by this goal is decomposed into subproblems (subgoals). Each subgoal represents a logical unit of the entire problem solution. The subgoals are further subdivided into steps which correspond to the basic methods and algorithms of the problem domain (i.e. subroutines of the package). This hierarchy of decomposition into three levels (goal, subgoal, step) is arbitrary and could also include more levels. The proposed three levels are considered as a minimal subdivision, which allows a semantic level of subproblems between the level of the entire problem and the one of the actually applied subroutines.

In each step, the system selects an appropriate algorithm as well as a corresponding subroutine. It also handles the parameters of that subroutine. In the case of the *interactive mode*, the values of all parameters are computed and the subroutine is executed immediately. After subroutine execution, the generated results are returned to the expert system and may be displayed to the user or be stored for later use. In the *program generating mode*, only those parameters which will have the same values in all runs of the generated program are computed immediately. A constant definition statement is generated for such parameters. For parameters

depending on specific input data, a calling statement for a subroutine which is able to compute or read in the required values is generated if the corresponding values are not already to be computed by a previous subroutine call. The strategies for the transfer of parameter values computed by one subroutine to another one are described by domain or subroutine package specific knowledge.

Our system also supports other than static flow control structures. In the *interactive mode*, alternative ways of problem solving due to computed subroutine results or error codes can be described easily by domain or package specific knowledge. The interactive mode also supports the repeated execution of one or more subroutines. In the present version of our system, the only really dynamic flow control structure provided by the *program generating mode* is an error code check routine which in the case of a non-zero return code cancels the execution of the generated program. More work would be necessary to introduce an *if-then-else* or a *loop* structure based on values computed during program execution.

2. Knowledge about image processing

The current version of our system includes knowledge about image analysis in about the same extent as version 1 of the SPIDER package except the fields of orthogonal transforms and texture analysis. This knowledge describes problem solving strategies of this domain as already discussed from a general point of view above.

An example of a rule used for problem classification is

IF the goal of processing is the extraction of numerical features
AND the kind of numerical features is location of lines
THEN the goal of the consultation is location of lines.

The problem solution is subdivided into subgoals by means of rules like

IF the goal of the consultation is location of lines
AND the user is interested in boudary lines based on region segmentation
AND region segmentation has not yet been performed
THEN the actual subgoal is region segmentation.

Similar rules deal with the subdivision of subgoals into steps. Given the purpose of the actual step, an algorithm has to be selected next. This is done by means of rules like

IF the purpose of the actual step is region segmentation in a binary picture
THEN the actual algorithm is extraction of connected components.

3. Knowledge about the SPIDER package

The knowledge about SPIDER relates algorithms of image processing with corresponding names of SPIDER subroutines. For those cases where there is only one subroutine for an algorithm, a table is used. Otherwise further constraints are taken into account, using rules like the following one:

IF the actual algorithm is affine transform
AND linear interpolation is to be used
THEN the actual subroutine is AFIN1.

Furthermore the knowledge about SPIDER contains a detailed description of all supported subroutines including information about the number and types of parameters, and instructions for finding their values. In this part of the knowledge, we can find the most important differences between the *interactive* and *program generating mode* because the values of several parameters are to be derived differently under the two modes of operation. An example is the SPIDER routine ACOE which computes the transformation matrix for an affine image transformation. This routine requires (among other) two input parameters XY1 and XY2 which provide the coordinates of some reference points in the original and result picture, respectively. In the *interactive mode*, the user is asked to type in the corresponding values. The *program generating mode* instead generates a calling statement for a special routine RDXY12 just before the calling statement for ACOE, but no concrete values are computed by the expert system. During the execution of the generated programm, RDXY12 will read in the values for XY1 and XY2 from the keybord.

IMPLEMENTATION

A prototype of our expert system has been implemented on a UNIX-machine using the expert system shell TWAICE by Nixdorf. Rules and frames are the main formalism for knowledge representation. The actual implementation covers about 90 subroutines which is approximately one third of the higher level SPIDER routines. The knowledge consists of about 600 rules and over 200 frames. Besides, there are two tables (for relating image processing algorithms with SPIDER subroutines). Our system also includes some Prolog and C procedures which are used for property inheritance of frames, interface to the subroutines, file handling, program generation, etc.

EXAMPLE

This example illustrates the application of our system on a digital image. As specified by the user, the system processed the image by a median filter smoothing, an affine transformation, a binarization, and a border detection. To perform this task, the SPIDER routines MEDI, ACOE, AFIN1, HIST1, THDS1, SLTH1, and BDR81 have been executed. The original and result pictures are shown in Fig. 3.

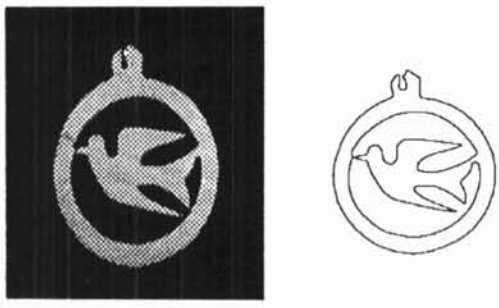

Fig. 3. Original and result picture

In the *program generating mode*, the following program has been generated by our system:

```
***  Program of consultation no. 1  ***
 PROGRAM Ptrbin
 DIMENSION IP(256,288)
 DIMENSION JPMEDI(256,288)
 DIMENSION IAFIN(256,230)
 DIMENSION JBINA1(256,230)
 DIMENSION JPBDR(256,230)
 DIMENSION IHST(256)
 DIMENSION XY1(2,3)
 DIMENSION XY2(2,3)
 DIMENSION T(2,3)
 DIMENSION RHIST1(256)
 DATA  ISX/256/
 DATA  ISY/288/
 DATA  IWX/3/
 DATA  IWY/3/
 DATA  NGR/256/
 DATA  ISXY/3/
 DATA  EPS/1e-6/
 DATA  KERR/0/
 DATA  ISX1/256/
 DATA  ISY1/230/
 DATA  ISW1/1/
 DATA  PTILE/79.0/
 DATA  ISWIT1/1/
 DATA  ISW/1/
 CALL FREAD ("taube",IP,ISX,ISY)
 CALL MEDI (IP,JPMEDI,ISX,ISY,IWX,IWY,IHST,NGR)
 CALL RDXY12 (XY1,XY2,ISXY)
 CALL ACOE (XY1,XY2,ISXY,EPS,T,KERR)
 CALL AFIN1 (JPMEDI,IAFIN,ISX,ISY,ISX1,ISY1,T,ISW1,KERR)
 CALL ERRCHK (KERR)
 CALL HIST1 (IAFIN,ISX1,ISY1,RHIST1,NGR)
 CALL THDS1 (RHIST1,NGR,PTILE,JTHRB1)
 CALL SLTH1 (IAFIN,JBINA1,ISX1,ISY1,JTHRB1,ISWIT1)
 CALL BDR81 (JBINA1,JPBDR,ISX1,ISY1,ISW)
 CALL BIOUT ("result",JPBDR,ISX1,ISY1)
 STOP
 END
***  Program ended  ***
```

CONCLUSIONS

An expert system for image processing based on the SPIDER subroutine package has been described in this paper. The two supported modes of operation allow an immediate experimental processing of a given picture as well as the construction of a program consisting of a number of SPIDER subroutine calls for later and repeated use. This system is a considerable help for users without any knowledge of the SPIDER package and without great experience in image processing.

The proposed structure of the knowledge base into general knowledge, knowledge about the application domain, and about the specific subroutine package can be considered as a generalized approach to expert systems for the application of subroutine packages. That part of our system which is independent of image processing and the SPIDER package can be applied as a shell for developing expert systems for other software packages and other application areas, too. Actually, we are testing the generality of our approach by adapting the expert system to the GIPSY image processing package [9].

ACKNOWLEDGMENT

We wish to thank Nixdorf Computer AG Switzerland for supporting our project by making us available a Nixdorf Targon /31 computer system including the expert system shell TWAICE.

REFERENCES

[1] H. Tamura, S. Sakane, F. Tomita, N. Yokoya, M. Kaneko, K. Sakaue, Design and Implementation of SPIDER - a Transportable Image Processing Software Package, Computer Vision, Graphics, and Image Processing 23, 1983, p. 273 - 294

[2] J.S. Bennett, L. Creary, R.S. Engelmore, SACON: A Knowledge-based Consultant for Structural Analysis, Stanford Computer Science Report CS-78-699 (1978)

[3] H. W. Gottinger, Statistical Expert Systems, Expert Systems 5/3, Aug. 1988, p. 186 - 196

[4] H. T. Bao, B. H. Khang, H. Kiem, RECTO: An Expert System in Classification and Recognition, Proceedings of the 8th Int. Conference on Pattern Recognition, Paris, October 1986, p. 657 - 659

[5] C. V. Apte, S. M. Weiss, An Expert System Methodology for Control and Interpretation of Applications Software, International Journal of Expert Systems, Vol. 1, Nr. 1, 1987, p. 17 - 37

[6] K. Sakaue, H. Tamura, Automatic Generation of Image Processing Programs by Knowledge-Based Verification, IEEE Proceedings on Computer Vision and Pattern Recognition, San Francisco, California, 1985, p. 189 - 192

[7] T. Toriu, H. Iwase, M. Yoshida, An Expert System for Image Processing, Fujitsu Scientific and Technical Journal 23/2 (1987), p. 111 - 118

[8] M. A. Billmers, M. G. Carifio, TVX: An Expert System Advisor for VMS, Digital Equipment Corporation, Hudson, Massachusetts, USA

[9] S. Krusemark, R.M. Haralick, V. Ramesh, C. Lee, UNIX GIPSY Users Manual, Intelligent Systems Laboratory, Dept. of Electrical Engineering, FT-10, University of Washington, Seattle WA 98195, USA, 1988